Administering the Domain/OS Registry

Order No. 015363-A00

Apollo Systems Division
A subsidiary of
HEWLETT
PACKARD

10 9 8 7 6 5 4 3 2 1

Preface

Administering the Domain/OS Registry describes the registry, a set of servers and tools used to control user accounts in a network environment. The book is intended for the administrator responsible for establishing and maintaining user accounts in a heterogeneous network environment.

We've organized this manual as follows:

Chapter 1	Provides an overview of the registry.
Chapter 2	Describes how to use the **edrgy** command to create and maintain registry accounts.
Chapter 3	Describes how to use the **edrgy** command to create and maintain the policies and procedures that govern registry objects.
Chapter 4	Describes how to perform routine maintenance procedures.
Chapter 5	Describes how to perform procedures that may be required if you reconfigure the network.
Chapter 6	Describes troubleshooting procedures.
Chapter 7	Describes how to set up the registry initially.
Appendix A	Describes the **import_passwd** command, used to make registry and nonregistry systems consistent.
Appendix B	Is a quick reference guide to using the **edrgy** command to create and maintain accounts and set policies and properties. Use it as a guide after you have read and understood Chapters 2 and 3.
Appendix C	Contains the manual pages for the commands and files associated with the registry.

Related Manuals

The file **/install/doc/apollo/os.***latest software release number***__manuals** lists current titles and revisions for all available manuals.

For example, at SR10.0 refer to **/install/doc/apollo/os.v.10.0__manuals** to check that you are using the correct version of manuals. You may also want to use this file to check that you have ordered all of the manuals that you need.

(If you are using the Aegis environment, you can access the same information through the Help system by typing **help manuals.**)

For a complete list of related Apollo documents, refer to the *Apollo Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242). For more information about how network configurations support the registry, refer to *Managing NCS Software* (011895). For more information about the consequences of merging master registries when you join networks, refer to *Managing Domain/OS and Domain Routing in an Internet* (005694). For more information about using the registry with your system software, refer to:

- *Using Your Aegis Environment* (011021)
- *Using Your BSD Environment* (011020)
- *Using Your SysV Environment* (011022)

You can order Apollo documentation by calling **1-800-225-5290.**

If you are using a Network Computing Kernel source or binary product on a system other than an Apollo system, see the accompanying release document, which may contain information pertinent to your particular system.

Does This Manual Support Your Software?

This manual supports Domain/OS Software Release 10.2 and 10.3. It was released with Domain/OS Software Release 10.3. To verify which version of operating system software you are running, type:

```
bldt
```

If you are running Domain/IX on a release of the operating system earlier than SR10.0, then type:

/com/bldt

If you are using a later version of software than that with which this manual was released, use one of the following ways to check if this manual was revised or if additional manuals exist:

- Read Chapter 3 of the release document that shipped with your product. The release document is online: **/install/doc/apollo/os.v.10.3__notes**. Check with your system administrator if you cannot find the release document.
- Telephone **1-800-225-5290**. If you are calling from outside the U.S., dial **(508) 256-6600** and ask for **Apollo Direct Channel**.
- Refer to the lists of manuals described in the preceding section, "Related Manuals."

To determine which of two versions of the same manual is newer, refer to the order number that is printed on the title page. Every order number has a 3-digit suffix; for example, **-A00**. A higher suffix number indicates a more recently released manual. For example, a manual with suffix **-A02** is newer than the same manual with suffix **-A01**.

Problems, Questions, and Suggestions

If you have any questions or problems with our hardware, software, or documentation, please contact either your HP Response Center or your HP Representative.

Alternatively, you can use the Reader's Response form at the back of this manual to submit comments about documentation.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values	In command formats and descriptions, words and characters in boldface type represent commands or keywords that you must use literally. Pathnames are also in boldface type. In text words in boldface indicate the first use of a new term.
user-supplied values	In command formats and descriptions, words in Italic type represent values that you must supply.

sample user input	In interactive examples, information that the user enters appears in boldface type.
`output/source code`	Information that the system displays appears in this `typeface`. Examples of source code also appear in this `typeface`.
[ ]	Square brackets enclose optional items in command formats and descriptions.
{ }	In command formats and descriptions, braces enclose a list from which you must choose an item.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
CTRL/ ^	The notation CTRL/ or ^ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.
. . .	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
⋮	Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.
——— ⊞ ———	This symbol indicates the end of a chapter or part of a manual.

——— ⊞ ———

Contents

Chapter 1 Overview

1.1 Introduction to the Registry 1–1
1.1.1 Automated Administrative Procedures 1–1
1.1.2 Levels of Administrative Control 1–1
1.1.3 Benefits of the Registry 1–2
1.2 Registry Servers and Clients 1–2
1.3 The Registry Database 1–3
1.4 How the Registry Database is Stored 1–4
1.5 Replicated Databases 1–5
1.6 How Database Updates Are Handled 1–6
1.6.1 Master and Slave Servers 1–7
1.6.2 Handling Database Updates 1–7
1.6.3 Propagating Database Changes 1–7
1.7 **/etc/passwd**, **/etc/group**, and **/etc/org** Files and the Registry 1–8
1.7.1 The **/etc/passwd**, **/etc/group**, and **/etc/org** Files under Domain/OS . 1–8
1.7.2 The **/etc/passwd**, **/etc/group**, and **/etc/org** Files on Workstations that Do Not Run Domain/OS 1–9
1.8 The Local Registry on Workstations that Run Domain/OS 1–9
1.9 Tools and Files for Administering the Registry 1–10

Chapter 2 Creating and Maintaining Accounts

2.1 Accounts in the Registry 2–1
2.2 Names in the Registry 2–1
2.3 Registry Owners and Access Control 2–3
2.3.1 Owner Rights Summary 2–3
2.3.2 Wildcards and Owner SIDs 2–4
2.3.3 Ownership Scenarios 2–4

2.4 Unique Identifiers (UIDs) and UNIX IDs 2–5
2.5 The **edrgy** Command 2–6
2.5.1 **edrgy** and Login Authentication 2–6
2.5.2 Default Ownerships 2–7
2.5.3 Reserved Names and Accounts 2–7
2.5.4 Name Formats 2–7
2.5.5 Number Formats 2–8
2.6 Using **edrgy** to Add and Maintain Persons 2–8
2.6.1 Example — Invoking **edrgy** and Adding Persons 2–8
2.6.2 Example — Creating Aliases 2–9
2.6.3 Example — Changing Persons 2–9
2.6.4 Example — Deleting Persons 2–10
2.7 Using **edrgy** to Add and Maintain Groups and Organizations 2–10
2.7.1 Group Passwords in the SysV Environment 2–11
2.7.2 Project Lists 2–11
2.7.3 Adding Groups and Organizations 2–12
2.7.4 Example — Adding a Group 2–12
2.7.5 Example — Adding an Organization 2–12
2.7.6 Changing Groups 2–12
2.7.7 Deleting Groups and Organizations 2–13
2.8 Using **edrgy** to Add and Maintain Accounts 2–13
2.8.1 Example — Adding Accounts 2–15
2.8.2 Example — Changing Accounts 2–15
2.8.3 Example — Deleting Accounts 2–16
2.9 Using **edrgy** to Maintain Membership Lists 2–16
2.9.1 Default Memberships Supplied with the System 2–16
2.9.2 Effects on Accounts of Deleting Members 2–16
2.9.3 Effects on Membership Lists of Account Creation 2–17
2.9.4 Example — Adding and Deleting Group Members 2–17

Chapter 3 Maintaining Policies and Properties

3.1 Policies 3–1
3.1.1 Password Expiration Date 3–1
3.1.2 Password Life Span 3–2
3.1.3 Password Format 3–2
3.1.4 Account Lifespan 3–2
3.2 Override Policies 3–3
3.3 Properties 3–3
3.3.1 Registry Owners 3–3
3.3.2 UNIX Restrictions 3–4
3.3.3 Read–Only Property 3–4
3.4 Handling Contradictory Registry and Organization Policies 3–4
3.5 The Effects of Changes on Existing Policies 3–5
3.6 Using **edrgy** to Set Policies and Properties 3–5

3.6.1 Example — Invoking **edrgy** and Displaying Current Policies and Properties 3–5
3.6.2 Example — Setting Registry Policy and Properties 3–6
3.6.3 Example — Setting Override Policy 3–7
3.6.4 Setting Organization Policy 3–7

Chapter 4 Performing Routine Maintenance Procedures

4.1 Adding New User Accounts 4–1
4.2 Creating Overrides 4–2
4.2.1 How Overrides Work 4–2
4.2.2 Override Policy 4–3
4.2.3 The **passwd_override** File Format 4–3
4.2.4 Creating Override File Entries 4–4
4.2.5 Using Wildcards in Subject Identifiers 4–5
4.2.6 Specifying Passwords for a Specific Machine 4–6
4.2.7 Preventing Log In to a Machine 4–6
4.2.8 Specifying a Home Directory and Log-In Shell for a Machine 4–6
4.2.9 Omitting Users from the Local Password Files 4–7
4.2.10 How **passwd_override** Handles Multiple Override Entries 4–7
4.2.11 Overrides and UNIX Commands to Change Account Information 4–8
4.2.12 Overrides and Domain/OS Commands to Change Account Information 4–8
4.3 Backing Up the Registry Database 4–9
4.3.1 Procedure Requirements 4–9
4.3.2 Procedures to Back Up the Registry Database 4–9
4.4 Ensuring Consistent Local Files 4–10
4.4.1 **passwd_refresh** Command Format 4–10

Chapter 5 Handling Network Reconfigurations

5.1 Changing the Master Registry Site 5–1
5.1.1 Procedure Requirements 5–1
5.1.2 Reversing the Master and Slave Sites 5–2
5.2 Removing a Node from the Network 5–2
5.2.1 Deleting a Slave Server 5–3
5.3 Joining Networks 5–3
5.3.1 Merging Registries — The **rgy_merge** Command 5–4
5.3.2 **rgy_merge** Syntax 5–5
5.3.3 Handling Conflicts 5–5
5.3.4 Running in **-compare** Mode 5–5
5.3.5 Using **rgy_merge** 5–6

Chapter 6 Troubleshooting Procedures

6.1 Re–Creating a Replica 6–2
6.2 Recovering the Master Registry 6–2
6.2.1 Turning a Slave into the Master 6–2
6.2.2 Turning a Master into a Slave 6–3
6.2.3 Restoring a Master Server to Its Original Site from a Backup 6–4
6.2.4 Restoring a Master Server to a New Site from a Backup 6–4
6.3 Forcibly Deleting a Registry Replica 6–5
6.4 Adopting File or Registry Orphans 6–5

Chapter 7 Setting Up the Registry

7.1 Steps to Set Up the Registry 7–1
7.2 Planning a Configuration 7–2
7.2.1 Registry Server Requirements 7–2
7.3 Ensuring Proper Location Broker Configurations 7–3
7.4 Creating a New Registry Database 7–4
7.5 Starting the Master Registry Server 7–4
7.6 Configuring the New Registry Database 7–4
7.6.1 Setting Policies, Properties, and Passwords 7–5
7.6.2 Adding Accounts 7–5
7.7 Creating Slave Registry Replicas 7–5
7.8 Verifying the Servers are Running 7–6
7.9 Establishing Uniform UNIX IDs 7–6
7.10 Establishing the Domain Mount Point Pathname 7–7
7.11 Restarting Registry Servers 7–7

Appendix A The import_passwd Command

A.1 How **import_passwd** Works A–1
A.1.1 Processing Steps A–2
A.1.2 Registry Entries Modified by **import_passwd** A–2
A.1.3 Registry Entries Created by **import_passwd** A–2
A.1.4 Resolving Conflicts A–3
A.2 **import_passwd** Syntax A–5
A.3 Using **import_passwd** A–6
A.3.1 Using Check Mode A–6
A.3.2 Answering Prompts A–6
A.3.3 Processing Prerequisites A–7
A.3.4 Synchronizing Domain UNIX IDs A–7
A.3.5 Synchronizing Foreign UNIX IDs A–7

A.4 Sample **import_passwd** Session A-7
A.4.1 Invoking **import_passwd** A-9
A.4.2 Examining the Group Entries A-10
A.4.3 Examining the Password File A-11
A.4.4 Adding Members to Groups A-13
A.4.5 Updating the Registry A-14

Appendix B Edrgy Quick Reference

B.1 Information Used by the **edrgy** Comand B-6
B.2 **edrgy** Command Examples B-6
B.2.1 Adding Accounts B-7
B.2.2 Adding Aliases B-7
B.2.3 Adding Groups B-7
B.2.4 Adding Members to Groups and Organizations B-8
B.2.5 Adding Organizations B-8
B.2.6 Adding Persons B-9
B.2.7 Changing Accounts B-9
B.2.8 Changing Defaults B-9
B.2.9 Changing Domains B-10
B.2.10 Changing Groups B-11
B.2.11 Changing Persons B-11
B.2.12 Deleting Accounts B-12
B.2.13 Deleting Aliases B-12
B.2.14 Deleting Groups and Organizations B-13
B.2.15 Deleting Members From Groups and Organizations B-13
B.2.16 Deleting Persons B-14
B.2.17 Displaying Accounts B-14
B.2.18 Displaying Groups and Organizations B-15
B.2.19 Displaying Persons B-16
B.2.20 Displaying Registry Policies and Properties B-17
B.2.21 Exiting from the **edrgy** Command B-18
B.2.22 Invoking the **edrgy** Command B-18
B.2.23 Setting Organization Policies B-19
B.2.24 Setting Registry Policy and Properties B-19

Appendix C Command and File Reference

edrgy C–2
rgy_admin C–11
rgy_create C–15
rgy_merge C–16
import_passwd C–17
passwd_override C–20
passwd_refresh C–25
pname_resolve C–26

Figures

1–1 Nodes, Servers, and the Database 1–3
1–2 Disk Memory and Virtual Memory Copies of the Registry Database ... 1–5
1–3 The Update Process 1–6

5–1 Joining Networks 5–4
5–2 Changed Names 5–8

Tables

1–1 UNIX Password and Group Files and Registry Database Comparison .. 1–4
1–2 Tools for Editing and Administering Registry 1–10

2–1 Types of Names 2–2
2–2 Rights of Owners of Registry Objects 2–4
2–3 Reserved Names and Accounts 2–7
2–4 Account Information Summary 2–14
2–5 Default Memberships 2–16

3–1 Strictest Policies 3–5

4–1 Summary of Wildcard Use 4–5

A–1 Effects of **import_passwd** Options on Conflict Resolutions A–5
A–2 Registry Group and Password Entries A–8
A–3 Foreign Group and Password File Entries A–9

B–1 Information Used by **edrgy** to Create and Maintain Persons B–2
B–2 Information Used by **edrgy** to Create and Maintain Groups and Organizations B–2
B–3 Information Used by the **edrgy** Command to Create and Maintain Accounts B–3
B–4 Standard Policy Information Used by the **edrgy** Command B–4
B–5 Override Policy Information Used by the **edrgy** Command B–5

B–6 Property Information Used by the **edrgy** Command B–5
B–7 Syntax to Change Domains in **edrgy** B–10
B–8 **edrgy** Syntax to Display Accounts B–14
B–9 **edrgy** Syntax to Display Groups and Organizations B–15
B–10 **edrgy** Syntax to Display Persons B–17
B–11 Syntax to Invoke **edrgy** B–19

Chapter 1

Overview

This chapter provides a brief introduction to the registry, including an overview of registry components.

1.1 Introduction to the Registry

The registry is an account registration system for controlling access to computers in a large heterogeneous network. Built on the Hewlett-Packard Network Computing System (NCS), the registry consists of replicated databases of account information, called **registries**, and servers to access those databases.

1.1.1 Automated Administrative Procedures

The registry meets the complex administrative needs of large heterogeneous networks. Typically, UNIX® account administration is handled by manually editing the **/etc/passwd** file. Consistency is maintained by periodically copying the file to each machine in the network. For large networks, this procedure places a heavy burden on system administrators. The registry reduces the administrative burden by controlling and automating the procedure.

Using registry tools, you access a common registry database from machines running on different platforms throughout the network. You can log in, change passwords, and update account information. The registry automatically propagates updates throughout the network, eliminating the need to manually copy the files.

1.1.2 Levels of Administrative Control

With the registry you can establish levels of administrative control over as many or as few machines as your needs dictate. With this ability, you can share administrative tasks among multiple administrators, each responsible for and in control of their own users. The level of administrative control that you implement can be as broad or as detailed as you require to provide a productive and efficient division of labor. Security mechanisms implemented in the registry allow mutually suspicious administrators to use a common machine while guaranteeing that the account data associated with their user community cannot be modified by other administrators.

1.1.3 Benefits of the Registry

The registry offers the following benefits to users:

- Replicated registry databases act as secure and highly available repositories of account information across the network.
- Users can have the same log–in ID, password, and group affiliations across different vendor platforms.
- Automated replication tools eliminate the need to manually replicate password files in heterogeneous networks.
- Authenticated operations allow secure updates from any machine in the internet or network.
- Access controls let you logically partition the registry database. A logically partitioned database allows multiple administrators to exercise independent control over person and account information.
- A registry override mechanism allows you to override certain registry information for individual machines thus preserving local autonomy.
- A person, group, organization structure for accounts provides a high level of control over data access.
- Tools for database management, including a structured editing tool and replication management tool, ease administrative burdens.

1.2 Registry Servers and Clients

The registry consists of registry databases, registry servers, and registry clients. The servers access the databases to perform queries and updates and to validate user logins. To access the registry database, nodes must run the registry client software. The registry client communicates with a registry server (**/etc/rgyd**) to request information and operations. For example, when you log in to a node, the client software on your node contacts a server and requests account information to verify your login. The server queries the registry database and passes the account information to the client. The client uses this information for login verification.

Server software runs only under the Domain/OS operating system. Client software runs under other operating systems and on other platforms. (See the release notes that came with your software for details on system compatibility.) Figure 1-1 is a simplified representation of the relationship between clients, servers, and a registry database.

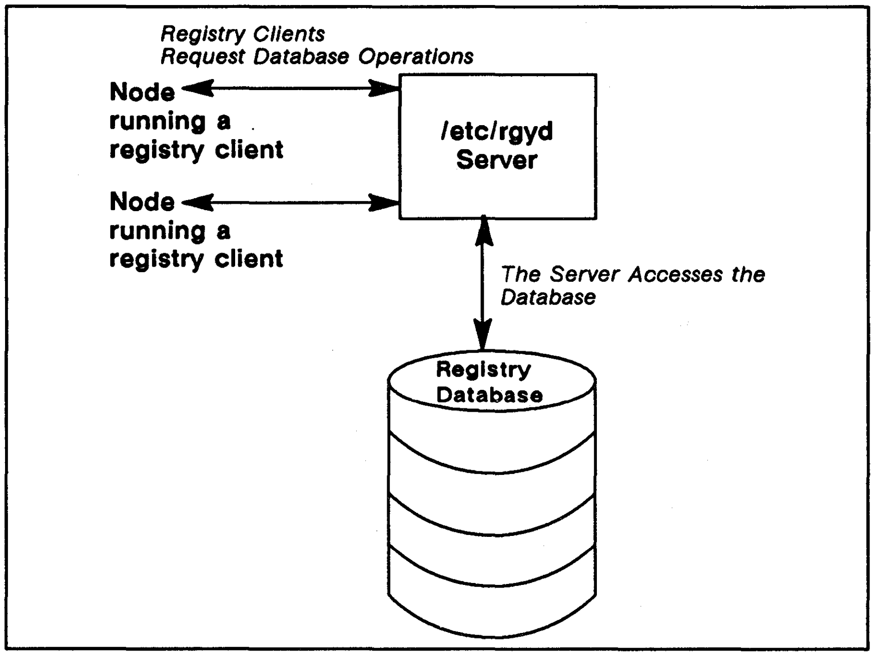

Registry Clients
Request Database Operations
Node running a registry client
Node running a registry client
/etc/rgyd Server
The Server Accesses the Database
Registry Database

Figure 1-1. Nodes, Servers, and the Database

1.3 The Registry Database

The registry database contains information similar to that found in UNIX group and password files. It contains:

- **Persons** — Who are users of the system.
- **Groups** — That are collections of users identified by a group name.
- **Organizations** — That are collections of users identified by an organization name. Like groups, organizations structure account data, generally for the purpose of access control.
- **Accounts** — That contain the passwords and accounting information that lets users log in to the system.

The registry also contains information on policies and properties, which regulate such things as password length and expiration and define such things as the owners of registry objects.

Table 1-1 compares the contents of the UNIX group and password files and the contents of the registry database. Note that the registry database stores more information than shown in Table 1-1. Chapters 2 and 3 describe the registry database in detail.

Table 1-1. UNIX Password and Group Files and Registry Database Comparison

UNIX Password File Contains	**Domain/OS Registry Contains**
Log-in name — User's login name	A **PGO triplet**, consisting of person (P) name, group (G) name, and organization (O) name, in the form: P{.G{.O}}. For example: **mahler.composers.classic**
Encrypted password — User's password	Same Information
User ID — Numerical user ID, a unique number assigned to the user	Same Information
Group ID — User's group ID	Same Information
GECOS — Text field	The user's fullname and freeform text in the form: fuiiname{,text}
Home directory — User's home directory	Same Information
Login shell — Name of shell to use at user login	Same Information
UNIX Group File Contains	**Domain/OS Registry Contains**
Group name — Name of the group	Same Information
Encrypted password — Group's password	Same Information
Group Id — Numerical group ID, a unique number assigned to the group	Same Information
Group members — List of group members	Same Information

1.4 How the Registry Database is Stored

Each registry server maintains a working copy of its database in virtual memory and a permanent copy on disk. All reads and updates operate on the copy in virtual memory. The servers use the copy on disk to initialize the copy in virtual memory when they start up. An atomic update log is used to guarantee the database state in the event of server failure.

Figure 1-2 shows the server and the disk and virtual memory copies of the database.

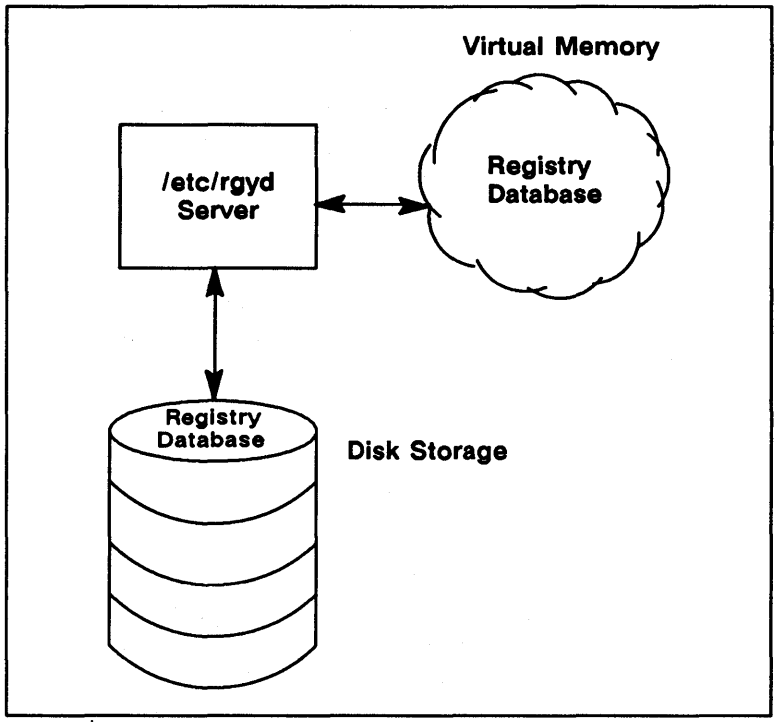


Figure 1-2. Disk Memory and Virtual Memory Copies of the Registry Database

Each registry server periodically saves its entire database from virtual memory to disk. The database is stored in the directory **/etc/rgy/rgy_data**.

1.5 Replicated Databases

The registry database may be replicated. Typically, you will create several replicated databases throughout the network. Replication of the registry database enhances performance and reliability, especially in an internet. Each replicated database must have an associated server to handle updates and queries against the database and to validate user logins. This document sometimes refers to the database as though it were one database, even though there may be several copies of the registry database.

The consistency of data in replicated databases is handled automatically by the registry servers. Any changes you make to a registry database are automatically reflected in all other registry databases as described in Section 1.6.

1.6 How Database Updates Are Handled

Updates are made to only one database and the changes are propagated to the replicas. While propagations are pending, all replicas are accessible even if they are not completely up-to-date. In other words, even replicas to which the changes have not been applied are available. This replication mechanism ensures that all replicas remain available to validate user login and for read operations even when changes are in the process of being propagated. Figure 1-3 illustrates the update process. The process is described in the sections that follow the figure.

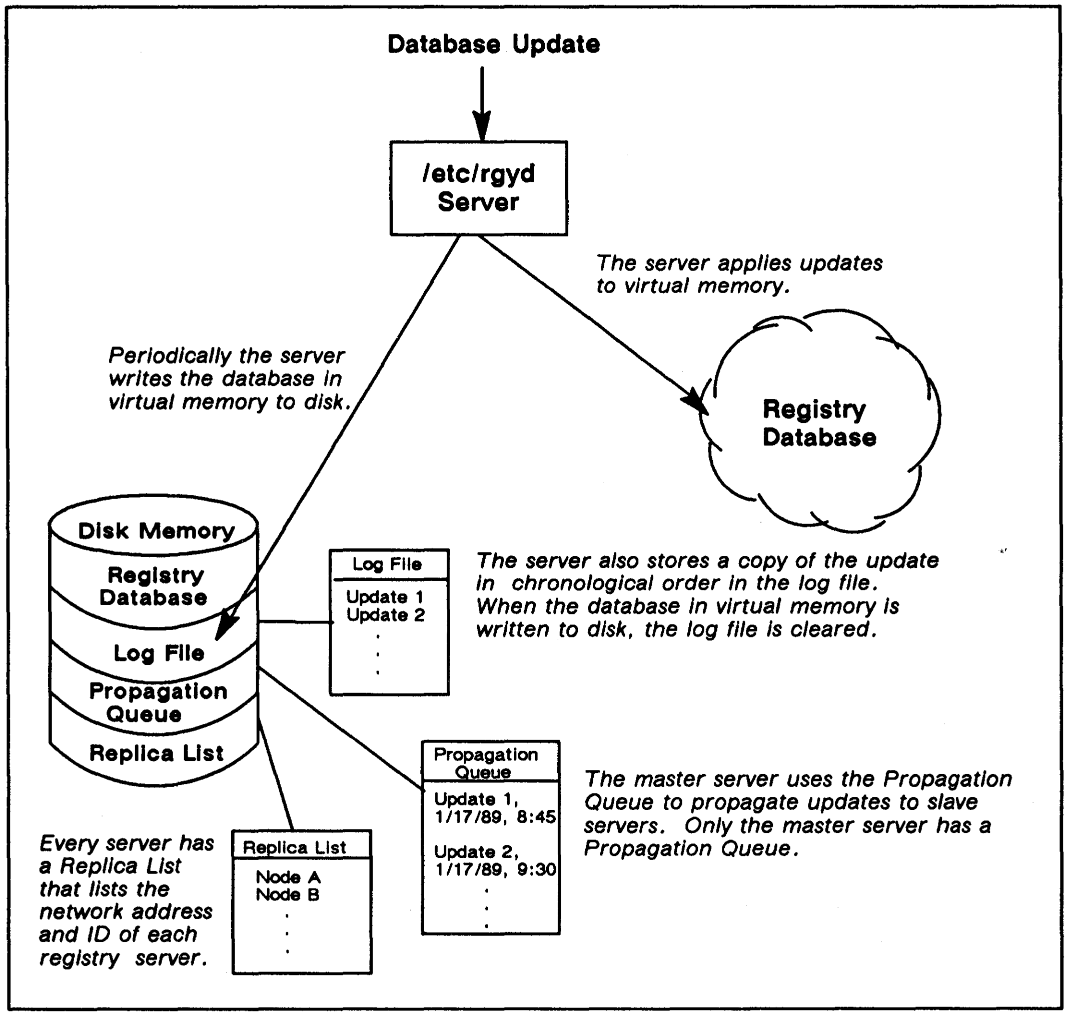

Database Update
/etc/rgyd Server
The server applies updates to virtual memory.
Registry Database
Periodically the server writes the database in virtual memory to disk.
Disk Memory
Registry Database
Log File
Propagation Queue
Replica List
Log File
Update 1
Update 2
The server also stores a copy of the update in chronological order in the log file. When the database in virtual memory is written to disk, the log file is cleared.
Propagation Queue
Update 1, 1/17/89, 8:45
Update 2, 1/17/89, 9:30
The master server uses the Propagation Queue to propagate updates to slave servers. Only the master server has a Propagation Queue.
Every server has a Replica List that lists the network address and ID of each registry server.
Replica List
Node A
Node B

Figure 1-3. The Update Process

1.6.1 Master and Slave Servers

Only a server designated as the **master server** will accept updates to its database. Other servers, called **slave servers**, will perform only reads on their databases. For example, either a master or a slave can provide account information to a client program such as **/bin/login**. However, if you are adding an account or changing password information, the database changes can be handled only by the master server. The master server updates its database and then propagates the changes to the slave server databases.

Except in highly unusual configurations called **cells**, there can be only one master server in a network or internet. (See *Managing NCS Software* for information about server configurations, including cell configurations.)

1.6.2 Handling Database Updates

When a master server receives updates, it follows the steps listed below:

1. The server applies the update to its database in virtual memory.

2. The server saves a copy of the update in a **log file** stored on disk. Updates accumulate in the log file in chronological order.

3. The server saves a copy of the update in a **propagation queue**, used to propagate the change to the slave servers as described in Section 1.6.3. The propagation queue contains all updates in chronological order that have not yet been delivered to the slave servers.

4. At some later time, the server writes the database in virtual memory to disk. When it performs a disk save, the server clears the log file.

When a server restarts, it initializes its database in virtual memory from the database on disk. It then applies any updates that remain in the log file to the database in virtual memory. This mechanism assures that no updates are lost when a server is shut down.

1.6.3 Propagating Database Changes

To propagate updates to the slave servers, the master server:

1. Updates its copy of the database using the process described in Section 1.6.2.

2. Marks both the database and the update with a timestamp.

3. Creates an indexed update for each slave in the propagation queue. The indexed update is the one the slave should receive next. In this way the unavailability of a single slave does not interfere with updates to the rest of the slaves.

4. Attempts to propagate the update to each slave server on its **replica list**, a list that contains the ID and network address of each host that runs a registry server. If the propagation of an update to one of the slaves does not succeed on the first attempt, the master server tries periodically until it succeeds. The master server always propagates updates in chronological order, according to their time stamps. It keeps track of which updates are pending propagation to which slaves.

5. Removes the update from the propagation queue when an update has been propagated to all the slaves.

If the master server loses communication with a slave server for a long time, the master server will reinitialize the slave server, giving it a fresh copy of the entire database, when communication is restored.

1.7 /etc/passwd, /etc/group, and /etc/org Files and the Registry

Versions of the **/etc/passwd**, **/etc/group**, and **/etc/org** files, the standard UNIX files for account registration, are maintained for all workstations in the network, including machines running Domain/OS. UNIX system calls such as **getpwent(3)** access the registry servers. If no server is available (for example if a registry server has not been started), these system calls access the **passwd**, **group**, and **org** files.

1.7.1 The /etc/passwd, /etc/group, and /etc/org Files under Domain/OS

Under Domain/OS, the **/etc/passwd**, **/etc/group**, and **/etc/org** files are specially typed file system objects used only to ensure compatibility with standard UNIX programs that require access to the files. When a user requests information stored in one of the files, the operating system finds the information in the registry database and presents it to the requester in the appropriate format. This information is then cached on the local disk to improve subsequent access.

If a registry server is not available to validate login, machines running Domain/OS use the local registry, described in Section 1.8.

The registry server automatically updates the information used to generate the **/etc/passwd**, **/etc/group**, and **/etc/org** files with any changes made to the registry database. On machines running Domain/OS, you cannot edit the **passwd**, **group**, and **org** files directly.

1.7.2 The /etc/passwd, /etc/group, and /etc/org Files on Workstations that Do Not Run Domain/OS

On workstations the do not run Domain/OS, the **/etc/passwd**, **/etc/group**, and **/etc/org** files are maintained in standard UNIX format (not as specially typed system objects as on machines that run Domain/OS). They are used to ensure compatibility with UNIX programs and, if a registry server is not available, to allow login.

To keep the **/etc/passwd**, **/etc/group**, and **/etc/org** files consistent with the registry database, the registry provides a utility called **passwd_refresh**. You should run this utility on a regular basis (preferably using **cron**). See Chapter 4 for details on **passwd_refresh**.

1.8 The Local Registry on Workstations that Run Domain/OS

Each machine that runs Domain/OS has a local registry that contains information about the machine's most recent users and the date and time they last logged in. If no registry server is available, the operating system checks the node's local registry for information about user's accounts. If the information exists in the local registry, the system allows the user to log in. (If both the network registry and the local registry are unavailable, the system always allows logn with the account name **user.none.none**.)

Provided a registry server is running somewhere on the network, Domain/OS automatically creates a local registry the first time a user logs into a node. Thereafter, it updates the local registry every time someone logs into the node. Users can edit some of the information in a node's local registry by invoking the **edrgy -l** command on that node.

1.9 Tools and Files for Administering the Registry

Table 1-2 lists tools and files for administering the registry.

Table 1-2. Tools and Files for Administering the Registry

Use the Tool...	**To...**
edrgy	Edit the registry database and on machines that run Domain/OS, the local registry. **edrgy** runs only under Domain/OS. Nearly all editing of the registry database must be done with this command. (The only exceptions are certain operations that users can perform on their own accounts, such as changing passwords.)
import_passwd	Create registry entries based on the group and password files from machines that do not run Domain/OS. Use this command if your Domain systems coexist with other UNIX systems that are not controlled by the registry, or as a first step to create registry entries when you bring a machine that does not run Domain/OS into an existing Domain network.
passwd_refresh	Create local password, group, and organization files from registry data on machines that do not run Domain/OS.
rgy_admin	List, reset, replace, and delete server replicas of the registry database and perform special functions such as changing the master server site and reinitializing a slave server.
rgy_merge	Merge data from registries that have previously operated independently.

Use the File...	**To...**
passwd_override	Override GECOS, home directory, and shell entries in the registry database for a local machine.
pname_resolve	Create and maintain a substitution string for pathnames to account for naming differences between file systems.

Chapter 2

Creating and Maintaining Accounts

This chapter describes how to use **edrgy** to create and maintain accounts. It first describes registry objects (persons, accounts, groups, and organizations) and then describes specifically how to create and maintain persons, groups, organizations, and accounts. Appendix B summarizes how to perform the tasks described in this chapter.

2.1 Accounts in the Registry

Accounts define who can log in to the computers on your network. They associate a person, group, and organization and include information that the operating system needs when a user logs in, such as the user's password and home directory.

The registry identifies each account by a **subject identifier (SID)** consisting of a person name, a group name, and an organization name, separated by periods. For example, **rubinstein.pianists.current** is a SID consisting of the person named **rubinstein**, the group named **pianists**, and the organization named **current**. We use the terms **pgo** and **pgo triplet** as synonyms for SID.

2.2 Names in the Registry

You must assign a name to each object in the registry. Although a person, a group, and an organization can have the same name, no two persons, groups, or organizations can have the same name, For example two persons cannot be named **smith**, but a person can be named **smith**, a group can be named **smith**, and an organization can be named **smith**.

You can assign up to three types of names as described in Table 2-1.

Table 2-1. Types of Names

This type of name...	Is used...	And can be assigned to a...
Primary	As the standard name associated with the object. This is the name used by most system utilities when a human-readable string is needed. When an object has multiple names (aliases), only one can be marked as primary. You must assign a primary name to each registry object. The primary name is a key field that you can use to query the registry database.	Person, group, or organization
Alias	As an optional alternate name. All aliases for a name share the same UID and UNIX ID. Aliases give you the flexibility to establish several accounts for the same person. For example, assume you create an alias of **gustav** for the primary name **mahler**. You can then create two accounts with different home directories and passwords: one for the person's primary name and one for the person's alias. Both accounts, however, can share the same access rights to files. The alias can also be used to add the person as a member in a group or organization. An alias name is a key field that you can use to query the registry database.	Person only
Full name	To allow easy recognition by users. A full name typically describes or expands the primary name. For example, a person could have a primary name of **jsbach** and a full name of **Johann S. Bach.** An organization could have the the primary name **rd** and the full name **Research and Development.** The person's full name concatenated with the account's user-supplied information is used to form the GECOS field in the **/etc/passwd** file and by UNIX library calls. A full name is a data field only. You cannot use it to query the registry database.	Person, group, or organization

2.3 Registry Owners and Access Control

Every object in the registry has an owner who is allowed to modify the object. (Account information such as the password and full name can be changed by the user of the account.) In addition to changing an object, an object's owner can change ownership of the object. In essence, an owner can "give away" ownership.

The registry uses the concept of ownership to help control access to the elements of the registry database and to provide a finely tuned level of control over accounts. Using owners, you can partition the user community into areas of administrative control each under the responsibility of an "owner" or administrator. All administrators are completely responsible for maintenance of their own areas. For example, only the owner of a person can create an account for that person, and only the owner of a group can add members to and delete them from the group. This ownership of registry objects allows the administrator to give (or restrict) control of registry objects to multiple administrators. These administrators then control the creation and maintenance of the objects they "own," while being restricted from modifying other portions of the database.

2.3.1 Owner Rights Summary

Table 2-2 summarizes the operations that can be performed by the owners of registry objects. Any of the operations described in the table can also be performed by a **root.%.%** or **%.locksmith.%** account, provided the user is logged in at the master registry node.

Note that the owner of the registry as a whole and the owners of the person, group, and organization domains are set as part of registry policies and procedures. Details on how to use **edrgy** to set policies and procedures are given in Chapter 3. Owners of individual persons, groups, and organizations are set when **edrgy** is used to create the person, group, or organization. **edrgy** can also be used to change ownership of persons, groups, and organizations. Those uses of **edrgy** are described in this chapter.

Table 2-2. Rights of Owners of Registry Objects

If you are the owner of...	**You can...**
The Registry	Change registry policies and properties. See Chapter 3 for information about changing policies. Change the owners of domains*. Use the **rgy_admin** and **rgy_merge** tools.
A Domain	Create entries in the domain. The owner of the person domain can create persons. The owner of the group domain can create groups. The owner of the organization domain can create organizations.
An Organization	Add and delete organization members. Change organization properties. Delete the organization.
A Group	Add or delete group members. Change information such as the password and full name. Delete the group.
A Person	Change the person's properties such as full name. Delete the person. Add or change an account for the person (provided the person is a member of the specified group and organization).
*The registry contains three domains to which you can assign owners: person domain, group domain, and organization domain.	

2.3.2 Wildcards and Owner SIDs

In SIDs that represent ownership, a percent character (%) can appear as a wildcard in place of a name. The wildcard character matches any registry object. For example, **mozart.%.%** matches **mozart.pianists.none** and **mozart.symphonists.classical**.

2.3.3 Ownership Scenarios

How you set up registry owners reflects the level of administrative control you want to implement. If you require no control over registry objects, assign the SID **%.%.%** as the owner of all objects or as the owner of the specific objects for which you require no control. This SID allows all accounts to have full ownership rights over the registry objects of which it is the owner. To model UNIX protection of the **/etc/passwd** file, use the SID **root.%.%**.

For moderate control, create a group specifically for registry administrators and assign the group as the owner of all or specific registry objects. For example, if you create a group named **rgy_admin** for registry administrators, you can assign the SID **%.rgy_admin.%** to registry objects. This SID ensures that only members of the **rgy_admin** group have ownership rights over the registry objects owned by **%.rgy_admin.%**.

To tightly control administrative access you can assign a fully specified SID. For example, if you assign **joe.rgy_admin.ics** as the owner of registry objects, only this specific account has ownership rights. So user **joe** could change these objects only when logged in as **joe.rgy_admin.ics**. He could not change them will logged in as **joe.dds.r_d**.

You can tailor administrative control to your specific needs. For example, assume that the registry administrator is an owner of the person domain and the payroll group administrator is the owner of the group **payroll**. Now assume that the user **franz** asks the registry administrator to create an account to allow him to access payroll information. When the registry administrator tries to create the account **franz.payroll.acct**, the system refuses the entry because the person **franz** is not a member of the **payroll** group and the registry administrator is not allowed to add to the group **payroll**. The payroll group administrator, who is an owner of the **payroll** group, must add **franz** to the group before the account can be created by the registry administrator.

2.4 Unique Identifiers (UIDs) and UNIX IDs

The registry automatically associates each primary name with a unique identifier (UID). In the registry and in Domain/OS, UIDs identify objects to the operating system, a function performed by UNIX numbers (UNIX IDs) in UNIX systems. (The registry also contains UNIX numbers, but they are used solely for compatibility with UNIX programs.) In this manual "UID" refers to Domain/OS Unique Identifiers; "UNIX ID" refers to UNIX numbers.

Normally, you do not have to be aware of UIDs. They are created and maintained automatically by the system. However, you should be aware that although the system prints names and you can access objects by name, the system identifies all objects by UID. If you delete a person from the registry, any registry objects (and the files) owned by the person are associated with an "orphaned" UID, that is a UID with no corresponding name. For registry objects, this means that no one has access rights to modify the objects owned by the deleted person. (For files this means that they can be accessed only according to the permissions previously given by the now deleted person.)

To solve this problem of orphaned registry objects, you must use the **edrgy adopt** command to associate the UID with a name and thus "adopt" the orphaned object. (UIDs are assigned by the system when the object is created using the **edrgy add** command. Therefore, you cannot simply add a new user and acquire a previously used UID. You must use **adopt** for this. Additionally, since there is no relationship between UNIX IDs and UIDs, you cannot re-create a deleted identity by creating a new user and assigning the user the deleted user's UNIX ID.)

2.5 The edrgy Command

The **edrgy** command is a structured editing interface to create and maintain registry information. Because it is aware of semantic and syntactic constraints placed on account information and of any policies that are in effect, **edrgy** can assist in performing semantically correct operations.

Using **edrgy**, you can add and maintain:

- Persons (users of the network)
- Groups and organizations (collections of users)
- Accounts (account identifiers in the form person.group.org that can log in to systems)

The remainder of this section describes the defaults used by **edrgy** for object owners, reserved names and accounts in the registry, and name and number formats in the registry. The remaining sections in this chapter show how to use the **edrgy** command and provide examples of adding and maintaining each type of registry object.

2.5.1 edrgy and Login Authentication

When you use the **edrgy** command to update the registry, **edrgy** checks to ensure that you have the appropriate ownership rights to update the registry object. (See Section 2.3 for a description of ownership rights.) If you do not have the appropriate rights, you cannot perform the update.

Your ownership rights are determined by the password you enter when you log in. If you logged in by creating a remote process (**crp -me**) or if you performed a remote login (**rlogin**) and did not supply a password, then there will be no way for **edrgy** to determine your ownership rights. Therefore, you will be unable to use **edrgy** to update the registry. If this happens, use the **login** command to log in again and supply your login name and password.

2.5.2 Default Ownerships

When you use **edrgy** to create a person, group, or organization, you are prompted for the object's owner:

```
Enter owner [p.g.o]: (%.sys_admin.%)
```

In the sample prompt above, the default owner, **%.sys_admin.%**, is shown in parentheses. As the registry is supplied, the default owner for persons, groups, and organizations is **%.%.%** (meaning that all users have rights to operate on all objects). You can use **edrgy** to change owners and owner defaults.

2.5.3 Reserved Names and Accounts

Some names and accounts are reserved for use by various system operations. You can change only the full name, password, and owner for reserved names and only the project list inclusion property for reserved groups. Table 2-3 lists reserved names and accounts.

Table 2-3. Reserved Names and Accounts

Names			Accounts
Person	**Group**	**Organization**	**Person.Group.Organization**
admin	backup	apollo	none.none.none
bin	bin	none	user.none.none
daemon	daemon	sys_org	sys_person.none.none
lp	locksmith		admin.none.none
root	login		daemon.none.none
sys_person	mail		bin.bin.none
user	none		lp.bin.none
uucp	server		uucp.daemon.none
none	sys		root.staff.none
	staff		
	sys_admin		
	sys_proj		
	wheel		

2.5.4 Name Formats

Names in the registry must:

- Begin with a lowercase letter
- Contain **only** lowercase letters, digits, or underscore characters
- Not exceed 32 characters in length, unless your system imposes UNIX restrictions, in which case they cannot exceed 8 characters in length

2.5.5 Number Formats

In the registry, numbers must range between 0 and 65535, inclusive. (We suggest that you not use numbers under 100, since these may be reserved in future releases.)

2.6 Using edrgy to Add and Maintain Persons

Persons make up the first part of the person.group.org subject identifier that identifies accounts. A person must exist in the registry before you can create an account for the person. Using the **edrgy** command, you can add, modify, and delete persons in the registry.

2.6.1 Example — Invoking edrgy and Adding Persons

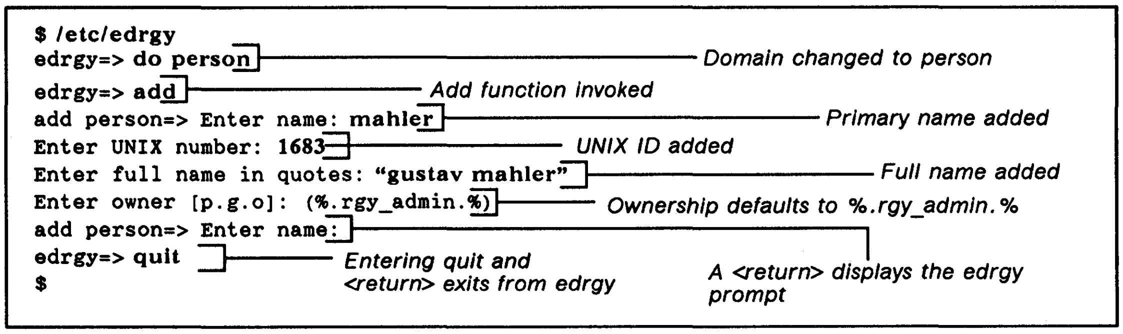
The following example shows the **edrgy** command used to add a person. The example consists of the following steps:

1. **edrgy** is invoked by entering:

   /etc/edrgy

2. The **do person** option is used to change the domain to the person domain. Since **edrgy** lets you add, change, and delete persons, groups, and organizations, you must indicate which domain you want to work in (person, account, group, or organization).

3. The person being added is assigned a primary name, UNIX ID, and full name. The person's owner is allowed to default to **%.rgy_admin.%**. (You can use the **edrgy** command default option to change the defaults displayed for **edrgy** prompts.)



```
$ /etc/edrgy
edrgy=> do person
edrgy=> add
add person=> Enter name: mahler
Enter UNIX number: 1683
Enter full name in quotes: "gustav mahler"
Enter owner [p.g.o]: (%.rgy_admin.%)
add person=> Enter name:
edrgy=> quit
$
```

Domain changed to person
Add function invoked
Primary name added
UNIX ID added
Full name added
Ownership defaults to %.rgy_admin.%
Entering quit and <return> exits from edrgy
A <return> displays the edrgy prompt

NOTE: Press CTRL/Q at any time to exit from **edrgy**.

2.6.2 Example — Creating Aliases

To create an alias for a person, invoke **edrgy** and then enter the **add** command in the following form:

add *person_name unix_ID* **-al**

Where

person_name	Is the alias name
unix_ID	Is the UNIX ID associated with the person for whom you are creating the alias
-al	Indicates that *person_name* is an alias.

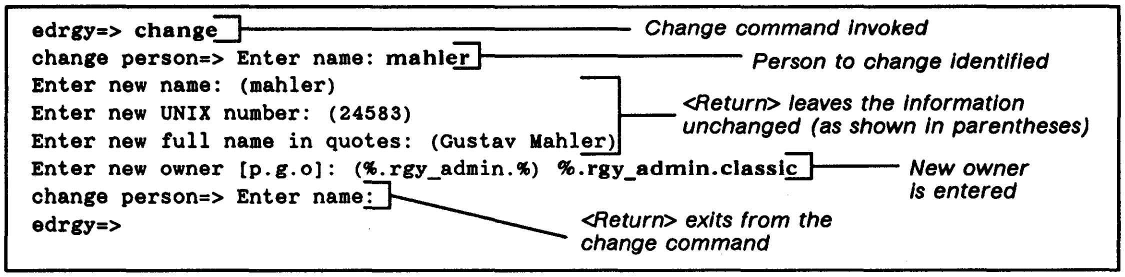
The **edrgy change** command (described in the following subsection) lets you make a primary name an alias and vice versa.

NOTE: When you create an alias, the system assigns the owner of the primary name as the alias owner. You can change alias owners with the **edrgy change** option.

2.6.3 Example — Changing Persons

You can change any of the information entered when a person was created (primary name, full name, UNIX ID, and owner). Additionally, you can change a primary name to an alias and an alias to a primary name. If you change a primary name to an alias and do not make an alias the primary name, the system will randomly choose an alias to be the primary name.

The following example shows the **edrgy change** command used to change the owner of a person from **%.rgy_admin.%** to **%.rgy_admin.classic**. The example assumes you are in the person domain.



```
edrgy=> change
change person=> Enter name: mahler
Enter new name: (mahler)
Enter new UNIX number: (24583)
Enter new full name in quotes: (Gustav Mahler)
Enter new owner [p.g.o]: (%.rgy_admin.%) %.rgy_admin.classic
change person=> Enter name:
edrgy=>
```

Change command invoked
Person to change identified
<Return> leaves the information unchanged (as shown in parentheses)
New owner is entered
<Return> exits from the change command

Effects of Changes

Changes to objects in the registry are immediately reflected in related objects. For example, name changes are reflected in any membership lists that use the name and in any owner SIDs. In addition, changing the UNIX ID of one of an object's aliases will change it for all of the object's aliases.

Objects that use registry information, but are external to the system, however, may require further action. For example, if you change a UNIX ID, files owned by the old number do not automatically show the new number as their owner. To remedy this on machines that run Domain/OS, execute the **syncids** command in the form:

/etc/syncids

To remedy this on other UNIX machines, use the **find** command. For example, if you change a person's UNIX ID from 333 to 444, you can run **find** in the following form to find all files owned by UNIX ID 333 and change the owner to UNIX ID 444.

find / -user 333 -exec chown 444 {} \;

Depending on how the files are protected, you may need to be root to run the **find** command on them.

2.6.4 Example — Deleting Persons

If you delete a person, the system automatically deletes any accounts for that person. For example, if you delete the person **mahler**, the accounts **mahler.symphonists.classic** and **mahler.composers.classic** will also be deleted. (Be aware that deleting a person could orphan the objects owned by the person/UID. See Section 2.4 for more details.)

The following example shows **edrgy** used to delete a person.

```
edrgy=> delete mahler ——————————————— The primary name is entered
Please confirm delete of primary name
"mahler" [y/n]: (n) y ———————————————————— Entering a y confirms
edrgy=>                                    the deletion
```

2.7 Using edrgy to Add and Maintain Groups and Organizations

Groups and organizations make up the second and third parts, respectively, of the subject identifier that identifies accounts. A group or organization must be in place before it can be used in a subject identifier.

2.7.1 Group Passwords in the SysV Environment

Each group can have a password. The password is relevant in a SysV environment only. In the SysV environment, if a group has been assigned a password, a user who is not a member of the group can acquire its privileges by invoking the **newgrp** command and entering that password. Many sites opt not to assign passwords to groups. Note that you can also assign passwords to organizations, but the acquisition of privileges does not apply to organization passwords.

2.7.2 Project Lists

As in the BSD environment, the registry allows a person to be associated with not just one group, but a set of groups, called a **project list**. The project list consists of all the groups in which the person named in the account is a member. The person has the access rights that accrue from membership in every group in the project list. For example, if group A can access file X and group B can access file Y, then persons with groups A and B on their project list can access files X and Y, regardless of which account they log in on.

Access Rights and Aliases

Persons accrue project list access rights only from the groups associated with the name or alias that they log in with. They do not accrue rights from all their names and aliases. For example, assume a person named **gustav** is a member of groups A and B. Under the alias **gus**, **gustav** is also a member of groups C and D. When the person **gustav** logs in, he accrues access rights from groups A and B only. He accrues access rights from groups C and D only when he logs in with the alias **gus**.

Prohibiting Inclusion on Project Lists

When you add groups, using the **edrgy** command, you can set a project list inclusion property that controls whether individual groups will be included in project lists. (Project lists do not apply to organizations.) You may decide to prohibit some groups from inclusion on the list. In Domain/OS, for example, the reserved group **locksmith**, which has access rights similar to root, is prohibited from inclusion on any project lists. Or, for example, if you are auditing access to certain files, you may want to prohibit the groups with access to those files from project lists in order to force a login with the audited accounts before users can access the files.

NOTE: To use project lists in Domain/OS, you must set the PROJLIST environment variable to true. (By default, PROJLIST is set to true automatically if your SYSTYPE is bsd4.3. See *Using Your Aegis Environment*, *Using Your BSD Environment*, or *Using Your SysV Environment* for more information about environment variables.)

2.7.3 Adding Groups and Organizations

Use the **edrgy** command to add groups and organizations. Either invoke the command in the appropriate domain (**edrgy -g** for groups or **edrgy -o** for organizations) or, if the command has been invoked, change to the appropriate domain (by entering **do group** for groups or **do org** for organizations). Once the group or organization is added, you can again use the **edrgy** command to add members as described in Section 2.9.

2.7.4 Example — Adding a Group

The following example shows **edrgy** used to add a group named **symphonists**. The example assumes you are in the group domain.

```
edrgy=> add
add group=> Enter name: symphonists
Enter UNIX number: 3332
Enter full name in quotes:  ]----------------  Full name and password
Enter new password:         ]                  are skipped with <return>
Include group on PROJLIST [y/n]? (y) y
                                               <Return> allows ownership to
Enter owner [p.g.o]: (%.rgy_admin.%) ]-------  default to %.rgy.admin.%
edrgy=>
```

2.7.5 Example — Adding an Organization

The following example is a continuation of the previous example. It first changes the domain to organization and then adds the organization named **classic**.

```
edrgy=>  do org         ]
Domain changed to: org  ]----------  Domain changed to organization
edrgy=> add
add org=> Enter name: classic                           Password is skipped
Enter UNIX number: 3333                                 with a <return>
Enter full name in quotes: "classic composers"                |
Enter new password: ]-----------------------------------------
Enter owner [p.g.o]: (%.rgy_admin.%) ]-------  <Return> allows ownership to default
edrgy=>                                        to %.rgy.admin.%
```

2.7.6 Changing Groups

You can change any of the information entered when a group or organization was created (primary name, full name, UNIX ID, and owner). In addition, for groups you can change whether or not the group can appear in project lists, and for organizations you can change policy. (See Chapter 3 for details on setting and changing organization policy.)

The following example shows the use of the **edrgy change** command to change the project list inclusion property from y (will appear in project lists) to **n** (will not appear in project lists). The example assumes you are in the group domain.

```
edrgy=> change
change group=> Enter name: symphonists ]---- Group to change identified
Enter new name: (symphonists)                        ]
Enter new UNIX number: (78)                          ]   <Return> leaves the infor-
Enter new full name in quotes: (classic composers)   ]-- mation unchanged (as
Enter new owner [p.g.o]: (%.rgy_admin.%)             ]   shown in parentheses)
Enter new password:                                  ]
Include group on PROJLIST [y/n]? (y) n ]--------------- Project list inclusion
change group=> Enter name:]-- <Return> exits from       property changed
edrgy=>                        the change command       from y to n
```

2.7.7 Deleting Groups and Organizations

If you delete a group or organization, you will also automatically delete any accounts that use the group or organization. For example, if you delete the group **symphonists**, you will also automatically delete the accounts **composers.symphonists.baroque** and **musicians.symphonists.classic**. The following example shows the **edrgy** command used to delete the group **symphonists** and then the organization **classic**.

```
edrgy=> do group           ]---- Domain changed to group
Domain changed to: group   ]
edrgy=> delete symphonists
Warning: any accounts for this group (%.symphonists.%)
will also be deleted.
Please confirm delete of primary name "symphonists" [y/n]: (n) y ]----
edrgy=> do org
Domain changed to: org ]---- Domain changed                     Entering a y
edrgy=> delete classic ]     to organization                    confirms the
                                                                deletion
Warning: any accounts for this org (%.%.classic) will also be de-
leted.
Please confirm delete of primary name "classic" [y/n]: (n) y ]----
edrgy=>
```

2.8 Using edrgy to Add and Maintain Accounts

Use the **edrgy** command to add accounts to the registry. (The person, group, and organization to be used in the account must exist in the registry before you can add the account.) Table 2-4 summarizes the account information stored in the registry.

Table 2-4. Account Information Summary

Information	**Example**	**How the Information is Used**
Subject Identifier	grappelli.violinists.jazz	To identify the account.
Abbreviation Type	p, pg, or, pgo	At login in place of the complete subject identifier. If you enter an abbreviation type of: • **p** = The user need enter only the person name to log in • **pg** = The user must enter the person and group name to log in • **pgo** = The user must enter the entire subject identifier to log in If the abbreviation type causes an abbreviation that is not unique, **edrgy** ignores the entry and assigns the shortest unique abbreviation possible. For example, if **mozart.composer.classic** exists with the abbreviation **mozart**, the shortest abbreviation **mozart.composer.none** can have is **mozart.composer**. **Note: Sendmail** requires the use of the **p** (person-only) abbreviation. Use it for all accounts that will use sendmail.
An encrypted password	sq1Rc1Urrb1L6	To help ensure log-in security. When users log in, they are prompted for a password. The system encrypts the password text string and checks the result against the encrypted string stored in the database. When you add an account, you supply a plain text password; the system performs the encryption.
A home directory	//walden/grappelli	When users log in, they are moved to their home directory.
A login shell	/bin/csh	To determine the shell to be executed when a user logs in.
Miscellaneous	login account for grappelli	A text string typically used to describe the use of the account.
An account-valid flag	Yes or No	To determine account validity. Invalid accounts cannot log in.
A password-valid flag	Yes or No	To determine whether the current password is valid. If this flag is set to No, the next time a user logs in to the account, the system prompts to change the password. (Note that this flag is separate from the password expiration policy, which sets time limits on password validity.)

2.8.1 Example — Adding Accounts

The following example shows the **edrgy add** command used to create the account **mahler.symphonists.classic**. The example assumes that you are in the account domain.

```
edrgy=> add
add account=> Enter account id [p.g.o]: mahler.symphonists.classic
Enter abbreviation type [p/pg/pgo]: (p) p
Enter new password: (-apollo-) sixth
Enter misc info in quotes: ()
Enter home directory: (/) //leitmotif/mahler
Enter shell: () /bin/csh
Password valid [y/n]? (y)                              <Return> accepts defaults
Enter expiration date [yy/mm/dd or 'none']: (none)     for these prompts
Account valid for login [y/n]? (y)
add account=>                                    <Return> exits from the add command
edrgy=>
```

2.8.2 Example — Changing Accounts

The following example shows the **edrgy change** command used to enter a new home directory (**//concert/mahler**) for the **mahler.symphonists.classic** account. The example assumes that you are in the account domain.

```
edrgy=> change
change account=> Enter account id [p.g.o]: mahler.symphonists.classic
Enter new abbreviation type [p/pg/pgo]:                      <Return> leaves the
Enter new password:                                          information unchanged
Enter new misc info in quotes:
Enter new home directory: //concert/mahler           New home directory entered
Enter new shell:
Password valid [y/n]?
Enter new expiration date [yy/mm/dd or 'none']:
Account valid for login [y/n]?
change account=> Enter account id [p.g.o]:     <Return> exits from the change command
edrgy=>
```

NOTE: If you change the group in an account SID, you must also explicitly change the account abbreviation to **p** (person-only) even if it is already coded as **p**.

2.8.3 Example — Deleting Accounts

The following example illustrates the use of **edrgy** to delete the account **mahler.symphonists.classic**. Note that the SID must be supplied as a parameter to the **delete** command. For example, to delete the account, you must enter the following command in response to the **edrgy=>** prompt:

delete mahler.symphonists.classic

```
edrgy=> delete mahler.symphonists.classic
Please confirm delete of "mahler.symphonists.classic" [y/n]: (n) y
edrgy=>
```

2.9 Using edrgy to Maintain Membership Lists

Each group or organization has a **membership list**, listing the persons who are members of the group or organization. Use **edrgy** to maintain membership lists.

2.9.1 Default Memberships Supplied with the System

Table 2–5 lists the default memberships supplied with the system:

Table 2–5. Default Memberships

The Person...	Is a member of	
	the Group...	and the Organization...
user	backup and sys_admin	Apollo
bin	mail	No Default
root	bin and sys	No Default

2.9.2 Effects on Accounts of Deleting Members

If you delete a member from a group or organization, any accounts that are made up of that user, account, and/or organization will also be deleted. For example, suppose you have two accounts: **mahler.symphonists.classic** and **mahler.composers.classic**. If you delete the person **mahler** from the group **symphonists**, then accounts that contain the person and group, such as **mahler.symphonists.classic**, will also be deleted. The account

mahler.composers.classic will remain unchanged. If you delete the person **mahler** from the organization **classic**, both accounts will be deleted.

2.9.3 Effects on Membership Lists of Account Creation

When you create accounts, the person named in the subject identifier must be a member of the group or organization named in the subject identifier. For example, if you create the account **mahler.symphonists.classic**, the person **mahler** must be a member of the **symphonists** group and the **classic** organization.

The **edrgy** command recognizes this requirement and, if you are the owner of a group or organization, will try to add the person to the group and organization. For example, assume you are the owner of the group **symphonists** and the organization **classic.** The person **mahler** is not a member of either. When you use **edrgy** to create the account **mahler.symphonists.classic, edrgy** automatically adds **mahler** to the **symphonists** and **classic** membership lists so that you can create the account in one step.

However, if you don't own the desired group, the command will fail. For example, suppose you do not own the specific group named **symphonists**. When you try to add **mahler.symphonists.classic, edrgy** will fail and display the message:

```
?(edrgy) Unable to update account "mahler.symphonists.classic" -
Not authorized to perform operation
```

Remember, however, if you log in to the master registry site as **root** or **locksmith**, you are allowed to perform any updates to the registry, regardless of ownership rights and restrictions.

2.9.4 Example — Adding and Deleting Group Members

The following example shows the **edrgy** command used to add **mahler** to and delete **strauss** from the group **symphonists**. The example assumes you are in the group domain.

```
edrgy=> view symphonists -m
symphonists    16984
   3 members:     brahms, britten, strauss
edrgy=> member
member=> Enter group name: symphonists
member=> Enter name to add: mahler
member=> Enter name to remove: strauss
Warning: any accounts for strauss.symphonists.% will be deleted.
Please confirm removal of "strauss" from membership list [y/n]: (n) y
member=> Enter name to remove:
member=> Enter group name:
edrgy=> view symphonists -m
symphonists    16984
   3 members:     brahms, britten, mahler
edrgy=>
```

view *group_name* **-m** *displays members of the named group*

The **member** *command adds a member by first prompting for the group to add the member to and then for the member name*

<Return> at this prompt initiates the deletion of a member from the list

<Return> exits from the member list edit

When the group is again viewed, you can see mahler has been added and strauss deleted

Chapter 3

Maintaining Policies and Properties

Registry policies regulate such things as overrides and certain account and password information. You can set policies both for the registry as a whole, and, with the exception of override policy, for individual organizations.

Registry properties specify such things as the owners of the registry and its domains. You set properties for the registry as a whole.

The chapter describes how to use the **edrgy** command to set and maintain registry policies and properties.

3.1 Policies

The **edrgy** command sets policies for specific organizations and for the registry as a whole. The policies you can set are described below.

3.1.1 Password Expiration Date

The password expiration date policy you set determines the exact date on which account passwords for a specific organization or for the registry as a whole will expire.

Password expiration dates are checked by the **login** command. If an account's password has expired, the next time the user logs in, **login** will invoke the **/bin/passwd** command, which requests that the user change the password. If the user does not change the password, login is denied.

If you define the password expiration date as **none**, the password will not have an expiration date.

NOTE: You can control individual account passwords by using the **edrgy add** or **change** command to set the account's password valid flag to N. In the same manner, individual accounts can be designated invalid by using the **edrgy add** or **change** command to set their account valid flag to N. See Section 2.8 for more information.

3.1.2 Password Life Span

The password life span you set determines the period of time for which account passwords for a specific organization or the registry as a whole will be valid. After the period of time passes, the account password expires.

If an account's password has expired, the next time the user logs in, the **login** command will invoke the **/bin/passwd** command, which requests that the user change the password. If the user does not change the password, login is denied.

You define the password life span as a number that indicates the number of days the password will be valid. If you define the password life span as **forever**, the password will have an unlimited life span.

3.1.3 Password Format

The password format policies determine the following for account passwords for a specific organization or the registry as a whole:

- The minimum length of passwords. If you define this policy, passwords cannot consist of fewer characters than the number you enter. If you enter 0 as the minimum length, no minimum length will be in effect.
- Whether passwords can consist entirely of spaces.
- Whether a password can consist entirely of alphanumeric characters.

3.1.4 Account Lifespan

The account life span you set determines the period of time for which accounts for a specific organization or the registry as a whole will be valid. After the period of time passes, the account becomes invalid and must be re-created.

You define the account life span as a number that indicates the number of days the account will be valid. If you define the account life span as **forever**, the account will have an unlimited life span.

3.2 Override Policies

You can establish overrides to the information contained in the network registry. The overrides are exceptions tied to a specific machine. For example, in the registry database, a user may have a home directory defined as **/home**. For a specific machine, you can establish an override that defines the home directory as **/new_home**. For more information on overrides see Section 4.2.

Override policy establishes whether or not the following types of overrides are allowed:

- **Password Exclusion** — Will you allow entry of nonvalid password strings as an override and thus prevent users from logging in to specific machines?
- **Root Password Overrides** — Will you allow the **root** password to be overridden for individual machines?
- **Nonroot Password Overrides** — Will you allow any user's password to be overridden for individual machines?
- **GECOS, Home Directory, and Login Shell Overrides** — Will you allow these registry entries to be overridden for individual machines?

In Domain/OS, the override policy you establish for **root** also applies to **%.locksmith**. If you allow **root** overrides, you are also allowing **%.locksmith** overrides. If you exclude **root** overrides, you are also excluding **%.locksmith** overrides.

3.3 Properties

The **edrgy** command sets properties for the registry as a whole. The properties you can set are described below.

3.3.1 Registry Owners

The registry owner property lets you name an account SID that will be the owner of the

- Registry as a whole
- Person domain
- Group domain
- Organization domain

Note that you use the **edrgy change** or **add** command to set the owner of individual person, groups, or organizations. See Chapter 2 for more information on the concept of owners in the registry.

3.3.2 UNIX Restrictions

The UNIX restriction property determines whether UNIX restrictions will be enforced. If you choose to enforce them:

- Names of registry objects cannot exceed eight characters.
- Accounts must have the **p** (person only) abbreviation.
- All password entries must use standard UNIX encryption.

3.3.3 Read-Only Property

The read-only property determines whether the master registry database is read-only. You can use it to allow no update operations to be performed on the master registry.

3.4 Handling Contradictory Registry and Organization Policies

All policies except override policies can be in effect for the registry as a whole or for individual organizations. If you set a policy for an organization and you also set the same policy for the registry, the stricter policy applies. For example, suppose registry policy specifies a minimum password length of six characters and policy for the organization named **classic** specifies eight characters. If you create an account named **bach.cantata.classic**, the stricter policy (in this case, the organization policy) applies and the account password must be at least eight characters long. Table 3-1 lists the strictest policy for each policy type.

Table 3-1. Strictest Policies

For This Type of Policy...	This Is the Stricter Policy...
Password Expiration	The shorter expiration period
Password Life Span	The shorter life span
Account Life Span	The shorter life span
Password Length	The greater length
Password Consisting of All Spaces	The password cannot consist of all spaces; it must include characters
Password Consisting of All Alphanumerics	The password cannot consist of all alphanumerics; it must include some nonalphanumeric characters

When the registry is created, policies are by default at their most permissive. To implement stricter policies, you must use the **edrgy prop** command to restrict them.

3.5 The Effects of Changes on Existing Policies

Changing policies affects only those registry objects created after the policy is changed. It has no effect on existing registry objects. For example, assume you have decided to change policy regarding passwords and enforce a longer length password. Existing passwords that are shorter than the length specified by the new policy are unaffected, and accounts with shorter passwords will not need to be changed. However, the next time an existing password is changed, the longer length policy will be enforced.

3.6 Using edrgy to Set Policies and Properties

The following sections show examples of using **edrgy** to set policies and properties.

3.6.1 Example — Invoking edrgy and Displaying Current Policies and Properties

The following example shows the use of the **edrgy prop** command to display current policies and properties.

1. **edrgy** is invoked by entering:

   /etc/edrgy

2. When the **prop** command is entered, **edrgy** displays current policies and properties and prompts for whether or not you want to make changes. Note that override policy for machines that run Domain/OS and machines that do not run Domain/OS is displayed separately.

```
edrgy=> prop ──────── entering the prop command initiates the
Registry Properties:  display of policies and properties
    Registry Owner: %.rgy_admin.ics
    Person   Owner: %.rgy_admin.%
    Group    Owner: %.rgy_admin.ics
    Org     Owner: %.rgy_admin.ics
    UNIX restrictions: are NOT enforced, are NOT met
    Registry is NOT read-only
  Registry Policy:
    Account lifespan:   forever
    Password min len:   0
    Password lifespan:  forever
    Password expiration date: none
    Passwords MAY be all spaces, MAY be all alphanumeric.
  Override Policy For "domain" Machines:
    Password exclusion is ALLOWED
    Root passwords MAY be overridden
    Non-root passwords MAY be overridden
    Non-password data MAY be overridden
  Override Policy For "non_domain" Machines:
    Password exclusion is ALLOWED
    Root passwords MAY be overridden
    Non-root passwords MAY be overridden
    Non-password data MAY be overridden               edrgy prompts for whether
Do you wish to make changes [y/n]? (n) ──────────  you want to make changes
```

3.6.2 Example — Setting Registry Policy and Properties

If you answer yes to the "Do you wish to make changes [y/n]? (n)" prompt, **edrgy** prompts for policies and properties. To make a change, simply enter it. To leave a value unchanged, press RETURN. (Current values are displayed in parentheses after the prompt.) In the example below, the account and password life spans are changed:

```
Do you wish to make changes [y/n]? (n) y ──────── y is entered to make changes
Enter new registry owner [p.g.o]: (%.rgy_admin.ics)
Enter new person owner [p.g.o]: (%.rgy_admin.%)
Enter new group owner [p.g.o]: (%.rgy_admin.ics)       ───── <Return> accepts the
Enter new org owner [p.g.o]: (%.rgy_admin.ics)               displayed default
Enforce UNIX restrictions [y/n]? (n)
Stamp registry read-only [y/n]? (n)                            Account lifespan is
Enter acct lifespan in days or 'forever': (forever) 365 ──── changed to 365 days
Enter password minimum length: (0)                                Password lifespan is
Enter password lifespan in days or 'forever': (forever) 180 ──── changed to 180 days
Enter password expiration date [yy/mm/dd or 'none']: (none)
May passwords be all spaces [y/n]? (y)
May passwords be all alphanumeric [y/n]? (y)
Do you wish to change override policy [y/n]? (n)
```

3.6.3 Example — Setting Override Policy

After you press **RETURN** at the "`May passwords be all alphanumeric [y/n]? (y)`" prompt, **edrgy** first prompts for whether you want to change override policy for machines that run Domain/OS and then override policy for machines that do not run Domain/OS.

In the following example, the root password override policy for all machines is changed to no.

```
Do you wish to change override policy [y/n]? (n) y <

Change override policy for "domain" machines [y/n]? (n) y  — y applies changes to ma-
Allow password exclusion [y/n]? (y)                          chines that run Domain/OS
Allow root password overrides [y/n]? (y)  n
Allow non-root password overrides [y/n]? (y)
Allow non-password data overrides [y/n]? (y)   n disallows the over-
                                               rides
Change override policy for "non_domain" machines [y/n]? (n) y  — y applies changes
Allow password exclusion [y/n]? (y)                              to machines that do
Allow root password overrides [y/n]? (y) n                       not run Domain/OS
Allow non-root password overrides [y/n]? (y)
Allow non-password data overrides [y/n]? (y)   n disallows the
                                               overrides
```

3.6.4 Setting Organization Policy

To change policy for an organization, follow the steps below.

1. Change to the organization domain and enter the **change** option and the name of the organization for which to set the policies. The following example specifies that policy for the organization named **classic** will be set.

```
edrgy=> do org
Domain changed to: org
edrgy=> change classic
```

2. **edrgy** first prompts for changes to the organization's full name, owner, and password. You can change this information or press RETURN to leave it unchanged. **edrgy** then prompts for whether you want to make changes to the organization's policy information.

```
Enter new name: (classic)
Enter new UNIX number: (12)
Enter new fullname in quotes: (No Organization)
Enter new owner [p.g.o]: (%.rgy_admin.ics)
Enter new password:
Do you wish to enter policy information [y/n]? (n)
```

3. Enter y at the "`Do you wish to enter policy information [y/n]? (n)`" prompt. **edrgy** prompts for the policy information. Enter the information to be

changed or press RETURN to leave it unchanged. The following example shows the password life span changed to 180 days. Every other policy is left unchanged by pressing RETURN.

```
Do you wish to enter policy information [y/n]? (n) y
Enter acct lifespan in days or 'forever': (forever)
Enter password minimum length: (0)
Enter password lifespan in days or 'forever': (forever) 180
Enter password expiration date [yy/mm/dd or 'none']: (none)
May passwords be all spaces [y/n]? (y)
May passwords be all alphanumeric [y/n]? (y)
edrgy=>
```

Chapter 4

Performing Routine Maintenance

This chapter describes registry maintenance procedures you will perform on a regular basis:

- Adding new users
- Creating overrides for individual machines
- Backing up the database
- Updating the local registry password, group, and organization files so they are consistent with the registry

4.1 Adding New User Accounts

To add new user accounts, invoke the **edrgy** command and then:

1. Add the person to be used in the account, if the person has not been added.
2. Add group to be used in the account, if it does not exist.
3. Add the person as a member of the group. (Note that if you are an owner of the group, the **edrgy** command does this automatically when you create the account.)
4. Add the organization to be used in the account, if it does not exist.

5. Add the person as a member of the organization. (Note that if you are the owner of the organization, the **edrgy** command does this automatically when you create the account.)

6. Add the account.

For details on how to use the **edrgy** command to add accounts, see Chapter 2.

4.2 Creating Overrides

You can override registry entries for local machines. Using overrides you can, for example, prevent individuals, groups, or organizations from logging in to a particular machine, establish local root passwords, and tailor local user environments. The override information is in effect for the local machine only and has no effect on the account information stored in the registry.

The override mechanism provides a high level of local autonomy and allows individual users to control their own machines. For example, an administrator responsible for a group of machines can use the override facility to restrict access to those machines. The administrator can allow access to a specific group only, or the administrator can allow access to everyone except specific groups or persons.

4.2.1 How Overrides Work

The **passwd_override** administrative file stored in the local machine's **/etc/rgy** directory contains override information. Using this file, you can enter overrides for a:

- Password
- GECOS information
- Home directory
- Log–in shell

The override information you enter is in effect only for the local machine—the machine on which the **passwd_override** file is stored. When a user logs in to a machine with an override file, any information for the user's account in the override file replaces the pertinent information obtained from the registry.

For example, assume that the registry entry for the **bach.cantata.classic** account specifies a Korn shell at login. Since the person who uses the account normally logs in to a machine that runs Domain/OS this is fine for a majority of the time. However, occasionally, this person works for another organization and logs in to a Sun machine that cannot produce a Korn shell. To accommodate this user's needs, you can create an override file on the Sun machine and specify that the login shell is a Bourne shell. If you do, when the user logs in to a machine that runs Domain/OS, registry data is used and a Korn login shell is invoked. However, when this person logs in to the Sun machine, override data is used and a Bourne shell is invoked.

4.2.2 Override Policy

The **edrgy properties** command sets the registry policy that specifies whether or not overrides will be allowed on machines that run Domain/OS and on machines running other operating systems. In order to use the override capabilities, override policy must be set to allow them. See Chapter 3 for more information on setting override policy.

4.2.3 The passwd_override File Format

Entries in the **passwd_override** file have the following format:

person.group.org:*passwd*:*unix_id*:*grp_id*:*GECOS*:*home_dir*:*shell*

where:

person.group.org	The subject identifier that identifies the account. Enter a subject identifier if you are creating overrides for a user's specific account or set of accounts. If you are creating overrides for all of the user's accounts, enter a UNIX ID. (If you enter a UNIX ID, it is as though you entered **person.%.%.** See Section 4.2.5 for a description of the use of **%**, the wildcard character. You must enter a person.group.org subject identifier or a UNIX ID to identify the account.
passwd	The encrypted password. If you specify an override, the password you enter here will be in effect for this local machine only. You can also specify "OMIT" in the *passwd* field to disallow login on the local machine. The use of OMIT in conjunction with an option to the **passwd_refresh** command prevents the inclusion of the user in the password file created by **passwd_refresh**. (See Section 4.2.9 for details.)
unix_ID	The UNIX ID that identifies the account. If you enter a UNIX ID, you must enter **%.%.%** for person.group.org. Enter a UNIX ID when you want to apply the overrides to all of a user's accounts (regardless of the group and organization) and to all of the user's aliases. UNIX IDs are especially useful for overrides to **root**. For example, if **root** has an alias of **wizard**, an override keyed by subject identifier applies only when **root** logs in as **root**. An override keyed by UNIX

ID applies when **root** logs in as **root**, as **wizard**, and under any other alias.

grp_id	The account's group ID. This field cannot be overridden and is ignored when the **passwd_override** file is processed. Normally you should leave this field empty.
GECOS	The account's GECOS field. If you specify an override, it is reflected in the information displayed by the UNIX **finger** command.
home_dir	The account's home directory. If you specify an override, the directory you specify will be the account's home directory on this machine only.
shell	The shell invoked when the account logs in. If you specify an override, the shell you specify will be invoked at login.

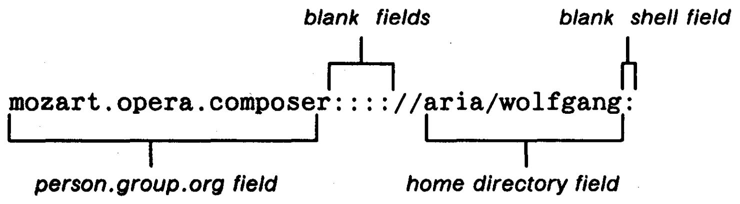
4.2.4 Creating Override File Entries

To create override file entries, edit the **passwd_override** file and follow the format described above to enter the override. You must supply either a subject identifier (*person.group.org*) or UNIX ID (*unix_ID*) to identify the accounts to which the overrides apply.

> **NOTE:** The UNIX commands to change account information (**passwd, chfn, chhd,** and **chsh**) have been modified to allow users to modify not only their registry entries but entries in the **passwd_override** file as well. See Section 4.2.11 for more information.

Leaving Fields Blank

If you don't want to override an item, leave its field blank, separating each blank field with a colon (:). For example, an entry to override the home directory for the account **mozart.opera.composer** looks like:



```
mozart.opera.composer::::://aria/wolfgang:
```

You must enter the colons associated with the blank fields, even if they are trailing, as in the example, above. In other words, you could *not* override **mozart's** password with the following entry:

```
mozart.opera.composer:clavier:
```

4.2.5 Using Wildcards in Subject Identifiers

You can use wildcards in place of the person, group, or organization in the subject identifier. If you specify each part of the subject identifier as in:

```
mozart.horn_concerti.composers
```

the override is restricted to that specific account. If **mozart** has other accounts, for example, **mozart.opera.composer** or **mozart.symphonists.composer**, these accounts are not affected by the override. To affect all of **mozart's** accounts, you could enter either **mozart.%.%** or the UNIX ID for **mozart**.

Table 4-1 summarizes the use of wildcards in override entries.

Table 4-1. Summary of Wildcard Use

The Entry...	**Affects...**
real_person.real_group.real_org	Only the account identified by the subject identifier
%.real_group.real_org	All accounts with the **real_group** group and/or the **real_org** organization
%.%.real_org	All accounts with the **real_org** organization
real_person.real_group.%	Only accounts for the user named **real_person** that specify group as **real_group**. For example, an override would apply to the accounts: • **real_person.real_group.real_org** • **real_person.real_group.pretend_org** • **real_person.real_group. none** It would not apply to the accounts: • **real_person.pretend_group.real_org** • **pretend_person.real_group.pretend_org** • **real_person.none.real_org**
real_person%.%	All accounts for the user named **real_person**

Wildcards as Defaults in Subject Identifiers

If you do not completely specify the subject identifier in an override entry, wildcards are assumed. For example, if you enter:

```
mozart:::::/rondo/mozart:
```

an entry of **mozart.%.%** is assumed for the subject identifier.

If you omit a trailing component, a wildcard is assumed. For example, if you enter **mozart.opera**, an entry of **mozart.opera.%** is assumed.

4.2.6 Specifying Passwords for a Specific Machine

You can use the **passwd** command with the **-l** option to enter a password for a specific machine into the **passwd_override** file on that specific machine. (See the **passwd** command manual page for details.) In addition you can edit the file on that machine and add the override password manually. The password you enter when you edit the file manually must be encrypted, but you can copy the encrypted password from the **/etc/passwd** file or you can write a program that generates encrypted programs. (The password you enter with the **passwd -l** command is not required to be encrypted.) The following example changes a specific machine's password for all of user **mozart's** accounts:

```
mozart.%.%:sqlRclUrrblL6:::::
```

NOTE: If your password is overridden and you then use **rlogin** or **rsh** to log in remotely to the machine with the overrides, you will be prompted for a password, regardless of what is in either the **/etc/hosts.equiv** or **.rhosts** file.

4.2.7 Preventing Log In to a Machine

To prevent users from logging in to a machine, create an override entry with an invalid string in the **passwd** field. Because the **passwd** field contains an encrypted password, any character string that is less than 13 or greater than 14 characters in length can be used as an invalid password.

For example, the following entry in the **passwd_override** file prevents users that are members of the organization named **evil_composers** from logging in.

```
%.%.evil_composers:exclude:::::
```

4.2.8 Specifying a Home Directory and Log-In Shell for a Machine

To change an account's home directory and log-in shell for a specific machine, create an override entry with a home directory name and a log-in shell name. For example, the following entry changes the home directory and log-in shell for all of user **mozart's** accounts:

```
mozart:::::/rondo/mozart:bin/sh
```

Note that because the SID contains only a person name (**mozart**), an entry of **mozart.%.%** is assumed.

4.2.9 Omitting Users from the Local Password Files

An invalid password entry in the **passwd_override** file prohibits users from logging in on the machine on which the file exists. However, the invalid entry **OMIT** has a special meaning. If you enter **OMIT**, the user cannot log in. In addition, on machines that do not run Domain/OS, you can specify, in the **passwd_refresh** command, that the user (or set of users) should also be omitted from **/etc/passwd**.

> **NOTE:** In Domain/OS, the TYPE manager controls the local password and group files, so use of **passwd_refresh** is not necessary. Therefore **passwd_refresh** is not shipped as part of the registry for Apollo machines.

Effects of Omitting Users

You should also be aware that if you have omitted users from the **/etc/passwd** file, information about those users will not be available to any programs that use the password file. For example, on machines that do not run Domain/OS, the **ls -l** and the **finger** commands both access the password file to obtain further information about a user identified by a UNIX ID. If the user is omitted, no password entry will exist and no information will be available. For this reason, you should omit users only if your user community is very large and either of the following conditions occur:

- The **passwd** file is taking up too much space
- User-ID-to-name mapping during, for example, **ls -l** is too slow

See Section 4.4 for how to use the **passwd_refresh** command.

4.2.10 How passwd_override Handles Multiple Override Entries

When more than one override entry applies to an account, the entry with the most specific key (either UNIX ID or person.group.org) is selected. Persons (person.%.%) are the most specific, followed by UNIX ID, group (%.group.%), and organization (%.%.org).

For example, assume the override file contains the following two entries that override the login shells:

```
mozart.%.%::::::/bin/ksh
%.opera.%::::::/bin/csh
```

If a person logs in with the subject identifier of **mozart.opera.composer**, the override keyed by **mozart.%.%** will be in effect. In this case, person (**mozart**) is more specific than group (**opera**).

Now assume the override file contains an additional entry:

```
mozart.%.%::::::/bin/ksh
%.opera.%::::::/bin/csh
mozart.opera.%::::::/bin/sh
```

Now when **mozart.opera.composer** logs on, the override keyed by **mozart.opera.%** will be in effect because person and group are more specific than person alone (**mozart.%.%**) and more specific than group alone (**opera.%**).

4.2.11 Overrides and UNIX Commands to Change Account Information

In UNIX environments, you can update account information using the following commands:

- **passwd** — To change passwords
- **chfn** — To change GECOS information
- **chhd** — To change home directories
- **chsh** — To change log-in shells

When you invoke these commands, you can supply an option (**-l** for local and **-n** for network) that tells the system whether you want to change the information stored in the network registry or in the local **passwd_override** file (the file on the machine from which you execute the command). If you do not enter an option and overrides exist for the information you are changing, you are prompted to re-invoke the command with one of the options. If no overrides exist for the information you are changing, you are not required to enter an option. The programs update the registry by default. See the **passwd** man page for details.

4.2.12 Overrides and Domain/OS Commands to Change Account Information

In Domain/OS, you can update account information using the following commands:

- **chpass** — To change passwords
- **chhdir** — To change home directories

- **edrgy** — To update all account information

These commands update only network registry information. They do not operate on registry overrides.

4.3 Backing Up the Registry Database

Because the registry server maintains current data in memory and saves the data to disk only periodically, you must use the exact procedures described here to back up the registry database. Use these procedures when you back up the disk containing the master database and when you back up only the database files. Only the master registry site needs to be backed up. When you restore the master, which is done only in the event of a catastrophic problem, the restored database will be propagated to the slaves automatically. (Note that slaves will refuse the maintenance request.) Restoring the master registry is described in Chapter 6.

4.3.1 Procedure Requirements

You must be an owner of the registry to perform Steps 3 and 5 (putting the registry into and out of maintenance mode).

4.3.2 Procedures to Back Up the Registry Database

To back up the registry database, follow the steps listed below.

1. Invoke **rgy_admin**:

   ```
   $ /etc/rgy_admin
   Default object:  rgy  default host : dds://music
   State:  in service  slave
   rgy_admin:
   ```

2. Ensure that the **rgy_admin** default host is the master registry site. (The name of the default host is displayed by **rgy_admin** when it is invoked.) If the master registry site is not the default host, enter:

   ```
   rgy_admin:  set -m
   ```

3. Put the master server in maintenance state:

   ```
   rgy_admin:  state -in_maintenance
   ```

   Putting a server in a maintenance state causes the server to save its database to disk and refuse all updates.

4. Back up the master registry by backing up either the entire volume or the **/sys/registry** tree. The exact commands you use for the backup are a matter of personal preference.

5. When the backup completes, take the master server out of maintenance state:

   ```
   rgy_admin: state -not_in_maintenance
   ```

   The server will resume accepting updates.

4.4 Ensuring Consistent Local Files

On machines that do not run Domain/OS, you should run the **passwd_refresh** command to make password, group, and organization files on local machines consistent with the registry database. On machines that run Domain/OS, the TYPE manager automatically controls local password, group, and organization files.

Run the **passwd_refresh** command on a regular, but staggered basis, preferably as part of **cron** processing. If **passwd_refresh** succeeds in creating the new password, group, and organization files, it saves the current files as backups named **passwd.bak, group.bak,** and **org.bak**. If it fails, it leaves the current files as is. (On diskless nodes, there is no need to run **passwd_refresh.**)

4.4.1 passwd_refresh Command Format

The **passwd_refresh** command has the following format:

passwd_refresh -f [*directory_name*] **-x**

where:

-f	Tells **passwd_refresh** to update the local password, group, and organization files even if no changes have been made to the registry. If you do not enter this option, the files are created only if changes have been made to the registry since the last time **passwd_refresh** was run on this machine.
[*directory_name*]	Specifies the name of a directory to store the local password, group, and organization files created by **passwd_refresh**. If you do not enter this option, the files are stored by default in the **/etc** directory on the local node. For example, to store the files in the directory called **/locals**, enter the command in the form:

```
$ /etc/rgy/passwd_refresh /locals
```

–x Prohibits the creation of entries for users with password overrides (on the local machine) that specify "OMIT" as their encrypted password. Use the –x option to exclude "omitted" users from the Password file created by **passwd_refresh**. To omit a user, you must create an override entry for the user and enter the word "omit" as the user's password field entry. Omitted users will be unable to log in to the local machine. (See Section 4.2.9.)

Chapter 5

Handling Network Reconfigurations

This chapter describes procedures for hardware and network reconfigurations. You will use these procedures when you

- Change the master registry site
- Remove a node from the network
- Join two or more networks, each with its own master registry

5.1 Changing the Master Registry Site

The node that runs the master registry server must be available at all times. If you are planning to remove this node from your network or to shut it down for an extended period, you should change the master registry site.

One efficient method for changing the master registry site is to reverse the roles of the master server and a slave server. In other words, make the master the slave and the slave the master.

Note that Chapter 6 also describes procedures to turn a master into a slave and a slave into a master. However the procedures described here are especially suited to instances of planned master server unavailability while the procedures in Chapter 6 are more suited to abnormal situations, including hardware or network failure.

5.1.1 Procedure Requirements

The procedures described in Section 5.1.2 assume that the registry servers at the master and slave sites are operating normally. In addition, you must be an owner of the registry in order to perform Step 4 of the procedure.

5.1.2 Reversing the Master and Slave Server Sites

1. Choose the new master site. A slave replica must exist at this site. If necessary, create the slave replica, as described in Chapter 7.

2. Invoke **rgy_admin**:

```
$ /etc/rgy_admin
Default object:  rgy  default host : dds://art
State:  in service  master
rgy_admin:
```

3. Ensure that the **rgy_admin** default host is the current master registry site. (The name of the default host is displayed by **rgy_admin** when it is invoked.) If the master registry site is not the default host, enter:

```
rgy_admin:  set -m
```

4. Issue the **change_master** command to reverse the roles of the master and slave. The example below makes **dds://music** the new master.

```
rgy_admin:  change_master -to dds://music
```

5. Verify that the master site has changed by issuing the **lrep** command:

```
rgy_admin:  lrep -state
   (master) dds://music  state:in service  1990/11/16.12:46:59
            dds://art    state:in service  1990/11/16.12:46:59
            dds://lit    state:in service  1990/11/16.12:46:59
```

Note that this command displays "(master)" to the left of the master registry site name.

5.2 Removing a Node from the Network

If you are planning to remove a node that runs a slave registry server from the network or to shut it down for an extended period, you should delete the registry server at that site. If you are removing a node running the master server, you must change the master server site as described previously before you remove the node.

5.2.1 Deleting a Slave Server

The steps to delete a slave server follow. (Note that you must be an owner of the registry in order to perform Step 3 of this procedure)

1. Invoke **rgy_admin**:

```
$ /etc/rgy_admin
Default object:  rgy  default host : dds://music
State:  in service  master
rgy_admin:
```

2. Because you must delete slaves from the master site, ensure that the **rgy_admin** default host is the current master registry site. (The name of the default host is displayed by **rgy_admin** when it is invoked.) If the master registry site is not the default host, enter:

```
rgy_admin:  set -m
```

3. Issue the **delrep** command. The example below deletes the registry server from **dds://lit**

```
rgy_admin: delrep dds://lit
```

   When you issue this command, the master server instructs the slave replica to delete itself.

4. Verify that the slave has been deleted by issuing the **lrep** command:

```
rgy_admin:  lrep -state
   (master) dds://music  state:in service  1990/11/16.12:50:39
            dds://art    state:in service  1990/11/16.12:50:39
```

   The deletion may take a while. Until the slave is actually deleted, **lrep -state** will show the slave as marked for deletion.

5.3 Joining Networks

A network or internet can have only one master registry. Therefore, if you connect networks with independently operating master registries, the registries must be merged into one master registry. For example, Figure 5-1 shows Network A consisting of networks 1230 and 1231 and Network B consisting of network 1233. The master registry is at **//music** in Network A and at **//art** in Network B. If you join Networks A and B, the resulting Network C will have two master registry sites, which must be merged into a single site.

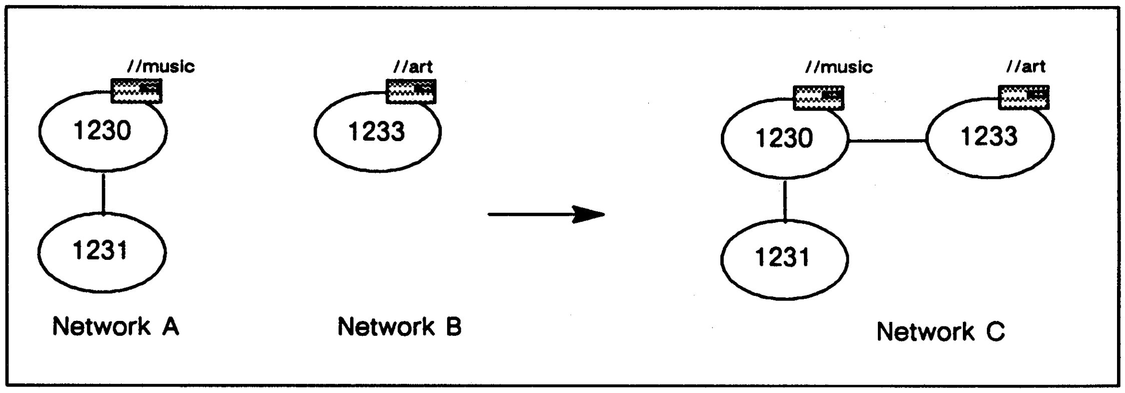


Figure 5–1. Joining Networks

NOTE: When you form an internet by partitioning one network into several networks, the master registry in the original network becomes the master for the internet. Therefore, the registries do not need to be merged.

This section describes how to use the **rgy_merge** command to merge master registries when you join networks that each have their own master registries. Note that you should read *Managing Domain/OS and Domain Routing in an Internet* for information about other considerations that will arise when networks are joined.

5.3.1 Merging Registries — The rgy_merge Command

Use the **rgy_merge** command to join registries. You must invoke **rgy_merge** while logged in as root at the **target site**: the site that will become the new master registry site. Then, you merge all other registries, called **source sites**, into the master.

Before it merges the registry databases, **rgy_merge** compares their contents. If two or more entries have the same name or UNIX ID, **rgy_merge** reports the conflict and does *not* perform the merge. You must resolve the conflict by changing the database at the source or the target *before* you can continue merging registries.

If **rgy_merge** finds no conflicts, it merges the registries. As a result of the merge:

- There is a single master registry for the network or internet. All other registries are slave replicas of the master.
- The registry supports login from anywhere in the network or internet.
- The replica lists are complete, and the replicas' network numbers are correct.
- The policies and properties, including override policies, in place at the target site are used for the new master.

5.3.2 rgy_merge Syntax

The **rgy_merge** command has the following syntax:

```
rgy_merge -from //source_site [ {-merge | -compare} -verbose ]
```

where:

-from *//source_site*	Identifies the node where the registry to be merged exists.
-merge	Indicates that **rgy_merge** should perform the merge (if no conflicts exist) without prompting for confirmation.
-compare	Indicates that **rgy_merge** should simulate a processing run, showing conflicts that will occur, but without performing the merge.
-verbose	Indicates that **rgy_merge** should run in verbose mode and generate a transcript of all activity.

5.3.3 Handling Conflicts

When **rgy_merge** runs, it examines the target and source database for duplicate names and duplicate UID numbers. **rgy_merge** can find two different types of conflicts:

- **Name Conflicts.** These conflicts arise when the same name string is defined in both the target and source registry.
- **UNIX ID Conflicts.** These conflicts arise when the same UNIX ID is defined in both the target and source registry for a person, group, or organization.

If **rgy_merge** reports duplicate names or UNIX IDs, you must change one of the names or IDs before you can continue the merge. You can make the changes in either the source or target database. The new number must be unique not only within the current target–source pair, but, to avoid future conflicts, within all subsequent target–source pairs.

5.3.4 Running in –compare Mode

You should first run **rgy_merge** in compare mode with the **–compare** option. In this mode, **rgy_merge** simulates the results of a processing run showing all conflicts. When **rgy_merge** is run without the **–compare** option, it halts and exits whenever it encounters an error. You must use **edrgy** to fix the error and then re–execute **rgy_merge**. Eliminating conflicts in advance will save you time.

To run **rgy_merge** in compare mode, follow the procedures described in Section 5.3.5, with one exception. When you enter the command, enter it in the form:

rgy_merge -from *//source_site* **-compare**

5.3.5 Using rgy_merge

You must execute **rgy_merge** from the node that has the registry that will become the new master registry. This registry is called the **target** of the merge. The master registry named in the **-from** *//source_site* argument is the **source** for the merge. After you run **rgy_merge**, a source is no longer a master; it becomes a slave of the target.

For example, if you select **//music** as the target (by executing **rgy_merge** at **//music**) and name **//art** in the **-from** *//source_site* argument, **//music** becomes the master registry and **//art** becomes a slave registry.

If you are merging more than one source registry, you must complete the merge in steps, one target-source pair at a time.

Requirements to Use rgy_merge

To use **rgy_merge**, you must be able to log in to the target node as root and log in to each source node as the owner of the source registry.

Invoking rgy_merge

1. Identify the nodes that run master registry servers and select one node as the node to run the master registry server (called the target in these procedures).

2. Log in to the target node as root.

3. Invoke **rgy_merge**, specifying the source site in the **-from** option. In the example below, **dds://music** is the source site.

```
%$rgy_merge -from dds://music
```

rgy_merge prompts you to log in with an account that owns the source registry.

4. Enter the log-in name and password of the account that owns the source registry:

```
Source Registry Login:  root
Password:
Source registry has been made read-only.
Placing master registry in maintenance mode...
```

5. What happens next depends on whether or not **rgy_merge** finds conflicts between the target and source databases.

   If there are no conflicts, **rgy_merge** asks to merge the source with the target. Answer affirmatively, as shown below:

   ```
   Merge successful; update registry? y
   ```

   rgy_merge then completes the merge and exits. The source registry becomes a slave replica of the target registry. If there are more registries to merge, repeat these procedures.

   If there are conflicts, follow the steps in "Resolving Registry Conflicts," below.

Resolving Registry Conflicts

If **rgy_merge** finds conflicts, it will display them. The following example shows the information displayed when **rgy_merge** finds that both the source and target database contain a person named **smith**.

```
Source registry has been made read-only.
Placing master registry in maintenance mode...
Collision on person:
Source: name = "smith" uid = 3c2b6022.60001ed4 unix id = 99
Dest:   name = "smith" uid = 3c2b7ea2.90012edf unix id = 101
Merge failed; 1 errors.
Since merged registry was not updated, source registry has been reenabled
for updates.
%
```

To resolve the conflict, use the **edrgy** command to eliminate the conflict by changing the conflicting name or UNIX ID, as appropriate, in either the source or target database. Then begin the merge process again, following the instructions described in "Invoking **rgy_merge.**" Using the **edrgy** command is described in Chapter 2.

If you make any changes to UNIX IDs, you must run the **/etc/syncids** tool on every registry-controlled node in your network. This tool ensures that the changes are synchronized with the UNIX IDs stored on disk. On nodes that have more than one disk volume, you must run **syncids** on each volume.

If the registry is a slave replica, before you run **syncids**, you should use the **lrep -state** command in the **/etc/rgy_admin** tool to verify that all slave replicas are up-to-date. This command shows the timestamps of the most recent changes to each replica. You should not run **syncids** until the timestamps for all replicas match the timestamp for the master.

See the online manual pages for information about **syncids**.

Keeping Track of Changes

You should keep track of the names you change so that you can inform users of the changes. You can use the form shown in Figure 5-2 to keep a list of names you change.

Current Name	Changed Name	Current Name	Changed Name

Figure 5-2. Changed Names

Updating the Master Registry Replica List

After you've merged all sources into the target, use the **rgy_admin lrep** command to examine the **rgyd** replica list at the new master registry site. Ensure that all slave registries are in the master's replica list and that they are listed with the correct network numbers. If necessary, use the **rgy_admin chrep** command to update them.

Chapter 6

Troubleshooting Procedures

This section contains procedures for troubleshooting the registry. You should use these procedures only when network or hardware failures have disrupted operation of the registry or you encounter problems that can be remedied in no other way. These procedures include how to:

- Recreate a registry replica.
- Recover the master registry.
- Forcibly delete a replica.
- Adopt registry objects that have been orphaned because their owner has been deleted. (These objects will appear to you as objects that you are unable to modify.)

There are two registry commands not described in this manual that you can also use for maintenance and troubleshooting. These commands are:

- **lb_admin** — A tool that monitors NCS servers (such as registry servers) registered with the NCS Location Broker.
- **drm_admin** — A tool that administers replicated servers (again, such as registry servers).

For more information on these commands, exmine their manual pages or consult *Managing NCS Software*.

6.1 Re-Creating a Replica

Use the steps listed below to re-create a slave replica that is corrupted or irrecoverably out of date.

1. Log in as root on the node where you want to re-create the slave replica.
2. Use the UNIX **ps** command or the Aegis **pst** command to check whether a registry server (**/etc/rgyd**) is running. If there is one, use the **rgy_admin stop** command to stop it. (You must be an owner of the registry to use the **stop** command.)
3. Re-create the slave replica entering:

   ```
   /etc/server -p /etc/rgyd -recreate &
   ```

   This command destroys the existing database, creates a new one, and starts a slave server.
4. Verify that **rgyd** will start automatically at system startup by using the UNIX **touch** command or the Aegis **crf** command to create a file named **rgyd** in **/etc/daemons**:

   ```
   touch /etc/daemons/rgyd
   ```

   ```
   crf /etc/daemons/rgyd
   ```

6.2 Recovering the Master Registry

If the master registry replica becomes inoperable or unavailable for an extended time, you may need to replace it or restore it. This section contains several procedures for recovery of the master replica. You should read through all of them and select the one most appropriate for your particular situation.

6.2.1 Turning a Slave into the Master

If a slave replica with a nearly current database exists, you can use the following procedure to turn this replica into the master. (Check the timestamps of slave databases using the the **rgy_admin lrep -state** command.) If no such slave exists, restore the registry database from a backup.

You must be an owner of the registry to perform Step 4 of this procedure.

1. Choose the slave replica that will become the new master.

2. Log in to any machine and invoke **rgy_admin**:

   ```
   $ /etc/rgy_admin
   Default object:  rgy  default host : dds://music
   State:  in service  slave
   rgy_admin:
   ```

3. If necessary, issue the **set** command to set the chosen slave to the default host. The following example sets the default host to **dds://art**:

   ```
   rgy_admin: set -h dds://art
   ```

4. Issue the following **become** command to change the default host to the master registry:

   ```
   rgy_admin: become -master
   ```

5. Use **lrep -state** to verify the change. If the old master replica becomes available again, immediately change either the old or new master to a slave replica, to prevent inconsistencies.

6.2.2 Turning a Master into a Slave

Use this procedure to turn a master replica into a slave. You should use this procedure only if you have more than one master running on your network or internet, a highly unusual condition. (If you are performing a merge of two disjoint registries, two masters will coexist temporarily, but **rgy_merge** will turn one of them into a slave.) You must be an owner of the registry in order to perform Step 2 of this procedure.

1. Choose the master replica that will become a slave.

2. Log in to any machine and invoke **rgy_admin**:

   ```
   $ /etc/rgy_admin
   Default object:  rgy  default host : dds://music
   State:  in service  slave
   rgy_admin:
   ```

3. If necessary, issue the **set** command to set the chosen master to the default host. The following example sets the default host to **dds://lit**:

   ```
   rgy_admin: set -h dds://lit
   ```

4. Issue the following **become** command to change the chosen master to a slave:

   ```
   rgy_admin: become -slave
   ```

5. Use **lrep -state** to verify the change.

6.2.3 Restoring a Master Server to Its Original Site from a Backup

Perform the following steps to restore a master registry from a backup to its original location. See Section 6.2.4 to restore a master to a different location.

1. Log in as **root** on the master site node.

2. Use the UNIX **ps** command or the Aegis **pst** command to check whether a registry server (**/etc/rgyd**) is running. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry to use the **stop** command.)

3. Restore the **/sys/registry** tree from the backup media.

4. Restart the server:

   ```
   /etc/server -p /etc/rgyd &
   ```

5. Verify that **rgyd** will start automatically at system startup by using the UNIX **touch** command or the Aegis **crf** command to create a file named **rgyd** in **/etc/daemons**:

   ```
   touch /etc/daemons/rgyd
   ```

   ```
   crf /etc/daemons/rgyd
   ```

6.2.4 Restoring a Master Server to a New Site from a Backup

Perform the following steps to restore a master replica from a backup to a new location. It is likely that you will need to perform this procedure only if no other replicas are running that can be made the master. The procedure assumes this is true and that no other replicas are running.

1. Log in as **root** on the node that will be the new master site.

2. Restore the **/sys/registry** tree from the backup media. (Alternatively, if the disk that contained the master replica on the original site is still intact, you can move the disk to the new node; in this case, you must run the **chuvol** utility on the volume you are moving.)

3. Restart the server by invoking **rgyd** with the **–restore_master** option:

   ```
   /etc/server -p /etc/rgyd -restore_master &
   ```

4. Verify that **rgyd** will start automatically at system startup using the UNIX **touch** command or the Aegis **crf** command to create a file named **rgyd** in **/etc/daemons**:

   ```
   touch /etc/daemons/rgyd
   ```

crf /etc/daemons/rgyd

5. Use the **rgy_admin delrep -force** command to delete the old master site from the replica list, as described in Section 6.3.

6.3 Forcibly Deleting a Registry Replica

This procedure to delete a registry replica uses the drastic **rgy_admin delrep -force** command. Use this method only when the ordinary method of deletion described in Chapter 5 has failed.

This command deletes the replica from the replica list at the master registry. The master then propagates the delete request to the other replicas. Since this operation never communicates with the deleted replica, you should use the **-force** option only when the replica has died irrecoverably. If a forcibly deleted replica later resumes operation, use the **rgy_admin reset** command to stop the server and delete its database.

Follow the steps listed below to forcibly delete a registry replica. Note that you must be an owner of the registry to perform Step 3 of this procedure.

1. Log in to any machine and invoke **rgy_admin**.

```
$ /etc/rgy_admin
Default object:  rgy  default host : dds://music
State:  in service  slave
rgy_admin:
```

2. Ensure that the **rgy_admin** default host is the current master registry site. (The name of the default host is displayed by **rgy_admin** when it is invoked.) If the master registry site is not the default host, enter:

```
rgy_admin: set -m
```

3. Use the **delrep** command with the **-force** option to delete the replica. The following example shows the replicas deleted from **dds://lit**:

```
rgy_admin: delrep dds://lit -force
```

6.4 Adopting File or Registry Orphans

Orphans are files or registry objects that are owned by UIDs that have been deleted from the registry. UIDs are deleted when the name associated with the UID is deleted. (Remember that Domain/OS identifies items by UID. For user convenience, however, Domain/OS displays names, not UIDs. When you delete names, either file names or the name of a registry object, you delete the UID associated with the name.)

Orphaned files can be accessed only according to the access rights specified by their now deleted owner. If the owner set up restrictive access rights, only root may have access to the file. Orphaned registry objects can be accessed only according to the rules of registry ownership. If the object is, for example, a domain, the results can be far reaching.

The following procedure describes how to reassociate a name and UID and thus adopt the orphan.

1. Invoke the **edrgy** command:

   ```
   $ /etc/edrgy
   edrgy=>
   ```

2. Use the **adopt** option in the following format to create a name and associate it with the orphaned UID:

   edrgy=> **adopt** *uid_high.uid_low name number*

   where:

uid_high.uid_low	Is the orphaned UID number, fully specified as 8 digits, a period, and 8 digits. You can obtain this UID number from any **edrgy** command that displays owners (such as **edrgy** with the v (view) option)
name	Can be a person, group, or organization name.
number	Is the new UNIX number to be associated with the name.

   The **adopt** option functions in a similar manner to the **add** option. With **adopt**, however, you supply the UID (the orphaned UID); with **add**, the system supplies it.

Chapter 7

Setting Up the Registry

This section describes how to set up the registry after Domain/OS has been successfully installed. Note that several of the procedures in this section must be performed by the **root** user.

Because the registry uses the NCS Location Broker to obtain information about network resources, this section assumes that your network is configured properly for Location Broker operation. Consult *Managing NCS Software* for considerations concerning Location Brokers in the network or internet. Consult *Managing Domain/OS and Domain Routing in an Internet* for information about using the registry in an internet.

7.1 Steps to Set Up the Registry

The following list summarizes the steps to set up the registry. The pages that follow describe the steps in more detail.

1. Plan the network configuration, considering which machines should run registry servers (**rgyd**), and which machine should run the master registry.
2. Ensure that the required Location Brokers are in place for the newly planned registry sites.
3. Create the registry database.
4. Start the master registry server.

5. Configure the registry database:

   — Set policies, properties, and passwords

   — Add names and accounts

6. Create slave registry replicas.

7. Set up **cron** to run **passwd_refresh** on nodes not running Domain/OS to ensure that the local password and group files are kept consistent with the registry. (The registry automatically maintains consistency with the local registry for Domain/OS machines.) The **passwd_refresh** command is described in Chapter 4.

8. Ensure that the **/etc/rgy/pname_resolve** file exists on each node that does not run Domain/OS. This file will supply a Domain mount point substitution string for nodes that do not run Domain/OS.

This chapter also describes how to restart a registry server.

7.2 Planning a Configuration

To plan a configuration, choose a site for the master registry server and, if you decide to run several servers, choose sites for them. The master and slave registry servers (**rgyd**) can reside only on machines that run Domain/OS. Client software can run on machines that run Domain/OS; SunOS 3.4, 3.5, or 4.0; or VAX ULTRIX 2.2.

7.2.1 Registry Server Requirements

Use the following considerations to plan the configuration of registry servers for your site:

- The hardware requirements for the nodes that will run registry servers are proportionate to the size of the registry database and the load imposed on the server. Use the following considerations as general guidelines:

  -- A node running a registry server should have free disk space equal to either 8 MB or 5 times the size of the registry database, whichever is greater.

  -- A node running a registry server should have at least 4 MB of memory. If the virtual memory size of the registry process is greater than 4 MB, the node's memory should be as close to the process virtual memory size as possible. The closer the memory size is to the process size, the more efficiently the registry process will run. (Use the **ps -v** command to determine process virtual memory size.)

  Take care to choose registry server nodes large enough to accommodate future growth of the registry.

- You can run more than one registry daemon (**rgyd**) on a network. In an internet, we recommend that you run at least one **rgyd** in every network.
- Nodes running **rgyd** should be up and available at all times. It is especially important that the node where the master server runs be available throughout the network or internet.

7.3 Ensuring Proper Location Broker Configurations

Machines in the network use Network Computing System (NCS) Location Broker servers to locate registry servers. In general terms, the Location Broker maps services and resources to network addresses. In this case, the Location Broker provides information about the network locations of registries.

To do this, the Location Broker uses two servers: a **glbd** (the replicatable Global Location Broker daemon) that provides information about the resources throughout the entire network or internet and a **llbd** (Local Location Broker daemon) that provides information about the local machine's resources.

To run the registry server or any NCS–based server:

- At least one Global Location Broker (**glbd**) must run on the network.
- A Local Location Broker (**llbd**) must run on every node that runs a **glbd** and on every node that will run a registry server (master or slave).

NOTE: In rare instances, a **nrglbd** (non–replicatable global location broker) may run in place of a **glbd**. We strongly recommend that you consult *Managing NCS Software* for detailed information on configuring, starting, and managing Location Brokers.

7.4 Creating a New Registry Database

If you are setting up a new network or internet that does not contain an existing registry, use the **/install/tools/rgy_create** utility to create a new database. When **rgy_create** creates a new database, it also initializes it with reserved names and accounts. Typically the new registry is created as part of Domain/OS installation. See *Installing Software with Apollo's Release and Installation Tools* for more information about **rgy_create.**

Because **rgy_create** is setuid as root, you may want to implement tighter protections on **rgy_create** after you create the new regitstry database.

7.5 Starting the Master Registry Server

Before you start the master registry, ensure that a **glbd** is running on the network. (You can use the **lb_admin** command described in *Managing NCS Software* to check if an **lb_admin** is running.)

1. Log in as **root** on the master registry site machine (which must be a machine running Domain/OS Software Release 10.2 or later).

2. Use **/bin/ps** or **/com/pst** to verify that an **llbd** is running on the node. If one is not, start one. To do so enter:

   ```
   /etc/server  /etc/ncs/llbd
   ```

   Refer to *Managing NCS Software* for detailed information on the configuration requirements and starting Location Broker daemons at the command line and automatically at system startup.

3. Start the registry server by entering:

   ```
   /etc/server -p /etc/rgyd &
   ```

   The **-p** option causes **/etc/server** to create a process that runs under the account that invoked the command, rather than the default of **user.server.none.**

4. To start the server automatically each time the server node reboots, create a file named **rgyd** in the **/etc/daemons** directory using one of the following commands

   ```
   touch /etc/daemons/rgyd          (UNIX environments)

   crf /etc/daemons/rgyd            (Aegis)
   ```

5. Use the **rgy_admin lrep -state** command to verify that the registry server is running.

7.6 Configuring the New Registry Database

If the registry database is newly created, you must configure the database by setting policies and procedures and adding accounts.

7.6.1 Setting Policies, Properties, and Passwords

Use the **edrgy prop** command to view policies and properties and change them as desired. As you examine the database, notice that it contains default passwords for all reserved accounts except **user.none.none**, which by default has no password. Use **edrgy**, as described in Chapter 3, to change or set the passwords for these accounts.

Now is also the best time to change the owners of reserved registry objects, if you do not want to use the defaults.

7.6.2 Adding Accounts

After a new registry database is created, it contains only the names added as reserved information by **rgy_create**. Use **edrgy** to add any other names and accounts that your site requires. You can do this now or at any time later. See Chapter 2 for how to add accounts.

7.7 Creating Slave Registry Replicas

To create slave replicas follow the steps listed below.

1. Log in as root on a machine running Domain/OS Software Release 10.2 or later that will run the slave.

2. Use **/bin/ps** or **/com/pst** to verify that an **llbd** is running on the node. If one is not, start one as described in Section 7.5.

3. Enter the following command to start the slave server:

   ```
   /etc/server -p /etc/rgyd -create &
   ```

   This command locates the master server, adds the local node to the master replica list, and causes the master to initialize a new database at the local node. It is used only to create a new slave replica. If you must restart an existing replica, following the instructions in Section 7.11.

4. To start the server automatically each time the server node reboots, create a file named **rgyd** in the **/etc/daemons** directory using one of the following commands

touch /etc/daemons/rgyd	(UNIX environments)
crf /etc/daemons/rgyd	(Aegis)

5. Repeat Steps 1 through 4 for each replica you want to create.

6. Use the **rgy_admin lrep –state** command to verify that the registry servers are running.

7.8 Verifying the Servers are Running

After the master and slave registries are in place and started, perform the following steps to ensure they are running:

1. Invoke **rgy_admin**:

   ```
   $ /etc/rgy_admin
   Default object:  rgy  default host : dds://music
   State:  in service  slave
   rgy_admin:
   ```

2. Issue the **lrep** command with the **state** option to display all registry servers and their status:

   ```
   rgy_admin:  lrep -state
      (master) dds://music     state:in service  1990/11/16.12:46:59
               dds://art       state:in service  1990/11/16.12:46:59
               dds://lit       state:in service  1990/11/16.12:46:59
   ```

7.9 Establishing Uniform UNIX IDs

If you share files with other systems that do not use the registry, you should ensure that names, UNIX IDs, and account information are consistent between the registry and the foreign **passwd** and **group** files. (For the purposes of this discussion, "file sharing" can take the form of direct access through facilities such as NFS or indirect file transfer via media such as **tar** tapes.) For example, assume your network contains a Pyramid workstation. Although the Pyramid cannot participate in the registry it will share files and resources with the registry–compatible systems.

We provide a tool called **import_passwd** that helps you to identify and resolve conflicts of names, UNIX IDs and account information. If you plan to share files between systems controlled by the registry and systems that are not, we recommend you run **import_passwd** now to minimize the number of changes you have to make. Typically, you run **import_passwd** in a mode that changes IDs in the Domain/OS registry to match the IDs on the foreign systems; afterward, you run another tool called **syncids** to ensure that the IDs stored in the Domain file systems match those stored in the registry. See Appendix B for details on how to run **import_passwd** and **syncids**.

7.10 Establishing the Domain Mount Point Pathname

The Domain/OS file system uses the string // to indicate the network-wide file system root. Other network file systems (notably NFS) do not accept this syntax. On machines using NFS, you must create a file named **/etc/rgy/pname_resolve** and edit it to contain the pathname on which the Domain/OS file system is mounted. For example, if the Domain/OS file system is mounted on **/apollo**, the file would contain the string **/apollo**.

7.11 Restarting Registry Servers

To restart a registry master or slave server, enter:

```
/etc/server -p /etc/rgyd &
```

The **-p** option causes **/etc/server** to create a process that runs under the account that invoked the command, rather than the default of **user.server.none**.

To start the server automatically each time the server node reboots, create a file named **rgyd** in the **/etc/daemons** directory using one of the following commands

touch /etc/daemons/rgyd	(UNIX environments)
crf /etc/daemons/rgyd	(Aegis)

Note that the server must have been created either with the **rgy_create** command if it is a master server or the instructions in Section 7.7 if it is a slave server.

Appendix A

The import_passwd Command

The **import_passwd** command creates entries in the registry based on information in UNIX password and group files. It provides a method of ensuring account consistency between machines controlled by the registry and UNIX machines not supported by the registry. **import_passwd** automatically creates registry entries based on UNIX password and group entries.

A.1 How import_passwd Works

When **import_passwd** processes entries, it compares group and password file entries to registry entries. It can find two types of conflicts:

- **Name Conflicts** — These conflicts arise when the same name string is defined in the registry and the group and password files. Joe 102 and Joe 555 exemplify such a conflict. The duplicate name may represent the same user or two different users.

- **UNIX ID Conflicts** — These conflicts arise when the same UNIX ID is defined in the registry and the group and password files for users with different names. Joe 102 and Ann 102 exemplify such a conflict.

These conflicts can be found separately, as in the examples above, or together. For example, a registry entry of Joe 102 and a foreign entry of Joe 102 are in conflict. Unless they represent the same user, one of the entries must be changed.

A.1.1 Processing Steps

As **import_passwd** processes entries, it performs the following steps in sequence:

1. It puts the registry in maintenance mode and reads the group and password files.
2. It compares the group file entries to the registry group entries. If there are no conflicts, it creates registry group entries corresponding to the groups. (Section A.1.4 describes what happens if there are conflicts.) Note that the members of the groups are not added at this time, but in Step 4.
3. It compares the entries in the password file to the registry person and account entries. Again, if there are no conflicts, it creates registry person and account entries corresponding to the foreign file.
4. If there are members in the foreign groups handled in Step 2, it adds them to the appropriate group in the registry.
5. It then prompts for whether or not to make the changes. If you specify that the changes should be made, it saves the changes to the registry.

A.1.2 Registry Entries Modified by import_passwd

The **import_passwd** command modifies only person names, person IDs, group names, group IDs, group members, and account passwords. It does not modify any of the additional information in the registry.

For example, assume you have a password entry for user **jack** and group **staff** and a registry account entry of **jack.staff.none**. You run **import_passwd** with the **-i** option. This option tells **import_passwd** to consider the entries identical. The home directory specified in the foreign network is **/usr/u/jack**; the home directory specified in the Domain network is **//gimli/jack**. **import_passwd** will not change the registry to match the foreign home directory. **jack.staff.none** in the registry will have a home directory of **//gimli/jack**, not **/usr/u/jack**.

If **jack.staff.none** did not exist in the registry, **import_passwd** would create a new registry entry.

A.1.3 Registry Entries Created by import_passwd

For the additional registry information, **import_passwd** takes the following entries:

- **For Person and Group Entries:**

  fullname = " (that is, empty)

owner = same as the owner of the domain as listed in the registry properties (that is, the owner for new person entries is set to Person Owner and the owner for new group entries is set to Group Owner.)

alias/primary = primary for first entry; alias for subsequent ones

projlist_ok (for groups only) = yes

passwd = for groups only, taken from the group file

membership list = for new groups only, all persons listed in the group file and all persons with accounts in the password file with that group

- **For Account Entries:**

  abbreviation = shortest possible abbreviation that does not conflict with pre-existing registry accounts

  account_valid = true

  gecos = same as password file

  homedir = same as password file

  shell = same as password file

  passwd = same as password file. Note that you must modify or reset imported passwords before user authentication is possible. You must also change them to be able to log in to a machine running a release of Domain/OS that is earlier than Software Release 10.

  passwd_dtm = date and time **import_passwd** was run

  passwd_valid = true

A.1.4 Resolving Conflicts

When you use **import_passwd**, you must decide how to resolve the conflicts it will encounter. The **import_passwd** command provides a number of options to help you. If the conflict cannot be resolved even with the option instructions, **import_passwd** will prompt you for resolution. The options are described in the following paragraphs.

The Identical User Option

The **-i** option lets you specify that duplicate names are not in conflict, but instead, represent the same identity. When **import_passwd** finds duplicate name entries, it processes them as though they are the same user. If you do not use the **-i** option, **import_passwd** will prompt you to resolve the name conflict.

The Favored Entry Option

The **–a** (favor registry) and **–f** (favor password and group) options let you specify whether the registry entry or the password and group entry is the favored entry. A favored entry is retained as it exists. You are prompted to modify non–favored entries.

For example, suppose you run **import_passwd** with the **–a** (favor registry) option and without the **–i** (identical user) option. During processing, the program encounters a registry entry of **joe 555** and a foreign entry of **joe 102**. Because the registry entry is favored, **joe 555** will be retained in the registry, and you will be prompted for a new name for **joe 102**.

import_passwd also uses the favored entry to resolve UNIX ID conflicts. For example, suppose you are running **import_passwd** with the options described above (**–a** and **–i**). During processing, **import_passwd** encounters a registry entry of **joe 555** and a foreign entry of **ann 555**. Because the registry entry is favored, **import_passwd** prompts for a new UNIX ID for **ann**.

Be aware, however, that reserved registry entries cannot be modified. (Reserved entries are listed in Table 2–3.) **import_passwd** will not modify a reserved entry even if it is the nonfavored entry. For example, suppose you are running **import_passwd** with the foreign entry as the favored entry. During processing, **import_passwd** encounters a foreign group entry of **misc 0** and a registry group entry of **wheel 0**. Because group **wheel 0** is a reserved registry entry, you will be prompted to modify the foreign entry, even though it is the favored entry.

Conflict Summary

Table A–1 summarizes the effects of the identical user and favored entry options.

Table A-1. Effects of **import_passwd** *Options on Conflict Resolution*

Options Used	Foreign Entry	Registry Entry	Result in Registry	Comments
–i, –a	joe 102	joe 555	joe 555	Name collision. Retain registry entry.
	joe 102	ann 102	ann 102	UNIX ID conflict. Request new UNIX ID for joe.
–i, –f	joe 102	joe 555	joe 102	Name collision. If 102 already exists in the registry, prompt for new UNIX ID for that entry.
	joe 102	ann 102	joe 102	UNIX ID conflict. Request new UNIX ID for ann.
–a	joe 102	joe 555	joe 555	Name conflict. Request new name for joe 102, and if 102 is already defined in the registry, a new UNIX ID as well.
	joe 102	ann 102	ann 102	UNIX ID conflict. Request new UNIX ID for joe.
–f	joe 102	joe 555	joe 102	Name conflict. Request new name for joe 555, and if 102 is already defined in the registry, prompt for a new UNIX ID for that registry entry.
	joe 102	ann 102	joe 102	UNIX ID conflict. Request new UNIX ID for ann.

A.2 import_passwd Syntax

import_passwd [**–i**] [**–a** | **–f**] [**–c**] [**–o** *org*] **–s** *pathname* [**–v**]

–i	Identical name strings are not in conflict, but represent the same identity.
–a	Favor registry entries. **–a** is the default.
–f	Favor foreign entries.
–c	Run in check mode: process the command showing conflicts, but make no changes to the files.
–o *org*	*org* is the name of the registry organization to be used for all imported account entries. The default is the organization named "none."
–s *pathname*	*pathname* is the path to the directory containing the foreign password and group files to be imported.
–v	Run in verbose mode: generate a verbose transcript of all activity.

A.3 Using import_passwd

This section describes how to use the **import_passwd** command.

A.3.1 Using Check Mode

You should first run **import_passwd** in check mode using the **-c** option. In this mode, **import_passwd** simulates the results of a processing run showing the conflicts that will be encountered when **import_passwd** is run without the **-c** option.

Check mode gives you a good idea of the quantity and complexity of the potential conflicts. However, check mode does not make any changes to the file. When you run without the **-c** option and make changes to resolve conflicts, these changes can in turn create further conflicts not readily apparent in check mode.

If you encounter numerous conflicts in check mode, you may find it more efficient to manually edit either the registry or the UNIX group and password files to resolve some obvious conflicts before you run **import_passwd**.

A.3.2 Answering Prompts

When you run **import_passwd**, you may be prompted for names and numbers (UNIX IDs). The following conventions apply to your answers to these prompts:

- Names must
  - Begin with a lowercase letter
  - Contain *only* lowercase letters, digits, or underscore characters
  - Not exceed 32 characters in length, unless your system imposes UNIX restrictions, in which case, they cannot exceed 8 characters in length

  If your system imposes UNIX name restrictions, you should carefully evaluate the effect of these restrictions on existing entries in the files against which you are running **import_passwd**. Some of the entries may be longer that 8 characters.

- Numbers must range between 0 and 65535, inclusive. (We suggest that you not use numbers under 100, since these may be reserved in future releases.)

If you enter a name or number in an incorrect format, **import_passwd** will ignore your entry and prompt you again.

A.3.3 Processing Prerequisites

The registry must exist before you can use **import_passwd**. If you are simply adding a few registry–controlled nodes to a foreign network, you can create a new, but empty registry to meet this requirement. Once the registry exists, the registry server must be running, and you must be logged in as root.

A.3.4 Synchronizing Domain UNIX IDs

If **import_passwd** makes any changes to registry UNIX IDs, you must run the **/etc/syncids** tool on every registry–controlled node in your network. This tool ensures that the changes are synchronized with the UNIX IDs stored on disk. On nodes that have more than one disk volume, you must run **syncids** on each volume.

If the registry is a slave replica, before you run **syncids**, you should use the **lrep –state** command in the **/etc/rgy_admin** tool to verify that all slave replicas are up–to–date. This command shows the timestamps of the most recent changes to each replica. You should not run **syncids** until the timestamps for all replicas match the timestamp for the master.

See the online manual pages for information about syncids.

A.3.5 Synchronizing Foreign UNIX IDs

If any of the conflicts found by **import_passwd** involve reserved registry information, you may have to resolve conflicts by changing names and IDs on the foreign systems. You may also have to change names and IDs on foreign systems if you run **import_passwd** with the **–a** (favor registry) option. In that case, you must then use the **chown** and **chgrp** utility to update the UNIX IDs on all affected files and directories in the foreign file systems.

A.4 Sample import_passwd Session

This section shows a simplified **import_passwd** session. The session consists of five phases:

- Invoking **import_passwd**
- Examining the group entries
- Examining the password entries
- Adding members to groups
- Updating the registry

The sample session uses the registry group file and password entries shown in Table A–2 and the foreign group file and password file entries shown in Table A–3.

Table A–2. Registry Group and Password Entries

Group Entries
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:
Password Entries
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::/:

Table A-3. Foreign Group and Password File Entries

Foreign Group File Entries
root::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:
Foreign Password File Entries
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: nuucp::10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/uucico 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

A.4.1 Invoking import_passwd

In the sample session, the following form of the **import_passwd** command is entered at the shell prompt:

```
$ import_passwd -s sys5.3_tapes/adm -i -v
```

This command specifies that:

- Identical names represent the same identify (**-i**)
- **sys5.3_tapes/adm** is the pathname to the foreign file
- Registry entries should be favored (default of **-a**)
- The command will not run in check mode (**-c** not specified)
- The command will run in verbose mode (**-v**)

The system puts the registry in maintenance mode and reads the foreign group and password files in preparation for creating group entries in the registry. As it performs these actions, it displays:

```
Preparing registry...
Preparing foreign files...
Creating Group entries from foreign group file...
```

A.4.2 Examining the Group Entries

As **import_passwd** reads the foreign group and password files, it informs you of any conflicts and prompts for their resolution. The steps below show how group conflicts are handled.

1. **import_passwd** first finds a conflict on UNIX IDs: as shown in Table A-2 and Table A-3, the name **root** in the foreign files and the name **wheel** in the registry both have UNIX IDs of 0. Since registry entries are favored, **import_passwd** prompts for a new UNIX ID for the foreign entry:

```
root::0:root
?(import_passwd) Group - UNIX id conflict
Foreign Group: "root" 0  Apollo: "wheel" 0 (reserved).
Please enter new UNIX id for Foreign Group 'root' 0: 100<Return>
>> Adding pgo entry for: root 100
```

In the example, **100** is entered as the new UNIX ID for the foreign group entry. Note that **import_passwd** displays the foreign entry as it examines it (in the sample above, **root: :0:root**). In addition, because it is running in verbose mode, **import_passwd** describes the actions it is taking. Each such description is prefaced with the symbols >>.

If you were running **import_passwd** in check mode, you would not be prompted for a UNIX ID. Instead, you would be informed of the need and processing would continue. The message would look like this:

```
root::0:root
?(import_passwd) Group - UNIX id conflict
Foreign Group: "root" 0  Apollo: "wheel" 0 (reserved).
    need new UNIX id for Foreign Group
      (UNIX id for Apollo "wheel" is currently 0)
```

2. **import_passwd** then finds another UNIX ID conflict:

```
other::1:
?(import_passwd) Group - UNIX id conflict
Foreign Group: "other" 1  Apollo: "daemon" 1 (reserved).
Please enter new UNIX id for Foreign Group 'other'1: 101<Return>
>> Adding pgo entry for: other 101
```

Note that **import_passwd** tells you that the UNIX ID of **1** is reserved. Because **1** is reserved, even if you had run **import_passwd** with the foreign entry favored, you would still be prompted for a new entry for the foreign UNIX ID. **import_passwd** will not change reserved entries.

3. If **import _passwd** finds no conflicts, it displays the group entries as it processes them and, because it is running in verbose mode, the actions it is taking:

```
bin::2:root,bin,daemon
>> Foreign Group: "bin" 2  Apollo: "bin" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id

sys::3:root,bin,sys,adm
>> Foreign Group: "sys" 3  Apollo: "sys" 9 (reserved)
>>   -i specified - retaining Apollo UNIX id
```

As **import_passwd** continues through the foreign group file, it finds two other UNIX ID conflicts: foreign entries **adm 4** and **rje 8**, which are in conflict with registry entries **locksmith 4** and **server 8**, respectively.

A.4.3 Examining the Password File

import_passwd then proceeds to create registry person and account entries from the foreign password file. It displays:

```
Creating Person entries and Accounts from foreign passwd file...
```

The following steps show how person and account conflicts are handled.

1. The first two entries are processed with no conflicts:

```
root::0:1:0000-Admin(0000):/
>> Foreign Person: "root" 0  Apollo: "root" 0 (reserved)
>>   -i specified and UNIX ids match - no change necessary
>> Adding account for root.other.none.

daemon::1:1:0000-Admin(0000):/
>> Foreign Person: "daemon" 1  Apollo: "daemon" 1 (reserved)
>>   -i specified and UNIX ids match - no change necessary
>> Adding account for daemon.other.none.
```

import_passwd displays the names of the registry accounts it is creating. If you had changed a person or group name to resolve a previous name conflict, the new name would be displayed. For example, suppose that you had a foreign and a registry entry of **joe.other.none**. During **import_passwd** processing, you changed the name of the registry **joe** to **smith**. When **import_passwd** created the registry person and account entries, a verbose display would look like:

```
joe::1:1:0000-Admin(0000):/
>> Adding account for smith.other.none.
```

The foreign entry is displayed first with its correct name, **joe**. The account being added to the registry is displayed by its new name, **smith**. **import_passwd** keeps track of changes that are made and the relationships that existed before the changes were made.

2. The third entry produces the following warning message:

```
bin::2:2:0000-Admin(0000):/bin
>> Foreign Person: "bin" 2  Apollo: "bin" 8 (reserved)
>>   -i specified - retaining Apollo UNIX id
>> Adding account for bin.bin.none .
(WARNING) "bin.bin.none": Account already exists and is retained
unchanged.
```

This message notifies you that the **bin.bin.none** account exists in the registry. The registry account will retain its information (that is, *abbreviation*, *account_valid*, *gecos*, *passwd*, *shell*, *home directory*, *passwd_dtm*, and *passwd_valid*), even though the foreign account may have information that differs from the registry account.

3. Processing continues. **import_passwd** discovers two more UNIX ID conflicts (foreign **sys 3** and registry **user 3**; foreign **adm 4** and registry **lp 4**). Then **import_passwd** finds an un-named group in the foreign password file and a UNIX ID conflict.

4. First **import_passwd** prompts for a group name:

```
uucp::5:5:0000-uucp(0000):/usr/lib/uucp
>> Foreign Person: "uucp" 5  Apollo: "uucp" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id
?(import_passwd) Foreign Group 5 is unnamed.
    (UNIX id also conflicts with Apollo "login" (reserved) ).
Please enter a name for Foreign Group 5: group_105 <Return>
```

In the sample session, **group_105** is entered as the new name.

Note that if you were running in check mode, **import_passwd** would supply a name for the group in order to keep processing. The prompt would look like:

```
uucp::5:5:0000-uucp(0000):/usr/lib/uucp
>> Foreign Person: "uucp" 5  Apollo: "uucp" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id
?(import_passwd) Foreign Group 5 is unnamed.
    (UNIX id also conflicts with Apollo "login" (reserved) ).
    Temporarily using the name of "group_5"
```

5. Then, **import_passwd** prompts for a new UNIX ID for the group:

```
?(import_passwd) Group - UNIX id conflict
Foreign Group: "g105" 5  Apollo: "login" 5 (reserved).
Please enter new UNIX id for Foreign Group 'g105' 5: 105 <Return>
>> Adding pgo entry for: g105 105
>> Adding account for uucp.g105.none.
```

A.4.4 Adding Members to Groups

When **import_passwd** completes processing of the foreign password file, it adds the members to the groups and displays information like the following:

```
Adding memberships from foreign group file...
root::0:root
  Member: root
other::1:
bin::2:root,bin,daemon
  Member: root
  Member: bin
  Member: daemon
```

If there are no members added (as in group **other::1:**), none are displayed. If you are not running in verbose mode, only the names of the groups are displayed as they are processed, not the members being added.

Handling Members not Yet Added as Persons

When **import_passwd** processes the **rje::8:rje,shger** entry, it displays the following message:

```
rje::8:rje,shqer
  Member: rje
  Member: shqer
?(import_passwd) Cannot add member - Person is unknown or invalid
```

In foreign UNIX systems, you can add members to groups even if the person has not been added as a person. In the registry you cannot. **import_passwd** has found a member of a foreign group who has not been added as a person. Therefore, **import_passwd** will not add that member to the group.

A.4.5 Updating the Registry

Processing continues until **import_passwd** finishes adding members to groups. It then completes processing following the steps shown below.

1. **import_passwd** recaps the results of its processing by displaying a message similar to the one below:

```
0 name conflicts, 8 UNIX id conflicts, 3 unnamed groups,1 error,
2 warnings.
```

2. **import_passwd** asks if you want to save the changes by displaying:

```
Import operation successful.  Update registry [y/n]?  y <Return>
```

   In this sample session, the answer is y (yes).

3. **import_passwd** then saves the changes and completes processing:

```
Closing registry...
Closing foreign files...
Done.
```

Sample Name Conflict

Because the **-i** option was specified in the command used in the sample session, no name conflicts were encountered. A sample of the prompt resulting from a name conflict is shown below:

```
bin::2:root,bin,daemon
?(import_passwd) Group - Name conflict
Foreign Group: "bin" 2  Apollo: "bin" 7 (reserved)
Please enter new name for Foreign Group "bin": fbin <Return>
>> Adding pgo entry for: fbin 2
```

In the sample above, a user assigns **fbin** to the entry in the Domain registry that represents the foreign group **bin**.

Appendix B

edrgy Quick Reference

This appendix contains:

- Tables that list the information you enter to use the **edrgy** command
- Examples of how to invoke **edrgy**
- Examples of how to use **edrgy** to create, change, delete, and view persons, groups, organizations, and accounts and to set policies and properties

Note that all examples in this section assume that the **edrgy** command has been invoked.

Use this appendix as a quick reference to registry editing. See Chapters 2 and 3 for detailed descriptions of the use of **edrgy**.

B.1 Information Used by the edrgy Command

The following tables list the information prompted for by the **edrgy** command. Table B–1 lists the information the **edrgy** command requires to create and maintain persons; Table B–2 lists the information required for groups and organizations; and Table B–3 lists the information required for accounts.

Table B-1. Information Used by **edrgy** *to Create and Maintain Persons*

Information	Description	Entry Format
Full Name	An optional name used to more fully describe a primary name.	A series of characters enclosed in quotes. Spaces are allowed.
Owner	The person's owner.	p.g.o
Primary Name	The required standard name associated with the person.	A series of up to 8 characters (if your system imposes UNIX restrictions) or up to 32 characters (if your system does not impose UNIX restrictions). Name can consist of only lowercase letters, digits, and underscore characters.
UNIX Number	The required UNIX ID associated with the person. (This ID is used for compatibility with existing UNIX programs.)	A number between 0 and 65535, inclusive. Avoid using numbers lower than 100 because they may be reserved in future releases.

Table B-2. Information Used by **edrgy** *to Create and Maintain Groups and Organizations*

Information	Description	Entry Format
Full Name	An optional name used to more fully describe a primary name.	A series of characters enclosed in quotes. Spaces are allowed.
Owner	The group or organization owner.	p.g.o
Password	*In SysV environments only*, a password that allows users who are not members of a group to acquire the group's privileges.	A series of characters. Depending on policies set for your company, passwords may have a minimum length and be restricted from containing all spaces and from consisting of all alpha characters.
Primary Name	The required standard name associated with the group or organization.	A series of up to 8 characters (if your system imposes UNIX restrictions) or up to 32 characters (if your system does not impose UNIX restrictions). Name can consist of only lowercase letters, digits, and underscore characters.
PROJLIST	**For groups only**, whether the group can be on project lists.	y or n
UNIX Number	The required UNIX ID associated with the group or organization (used for UNIX compatibility).	A number between 0 and 65535, inclusive. Numbers below 100 may be reserved in future releases.

Table B-3. Information Used by the **edrgy** *Command to Create and Maintain Accounts*

Information	**Example**	**How the Information is Used**
Subject Identifier	grappelli.violinists.jazz	To identify the account.
Abbreviation Type	p, pg, pgo	At login in place of the complete subject identifier. If you enter an abbreviation type of: • **p** = The user need enter only the person name to log in • **pg** = The user must enter the person and group name to log in • **pgo** = The user must enter the entire subject identifier to log in If the abbreviation type causes an abbreviation that is not unique, **edrgy** ignores the entry and assigns the shortest unique abbreviation possible. For example, if **mozart.composer.classic** exists with the abbreviation **mozart**, the shortest abbreviation **mozart.composer.none** can have is **mozart.composer**. Note: **Sendmail** requires the use of the **p** (person-only) abbreviation. Use it for all accounts that will use sendmail.
An encrypted password	sq1Rc1Urrb1L6	To help ensure log-in security. When users log in, they are prompted for a password. The system encrypts the password text string and checks the result against the encrypted string stored in the database. When you add an account, you supply a plain text password; the system performs the encryption.
A home directory	//walden/grappelli	When users log in, they are moved to their home directory.
A login shell	/bin/csh	To determine the shell to be executed when a user logs in.
Miscellaneous	login account for grappelli	A text string typically used to describe the use of the account.
An account-valid flag	Yes or No	To determine account validity. Invalid accounts cannot log in.
A password-valid flag	Yes or No	To determine whether the current password is valid. If this flag is set to No, the next time a user logs into the account, the system prompts change the password. (Note that this flag is separate from the password expiration policy, which sets time limits on password validity.)

Table B-4. Standard Policy Information Used by the **edrgy** *Command*

Policy	**Purpose**	**Effect of the Policy**
Password Expiration Date	A date that determines when passwords expire. An entry of **none** indicates there is no expiration date in effect.	Can be set for an individual account or all accounts in an organization. If an account's password has expired, the next time the user logs in, **login** will invoke the **/bin/passwd** command, which requests that the user change the password. If the user does not change the password, login is denied.
Password Life Span	A number that determines the number of days before passwords expire. An entry of **forever** indicates there is no limit to the life span.	Can be set for an individual account or all accounts in an organization. If an account's password has expired, the next time the user logs in, **login** will invoke **/bin/passwd**, which requests that the user change the password. If the user does not change the password, login is denied.
Password Format	Determines the following characteristics of account passwords: • Minimum length (a zero indicates no minimum length) • Whether passwords can consist entirely of spaces • Whether passwords can consist entirely of alphanumeric characters	Can be set for an individual account or for all accounts. Account passwords must be constructed to follow the format.
Account Life Span	A number that determines the number of days before accounts expire. An entry of **forever** indicates there is no limit to the life span.	Can be set for an individual account or all accounts in an organization. If an account has expired, it is invalid, and must be re-created.

Table B-5. Override Policy Information Used by the **edrgy** *Command*

Policy	**Purpose**
Password Exclusion	Determines whether nonvalid strings can be entered as a password override and thus prevent users from logging in to specific machines.
Root Password Overrides	Determines whether the root password can be overridden and thus allow individual machines to require a root password different from the one stored in the registry.
Nonroot Password Overrides	Determines whether any user's password can be overridden and thus allow individual machines to require an account password different from the one stored in the registry.
Home Directory Overrides	Determines whether an account's home directory can be overridden for a specific machine and thus allow the account to have a home directory different from the one named in the registry.
Login Shell Overrides	Determines whether an account's login shell can be overridden for a specific machine and thus allow the account to have a login shell different from the one named in the registry.
GECOS Overrides	Determines whether an account's GECOS information can be overridden for a specific machine and thus allow the account to have GECOS information different from the information defined in the registry.

Table B-6. Property Information Used by the **edrgy** *Command*

Property	**Purpose**
Registry Owner	Identifies the owner of the registry as a whole.
Domain Owners	Identifies owners of the person, group, and organization domains.
UNIX Restrictions	Determines whether UNIX restrictions will be enforced. If they are: • Names of registry objects can not exceed eight characters. If you do not apply this property, names can be up to 32 characters in length. (UNIX name length restrictions do not apply to reserved names.) • Accounts must have the p (person only) abbreviation. • All password entries must use standard UNIX encryption.
Read-Only Property	Determines whether the database is read-only.

B.2 edrgy Command Examples

The following pages provide examples of how to invoke, exit, and use the **edrgy** command. The examples are organized alphabetically by task. For example, if you want to see an example of deleting an account, look under "deleting." Each page in this section has a header that names the first example on the page. For example, if a page contains examples for creating aliases, deleting accounts, and deleting groups and organizations, the heading would be Creating Aliases. Use the headings to access the task examples as you would use key words in a dictionary.

B.2.1 Adding Accounts

Use the **edrgy add** command to create accounts. You must be in the account domain to add accounts. Before you create an account, the person, group, or organization that will make up the account must have been added to the database.

The following example shows the **edrgy add** command used to create the account **mahler.symphonists.classic**. The example assumes that you are in the account domain.

```
edrgy=> add
add account=> Enter account id [p.g.o]: mahler.symphonists.classic
Enter abbreviation type [p/pg/pgo]: (p) p
Enter new password: (-apollo-) sixth
Enter misc info in quotes: ()
Enter home directory: (/) //leitmotif/mahler
Enter shell: () /bin/csh
Password valid [y/n]? (y)                                 <Return> accepts defaults
Enter expiration date [yy/mm/dd or 'none']: (none)        for these prompts
Account valid for login [y/n]? (y)
add account=>                         <Return> exits from the add command
edrgy=>
```

B.2.2 Adding Aliases

Use the **edrgy add** command to create aliases. To create aliases, you must be in the person domain.

The following example creates the alias **gus** for the person **mahler**. In the example, the UNIX ID identifies the person for whom the alias is being added.

```
edrgy=> do person              Domain changed to person
edrgy=> add gus 1683 -al             The -al option adds the alias "gus" for
edrgy=>                              the person identified by UNIX ID 1683.
```

B.2.3 Adding Groups

Use the **edrgy add** command to add groups. You must be in the group domain to add groups.

The following example shows the **edrgy add** command used to add a group named **symphonists**. The example assumes you are in the group domain.

```
edrgy=> add
add group=> Enter name: symphonists
Enter UNIX number: 3332
Enter full name in quotes:
Enter new password:
Include group on PROJLIST [y/n]? (y) y
Enter owner [p.g.o]: (%.rgy_admin.%)
edrgy=>
```

Full name and password are skipped with <return>

<Return> allows ownership to default to %.rgy.admin.%

B.2.4 Adding Members to Groups and Organizations

Use the **edrgy member** command to add members to groups and organizations. To add members to groups, you must be in the group domain. To add members to organizations, you must be in the organization domain.

The following example shows the **edrgy** command used to add **mahler** to the group **symphonists.** The example assumes you are in the group domain.

```
edrgy=> member
member=> Enter group name: symphonists
member=> Enter name to add: mahler
member=> Enter name to remove:
member=> Enter group name:
edrgy=>
```

The **member** *command adds a member by first prompting for the group to add the member to and then for the member name*

edrgy *next prompts for the name of a member to delete. <Return> prompts for the name of another group to add a member to. A second <Return> exits from the member list edit.*

B.2.5 Adding Organizations

Use the **edrgy add** command to add organizations. You must be in the organization domain to add organizations. The following example adds the organization named **classic.** The example assumes you are in the organization domain.

```
edrgy=> add
add org=> Enter name: classic
Enter UNIX number: 3333
Enter full name in quotes: "classic composers"
Enter new password:
Enter owner [p.g.o]: (%.rgy_admin.%)
edrgy=>
```

Password is skipped with a <return>

<Return> allows ownership to default to %.rgy.admin.%

B.2.6 Adding Persons

Use the **edrgy add** command to add persons. You must be in the person domain to add persons.

The following example shows the **edrgy** command used to add the person identified by a primary name of **mahler**. The example assumes you are in the person domain.

```
edrgy=> add                                   Add function invoked
add person=> Enter name: mahler                                          Primary name added
Enter UNIX number: 1683                               UNIX ID added
Enter full name in quotes: "gustav mahler"                               Full name added
Enter owner [p.g.o]: (%.rgy_admin.%)              Ownership defaults to %.rgy_admin.%
add person=> Enter name:
edrgy=>                            A <return> displays the edrgy prompt
```

B.2.7 Changing Accounts

Use the **edrgy change** command to change any account information except the account SID. You must be in the account domain to change account information.

The following example shows the **edrgy change** command used to enter a new home directory (**//concert/mahler**) for the **mahler.symphonists.classic** account. The example assumes that you are in the account domain.

```
edrgy=> change
change account=> Enter account id [p.g.o]: mahler.symphonists.classic
Enter new abbreviation type [p/pg/pgo]:                        <Return> leaves the
Enter new password:                                             information unchanged
Enter new misc info in quotes:
Enter new home directory: //concert/mahler         New home directory entered
Enter new shell:
Password valid [y/n]?
Enter new expiration date [yy/mm/dd or 'none']:
Account valid for login [y/n]?
change account=> Enter account id [p.g.o]:   <Return> exits from the change command
edrgy=>
```

B.2.8 Changing Defaults

Use the **edrgy defaults** command to change the defaults used (and displayed) by the **edrgy** command.

The following example shows the **edrgy** command used to change the default owner for person entries from **%.rgy_admin.%** to **%.composers.%**.

```
edrgy=> defaults
  Person entries:
    Owner: %.rgy_admin.%
  Group entries:
    Owner: %.rgy_admin.ics
  Org entries:
    Owner: %.rgy_admin.ics
  Accounts:
    Abbreviation type: p
    Password: "-apollo-"
    Misc info: ""
    Home directory: "/"
    Shell: ""
    Password is: valid
    Account expires:  N/A          Entering a y here indicates you want to make changes
    Account is: valid
Do you wish to make changes [y/n]? (n) y
Enter new default owner for person entries [p.g.o]: (%.rgy_admin.%) %.composers.%
Enter new default owner for group entries [p.g.o]: (%.rgy_admin.ics)
Enter new default owner for org entries [p.g.o]: (%.rgy_admin.ics)
Enter new default abbreviation type [p/pg/pgo]: (p)
Enter new default password: (-apollo-)
Enter new default misc info in quotes: ()
Enter new default home directory: (/)
Enter new default shell: ()
Password valid [y/n]? (y)
Enter new default expiration date [yy/mm/dd or 'none']: (none)
Account valid [y/n]? (y)
edrgy=>
```

B.2.9 Changing Domains

Once **edrgy** is invoked, you can use the **do** command to change domains at any time. To do so, follow the instructions in Table B–7.

Table B–7. Syntax to Change Domains in **edrgy**

Enter...	**To Add, Change, or View...**
do person	Persons
do group	Groups
do org	Organizations
do account	Accounts

The following example shows how to change to the person domain.

```
$ /etc/edrgy
edrgy=> do person
edrgy=>
```

B.2.10 Changing Groups

Use the **edrgy change** command to change any group information. You must be in the group domain to change groups.

The following example shows the use of the **edrgy change** command to change the project list inclusion property from **y** (will appear in project lists) to **n** (will not appear in project lists). The example assumes you are in the group domain.

```
edrgy=> change
change group=> Enter name: symphonists       Group to change identified
Enter new name: (symphonists)
Enter new UNIX number: (78)                                  <Return> leaves the infor-
Enter new full name in quotes: (symphonist composers)        mation unchanged (as
Enter new owner [p.g.o]: (%.rgy_admin.%)                     shown in parentheses)
Enter new password:
Include group on PROJLIST [y/n]? (y) n                        Project list inclusion
change group=> Enter name:    <Return> exits from             property changed
edrgy=>                       the change command              from Y to N
```

B.2.11 Changing Persons

Use the **edrgy change** command to change any person information except the name's alias. (Aliases cannot be changed, you must delete and re–enter them.)

The following example shows the **edrgy change** command used to change the owner of a person from **%.rgy_admin.%** to **%.rgy_admin.classic**. The example assumes you are in the person domain.

```
edrgy=> change                                  Change command invoked
change person=> Enter name: mahler                  Person to change identified
Enter new name: (mahler)
Enter new UNIX number: (24583)                  <Return> leaves the information
Enter new full name in quotes: (Gustav Mahler)  unchanged (as shown in parentheses)
Enter new owner [p.g.o]: (%.rgy_admin.%) %.rgy_admin.classic      New owner
change person=> Enter name:                                        is entered
edrgy=>                                <Return> exits from the
                                       change command
```

NOTE: If you change a UNIX ID, objects external to the registry, such as the files owned by the old UNIX ID, do not automatically show the new UNIX ID as their owner. To remedy this on nodes that run Domain/OS, run the **syncids** command in the form:

```
/etc/syncids
```

To remedy this on nodes that run the UNIX operating system, use the **find** command. For example, if you change a person's UNIX ID from 333 to 444, you can run **find** in the following form to find all files owned by UNIX ID 333 and change the owner to UNIX ID 444.

```
find / -user 333 -exec chown 444 {} \;
```

Depending on how the files are protected, you may need to be root to run the **find** command.

B.2.12 Deleting Accounts

Use the **edrgy delete** command to delete accounts. You must be in the account domain to delete accounts, and you must supply the account SID with the **delete** command.

The following example illustrates the use of **edrgy** to delete the account **mahler.symphonists.classic**. The example assumes you are in the account domain.

```
edrgy=> delete mahler.symphonists.classic
Please confirm delete of "mahler.symphonists.classic" [y/n]: (n) y
edrgy=>
```

B.2.13 Deleting Aliases

Use the **edrgy delete** command to delete aliases. To delete aliases, you must be in the person domain. To use the command, enter it followed by the alias name.

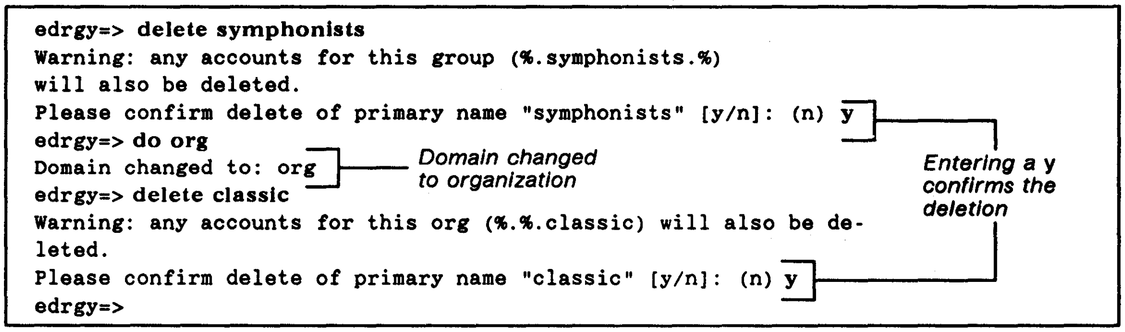
The following example deletes the alias **gus** for the person **mahler**.

```
edrgy=> delete gus
edrgy=>
```

B.2.14 Deleting Groups and Organizations

Use the **edrgy delete** command to delete groups and organizations. You must be in the group domain to delete groups and the organization domain to delete organizations.

The following example shows the **edrgy** command used to delete the group **symphonists** and then the organization **classic**. The example assumes you are in the group domain.



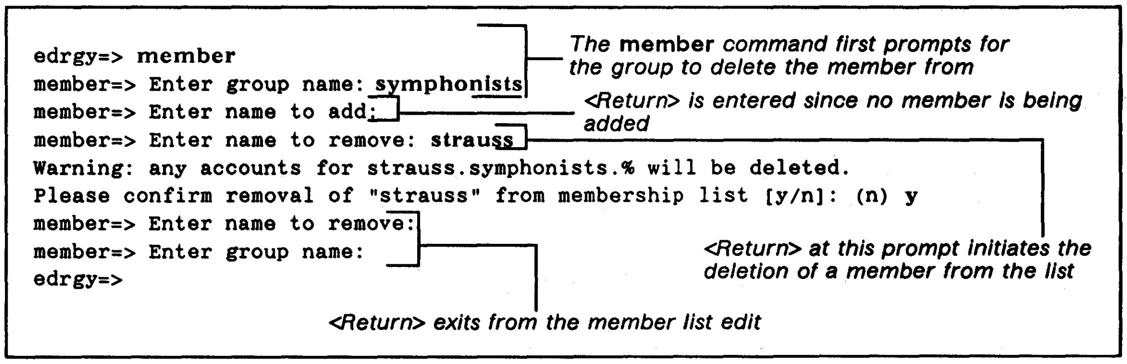
```
edrgy=> delete symphonists
Warning: any accounts for this group (%.symphonists.%)
will also be deleted.
Please confirm delete of primary name "symphonists" [y/n]: (n) y
edrgy=> do org
Domain changed to: org
edrgy=> delete classic
Warning: any accounts for this org (%.%.classic) will also be de-
leted.
Please confirm delete of primary name "classic" [y/n]: (n) y
edrgy=>
```

Domain changed to organization
Entering a y confirms the deletion

B.2.15 Deleting Members From Groups and Organizations

Use the **edrgy member** command to delete members from groups and organizations. You must be in the group domain to delete members from groups and the organization domain to delete members from organizations.

The following example shows the **edrgy** command used to delete **strauss** from the group **symphonists**. The example assumes you are in the group domain.



```
edrgy=> member
member=> Enter group name: symphonists
member=> Enter name to add:
member=> Enter name to remove: strauss
Warning: any accounts for strauss.symphonists.% will be deleted.
Please confirm removal of "strauss" from membership list [y/n]: (n) y
member=> Enter name to remove:
member=> Enter group name:
edrgy=>
```

The member command first prompts for the group to delete the member from
<Return> is entered since no member is being added
<Return> at this prompt initiates the deletion of a member from the list
<Return> exits from the member list edit

B.2.16 Deleting Persons

Use the **edrgy delete** command to delete persons. You must be in the person domain to delete persons.

The following example shows **edrgy** used to delete the person **mahler**. The example assumes you are in the person domain.

```
edrgy=> delete mahler ]------------------ The primary name is entered
Please confirm delete of primary name
"mahler" [y/n]: (n) y ]------------------------ Entering a y confirms
edrgy=>                                          the deletion
```

B.2.17 Displaying Accounts

Use the **edrgy view** command (abbreviated to **v**) to display account information. You must be in the account domain to display accounts.

Table B–8 shows the **edrgy view** command syntax used to display accounts. An example that displays the account **mahler.symphonists.classic**, including the account's administrative information, follows the table.

Table B–8. **edrgy** *Syntax to Display Accounts*

Enter...	To Display....
v	All accounts in the registry database.
v *p.g.o*	A specific account. *p.g.o* is the account SID. You can use a wildcard character (%) in place of *p*, *g* or *o*. If you enter only the person name, **edrgy** displays all accounts for that person.
v *p.g.o* **–f**	An account's administrative information.

```
edrgy=> do account
Domain changed to: account
edrgy=> v mahler.synphonists.classic -f
mahler[.symphonists.classic]:6XPbd13hiftzTE:24583:12:://fugue/mahler::
  created by: root.locksmith.r_d  1987/05/26.08:12       ]
  changed by: atatt.rgy_admin.ics  1988/08/31.08:16      ]__ Administrative information
  password is: valid, was last changed: 1990/01/10.08:12 ]   displayed by the -f option.
  account expires: N/A  account is: valid                ]
edrgy=>
```

B.2.18 Displaying Groups and Organizations

Use the **edrgy** view command (abbreviated to v) to display group and organization information. You must be in the group domain to display groups and the organization domain to display organizations.

Table B–9 shows the **edrgy view** command syntax used to display groups and organizations. Following the table is an example that displays the group **symphonists,** including its members and full information, and the policy information for the organization **classic.**

Table B–9. **edrgy** *Syntax to Display Groups and Organizations*

Enter...	**To Display...**
v	All groups or organizations in the registry database. (If you are in the group domain, **edrgy** displays all groups. If you are in the organization domain, **edrgy** displays all organizations.)
v [*name* | *UNIX_ID*]	A specific group or organization. You must enter the appropriate name or UNIX ID.
v [*name* | *UNIX_ID*] –f	A group's or organization's full information, which includes: • The group's or organization's primary and full name, UNIX ID, and UID. • The owner of the group or organization. • An indication of whether the primary name is an alias: **PR** — Primary name **AL** — Alias • The group's project list inclusion flag: **NL** — Cannot be included on project lists **L** — Can be included on project lists • An indication that the group or organization is a reserved name and cannot be deleted (the letters **RQ**). • The group's or organization's password enclosed in quotes. If no password is assigned, only the quotes appear • The date the group or organization's password was last changed.
v [*name* | *UNIX_ID*] –m	A group's or organization's members.
v [*name* | *UNIX_ID*] –po	An organization's policy information.

```
edrgy=> do group
Domain changed to:  group
edrgy=> v symphonists
symphonists        193
edrgy=> v symphonists -f
symphonists 193  Symphony Composers 39A79238.B0004108  %.rgy_admin.ics   pr
-- 1   ""  1988/05/26.08:09
edrgy=> v symphonists -m
symphonists    193                                  The -m option displays members
  3 members:      brahms, britten, mahler
edrgy=> do org                                    Domain is changed to organization
Domain changed to:  org
edrgy=> v classic -po                          The -po option displays organization policy
classic           12
  Policy:  <classic>
    Account lifespan:    forever
    Password min len:    0
    Password lifespan:   forever
    Passwords expire:    N/A
    Passwords MAY be all spaces, MAY be all alphanumeric.
  Effective Policy:
    Account lifespan:    forever
    Password min len:    0
    Password lifespan:   forever
    Passwords expire:    N/A
    Passwords MAY be all spaces, MAY be all alphanumeric.
edrgy=>
```

B.2.19 Displaying Persons

Use the **edrgy** view command (abbreviated to v) to display person information. You must be in the person domain to display persons.

Table B-10 shows the **edrgy view** command syntax used to display persons. An example that displays the person **mahler**, including the person's full information and the groups of which the person is a member, follows the table.

Table B-10. **edrgy** *Syntax to Display Persons*

Enter...	**To Display...**
v	All persons in the registry database.
v [*name* | *UNIX_ID*] v [*name* | *UNIX_ID*] -f	A specific person. You must enter the person's name or UNIX ID. A specific person's full information, which includes: • The person's primary and full name, UNIX ID, and UID. • The owner of the group or organization. • An indication of whether the primary name is an alias: **PR** — Primary name **AL** — Alias
v [*name* | *UNIX_ID*] -m	All groups of which the specified person is a member.

```
edrgy=> do person
Domain changed to: person
edrgy=> v mahler                           The -f option displays the
mahler     24583                           person's full information
edrgy=> v mahler -f
mahler 24583  Gustav Mahler  37FC556C.30004108  %.rgy_admin.ics  pr --
edrgy=> v mahler -m          The -m option displays the person's groups
mahler     24583
  Member of 1 group:
  symphonists
edrgy=>
```

B.2.20 Displaying Registry Policies and Properties

Use the **edrgy prop** command to display policies and properties.

The following example shows the use of the **edrgy prop** command to display policies and properties.

```
edrgy=> prop ──────── entering the prop command initiates the
Registry Properties:  display of policies and properties
    Registry Owner: %.rgy_admin.ics
    Person   Owner: %.rgy_admin.%
    Group    Owner: %.rgy_admin.ics
    Org     Owner: %.rgy_admin.ics
    UNIX restrictions: are NOT enforced, are NOT met
    Registry is NOT read-only
  Registry Policy:
    Account lifespan:   forever
    Password min len:   0
    Password lifespan:  forever
    Password expiration date: none
    Passwords MAY be all spaces, MAY be all alphanumeric.
  Override Policy For "domain" Machines:
    Password exclusion is ALLOWED
    Root passwords MAY be overridden
    Non-root passwords MAY be overridden
    Non-password data MAY be overridden
  Override Policy For "non_domain" Machines:
    Password exclusion is ALLOWED
    Root passwords MAY be overridden
    Non-root passwords MAY be overridden
    Non-password data MAY be overridden     edrgy prompts for whether you want to
Do you wish to make changes [y/n]? (n) ──── make changes to the policies and properties
```

B.2.21 Exiting from the edrgy Command

Use the quit command to exit from **edrgy**. The following example illustrates the use of the quit command.

```
edrgy=> quit
$
```

B.2.22 Invoking the edrgy Command

The following example shows how to invoke the **edrgy** command.

```
$ /etc/edrgy
edrgy=> ──────── This prompt indicates you are in the edrgy environment
```

When **edrgy** is invoked as in the previous example, you are in the account domain where you can add, change and view accounts. You can invoke **edrgy** in the person, group, or organization domain. To do so, follow the instructions in Table B-11.

Table B-11. Syntax to Invoke **edrgy**

Enter...	**To Invoke edrgy...**
/etc/edrgy -p	In the person domain and add, change, or view persons.
/etc/edrgy -g	In the group domain and add, change, or view groups.
/etc/edrgy -o	In the organization domain and add, change, or view organizations.

B.2.23 Setting Organization Policies

Use the **edrgy change** command to set and change policies for an organization. To use this command first set the domain to org, then enter the change command. When you enter the command, **edrgy** first displays existing policies and procedures. Then it prompts for whether you want to change them. To make a change, enter y in response to the prompt. **edrgy** then prompts for changes to each policy and procedure individually. Enter the appropriate change or press RETURN to leave the value unchanged. (Current values are displayed in parentheses after the prompt.)

The following example shows the use of the **edrgy prop** command to change policies and procedures.

```
edrgy=> do org                                   Domain changed to organization
Domain changed to: org
edrgy=> change classic                  change command and name of organization
Enter new name: (classic)               to be changed is entered
Enter new UNIX number: (12)
Enter new fullname in quotes: (No Organization)    <Return> leave basic organization
Enter new owner [p.g.o]: (%.rgy_admin.ics)         information unchanged
Enter new password:
Do you wish to enter policy information [y/n]? (n) y    y indicates changes to policies
Enter acct lifespan in days or 'forever': (forever)
Enter password minimum length: (0)
Enter password lifespan in days or 'forever': (forever) 180
Enter password expiration date [yy/mm/dd or 'none']: (none)
May passwords be all spaces [y/n]? (y)              Password lifespan is changed to
May passwords be all alphanumeric [y/n]? (y)        180; <return> leaves all other
edrgy=>                                             policy information unchanged
```

B.2.24 Setting Registry Policy and Properties

Use the **edrgy prop** command to set and change registry policies and properties. When you use this command, **edrgy** first displays existing policies and procedures. Then it prompts for whether you want to change them. To make a change, enter y in response to the prompt. **edrgy** then prompts for changes to each policy and procedure individually. Enter the appropriate change or press RETURN to leave the value unchanged. (Current values are displayed in parentheses after the prompt.)

The following example shows the use of the **edrgy prop** command to change policies and procedures.

```
edrgy=> prop
Registry Properties:
    Registry Owner: %.rgy_admin.ics
    Person   Owner: %.rgy_admin.%
    Group    Owner: %.rgy_admin.ics
    Org      Owner: %.rgy_admin.ics
    UNIX restrictions: are NOT enforced, are NOT met
    Registry is NOT read-only
  Registry Policy:
    Account lifespan:   forever
    Password min len:   0
    Password lifespan:  forever
    Password expiration date: none
    Passwords MAY be all spaces, MAY be all alphanumeric.
  Override Policy For "domain" Machines:
    Password exclusion is ALLOWED
    Root passwords MAY be overridden
    Non-root passwords MAY be overridden
    Non-password data MAY be overridden
  Override Policy For "non_domain" Machines:
    Password exclusion is ALLOWED
    Root passwords MAY be overridden
    Non-root passwords MAY be overridden
    Non-password data MAY be overridden
Do you wish to make changes [y/n]? (n) y
Enter new registry owner [p.g.o]: (%.rgy_admin.ics)
Enter new person owner [p.g.o]: (%.rgy_admin.%)
Enter new group owner [p.g.o]: (%.rgy_admin.ics)
Enter new org owner [p.g.o]: (%.rgy_admin.ics)
Enforce UNIX restrictions [y/n]? (n)
Stamp registry read-only [y/n]? (n)
Enter acct lifespan in days or 'forever': (forever) 365
Enter password minimum length: (0)
Enter password lifespan in days or 'forever': (forever) 180
Enter password expiration date [yy/mm/dd or 'none']: (none)
May passwords be all spaces [y/n]? (y)
May passwords be all alphanumeric [y/n]? (y)
```

entering the **prop** *command initiates the display of policies and properties*

edrgy *prompts for whether you want to make changes and y is entered to answer yes*

<Return> accepts the displayed default

Account lifespan changed to 365 days

Password lifespan changed to 180 days

Setting Override Policy

After you press RETURN at the "May passwords be all alphanumeric [y/n]? (y)" prompt, **edrgy** first prompts for whether you want to change override policy for machines that run Domain/OS and then whether you want to change override policy for machines that run other operating systems.

Setting Registry Policy And Properties

```
Do you wish to change override policy [y/n]? (n) y

Change override policy for "domain" machines [y/n]? (n) y
Allow password exclusion [y/n]? (y)
Allow root password overrides [y/n]? (y) n
Allow non-root password overrides [y/n]? (y)
Allow non-password data overrides [y/n]? (y)

Change override policy for "non_domain" machines [y/n]? (n) y
Allow password exclusion [y/n]? (y)
Allow root password overrides [y/n]? (y) n
Allow non-root password overrides [y/n]? (y)
Allow non-password data overrides [y/n]? (y)
```

Entering y *applies the changes to machines that run DomainIOS*

Entering n *disallows the overrides*

Entering y *applies the changes to machines not running DomainIOS*

Entering n disallows the overrides

Appendix C

Command and File Reference

This appendix contains the manual pages for commands and files associated with the registry, The registry commands, which are part of the server software that runs only on Apollo machines, are:

- **edrgy** — To edit the registry database
- **rgy_admin** — To admininster registry servers
- **rgy_create** — To create a new registry database
- **rgy_merge** — To merge the contents of two registry databases
- **import_passwd** — To create registry entries based on information in UNIX group and password files

The files, which are part of the client software that runs on Domain/OS and under Ultrix 2.2 and SunOS 3.4, 3.5, and 4.0, are:

- **passwd_override** — To create overrides to the registry database
- **passwd_refresh** — To create password and group files on workstations that do not run Domain/OS
- **pname_resolve** — To accommodate registry calls from file systems other than Domain/OS

The following UNIX commands have been changed to accommodate the registry: **login, passwd, rlogin** and **su**. Manual pages for these commands are not provided here because they depend on the environment the commands run in and the operating system under which they run. The man pages for these commands are available online.

NAME

edrgy – edit the network registry database

SYNOPSIS

/etc/edrgy [**–a** | **–p** | **–g** | **–o**] [**–l**] [**–s** *//site*] [**–synch**] [**–v**]

DESCRIPTION

The **edrgy** tool views and edits information in the registry database. You can invoke **edrgy** from any node.

Though anyone can read information in the registry database, you can usually change information only if you own the affected database entries. For example, only the owner of a group can add a name to the group's membership list.

With **edrgy**, you can edit and view names, accounts, and policies in the network registry, as well as entries in the local registry. The tool operates in one of four domains: person names, group names, organization names, and accounts.

OPTIONS

You can specify only one of **–a**, **–p**, **–g**, and **–o**.

–a (default)	Edit or view accounts.
–p	Edit or view persons.
–g	Edit or view groups.
–o	Edit or view organizations.
–l	Edit or view entries in local registry.
–s	Use the specified registry site.
–synch	Synchronize local registry with network registry.
–v	View selected entries.

Unless you specify the **–v** option, **edrgy** operates interactively. The following sections describe the commands you can enter in the interactive mode.

COMMANDS FOR PERSONS, GROUPS, AND ORGANIZATIONS

v[iew] [*name* | *number*] [**–f**] [**–m**] [**–po**]

View *name* entries.

If you specify a *number*, **edrgy** displays all matching entries, including any aliases.

The **–f** option displays entries in full (all fields except the membership list and organization policy).

If you are viewing groups or organizations, **–m** displays the membership list. For persons, **–m** lists all groups of which the person is a member, including groups that cannot appear in a project list.

If you specify **–po** while viewing organizations, **edrgy** displays policy information. Otherwise, it shows only the name and the UNIX number.

a[dd] [*person number* [*fullname*] [**–al**] [**–o** *owner*]]
a[dd] [*group number* [*fullname* [*password*]] [**–nl**] [**–o** *owner*]]
a[dd] [*organization number* [*fullname* [*password*]] [**–o** *owner*]]

Create a new name entry.

If you do not specify a *person*, *group*, or *organization* name, the **add** command enters an interactive mode and prompts you for each field in the entry. If you are adding organizations in the interactive mode, the command prompts you for policy information as well.

Specify the *owner* as a *person.group.organization* triplet. You can use % as a wildcard for any or all of the components. If you do not use the **–o** option, **edrgy** assigns the default owner, which you can set or display with the **defaults** command.

For persons, the **–al** option creates an alias entry. If *number* (the UNIX number) is already assigned to a person and you do not specify **–al**, an error occurs and you must either choose a different *number* or specify **–al**. If you use **–al** to create an alias and *number* is not already associated with a primary name, **edrgy** issues a warning but creates the alias.

For groups, the **–nl** flag indicates that the group is not to be included on project lists; omitting this flag allows the group to appear on project lists.

For groups and organizations, a space between quotation marks indicates a nil password.

Use quotation marks to embed spaces (or quotation marks) in a *fullname*. A single space between quotation marks indicates a nil *fullname*.

c[hange] [*person* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–al** | **–pr**]]
c[hange] [*group* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–p** *password*] [**–nl** | **–l**]]
c[hange] [*organization* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–p** *password*]]

Change a *name* entry.

If you do not specify a *person*, *group*, or *organization* name, the **change** command enters an interactive mode and prompts you for a name. If you do not specify any fields, the command prompts you for each field in succession. To leave a field unchanged, press <RETURN> at the prompt. If you are changing organization entries in the interactive mode, the command prompts you for policy information as well.

For person entries, the **–al** flag changes a primary name into an alias, while the **–pr** flag changes an alias into a primary name. This change can be made only from the command line, not in the interactive mode.

For group entries, the **–nl** flag disallows the group from appearing in project lists, while the **–l** flag allows the group to appear in project lists.

For organization entries, you can change policy information only in the interactive mode.

A single space between quotation marks indicates a nil *fullname* or *password.*

Specify the *owner* as a *person.group.organization* triplet. You can use % as a wildcard for any or all of the components.

Changes to a person name are reflected in membership lists that contain the person name. For example, if the person **ludwig** is a member of the group **composers** and the person name is changed to **louis**, the membership list for **composers** is automatically changed to include **louis** but not **ludwig**.

Changes to *number* (the UNIX number) cause the operating system to change its mapping of the UID, the primary name, and any aliases from the old *number* to the new one. However, files owned by the old *number* do not automatically show the new *number* as their owner.

The only fields of reserved entries that you can change are the *fullname*, the *password*, the *owner*, and (for *groups*) the property that allows a *group* to appear in project lists. If a reserved *group* is allowed to appear in project lists, you can disallow it; but if the *group* is disallowed, you cannot allow it.

m[ember] [*group* | *organization* [**–a** *member_list*] [**–r** *member_list*]]

Edit the membership list for a group or organization.

If you do not specify a group or organization, the **member** command enters an interactive mode and prompts you for names to add or remove.

The **–a** flag precedes the person names (separated by spaces) to be added to the membership list, while the **–r** flag precedes those to be removed. If you do not include either flag on the command line, **edrgy** prompts you for names to add or remove.

Adding a person to a membership list permits creation of a login account for that person with that group or organization.

Removing *person* from the membership list for *group* has the side effect of deleting all login accounts of the form *person.group*, and likewise for organizations.

del[ete] { *person* | *group* | *organization* }

Delete a name entry.

You cannot delete reserved names. Deleting a group or organization has the side effect of deleting any accounts with that group or organization.

adopt *uid_high.uid_low person number* [*fullname*] [**–o** *owner*]
adopt *uid_high.uid_low group number* [*password* [*fullname*]] [**–nl**] [**–o** *owner*]
adopt *uid_high.uid_low organization number* [*password* [*fullname*]] [**–o** *owner*]

Create a primary name entry for the specified UID.

The UID must be an orphan (a UID for which no name exists in any domain). The *uid_high* and *uid_low* are hexadecimal numbers.

An error occurs if you specify a name or UNIX number that is already defined within the same domain of the database.

A single space between quotation marks indicates a nil *fullname* or *password.*

Specify the *owner* as a *person.group.organization* triplet. You can use **%** as a wildcard for any or all of the components. If you do not use the **–o** option, **edrgy** assigns the default *owner*, which you can set or display with the **defaults** command.

COMMANDS FOR ACCOUNTS

In all of the account operations, the *account* argument is a *person.group.organization* triplet such as **jones.graphics.research**. Unless otherwise specified, any or all of the components can be the wildcard character, **%**. For example, **view %.dev.%** views all accounts associated with the group **dev**.

In an *account* argument, if you omit a trailing *organization* (or *group.organization*), **%** (or **%.%**) is assumed. Thus, **keats.%.%**, **keats.%**, and **keats** are equivalent.

v[iew] [*account*] [**–f**]

Display login accounts specified by the *account* pgo (*person, group, organization*) triplet.

Without the **–f** flag, **view** displays only the user fields in each account entry: account SID, encrypted password, miscellaneous information, home directory, and login shell.

With **–f**, **view** displays the full entry, including the administrative fields as well as the user fields. Administrative information includes who created the account, when it was created, who last changed it, when it was last changed, when it expires, whether it is valid, whether the password is valid, and when the password was last changed.

a[dd] [*account* [**–a** { **p** | **pg** | **pgo** }] [*password* [*misc* [*homedir* [*shell*]]]]
[**–pnv**] [**–x** *account_exp* | **none**] [**–anv**]]

Create a login account.

Specify *account* as a pgo triplet. Wildcards are not allowed. If you do not supply an *account* on the command line, **add** enters an interactive mode and prompts you for each field in succession.

If the person specified in *account* is not already a member of the specified group and/or organization, **edrgy** automatically attempts to add the person to the membership lists. If you are not an owner of the group and/or organization, the attempt will fail and the account will not be created.

The **–a** flag indicates the degree of abbreviation allowed for login: **p** means that only the person is required; **pg** means the person and the group; **pgo** means that all three components of the account SID are required. (Of course, a user can always supply more components than are required.) If the abbreviation you specify is already defined for another account, **edrgy** automatically uses the shortest unique abbreviation and issues a warning.

For example, if you create an account **babar.elephants.none** with the abbreviation **p**, a user need only enter **babar** at the login prompt to use the account. If you then create an account **babar.kings.none**, the **p** abbreviation will conflict with the existing account, so the **pg** abbreviation, **babar.kings**, will be the shortest unique one.

Omitting the **–a** is equivalent to specifying **–a p** and results in use of the shortest unique abbreviation.

The *password* must adhere to the policy of the associated organization or the policy of the registry as a whole, whichever is more restrictive.

The *misc* field is not used by the operating system. The *gecos* field of each account's entry in the **/etc/passwd** file is the concatenation of the person's full name and the account's *misc*. Use quotes to include spaces, hyphens, or quotes in *misc*.

The *homedir* and *shell* are pathnames. The default *homedir* is /. The default *shell* is the null string.

Use a single space between quotation marks to indicate a nil *password*, *misc*, *homedir*, or *shell*.

The **–pnv** (password not valid) flag specifies that at the next login (for a newly created account, the first login), the user must change the password. If you omit this option, the password is valid.

The **–x** flag sets an expiration date for the account; the default is none.

The **–anv** (account not valid) flag specifies that the account is not currently valid for login. If you omit this option, the account is valid.

c[**hange**] [*account* [**–n** *new_account*] [**–a** { **p** | **pg** | **pgo** }]
[**–p** *password*] [**–m** *misc*] [**–h** *homedir*] [**–s** *shell*]
[**–pnv** | **–pv**] [**–x** *account_exp* | **none**] [**–anv** | **–av**]

Change one or more account entries.

Specify *account* as a pgo triplet. Wildcards are allowed, unless you use the **–n** option. If you do not supply an *account* on the command line, **change** enters an interactive mode and prompts you for each field in succession. Press <RETURN> to leave a field unchanged.

The command line arguments are largely the same as those of the **add** command. The **–n** flag enables you to change the account SID to *new_account*, a pgo triplet that cannot contain wildcards. The **–pv** flag specifies that the password is valid. The **–av** flag specifies that the account is valid.

You can enter a single space between quotation marks to indicate a nil *password*, *misc*, *homedir* or *shell*.

del[ete] *account*

Delete the entry for *account*, a pgo triplet that cannot contain wildcards.

MISCELLANEOUS COMMANDS

do[main] [p | g | o | a]

Change or display the type of registry information being viewed or edited.

You can specify **p** for persons, **g** for groups, **o** for organizations, or **a** for accounts. If you supply no argument, **edrgy** displays the current domain.

s[ite] [*//site* **] [–l]**

Change or display the registry site being viewed or edited.

If you specify a *//site*, **edrgy** attempts to use the registry server at the named site. If you specify **-l**, **edrgy** uses the local registry. If you supply no argument, **edrgy** displays the current site.

prop[erties]

Change and/or display the registry properties and policies.

This command prompts you for any changes to make. Press <RETURN> to leave information unchanged.

synch[ronize]

Update the local registry to match the master registry.

If a matching entry cannot be retrieved from the network registry, the local entry is marked invalid for login, and its UNIX numbers are updated.

co[py] [*account* **]**

Copy information for the specified accounts from the master registry to the local registry.

The *account* is a pgo triplet that can contain wildcards; trailing wildcard components can be omitted. If a matching account already exists in the local registry, **edrgy** updates the information to match that in the master registry; otherwise, **edrgy** adds the entry. If all entries in the local registry are used, **copy** reports an error and terminates.

def[aults]
: Change and/or display the default values that **edrgy** uses.

h[elp] [*command*]
: Display usage information for **edrgy**.

  If you do not specify a particular command, **edrgy** lists the available commands.

q[uit]
: Exit **edrgy**.

COMMANDS VALID FOR THE LOCAL REGISTRY

To edit or view the local registry, use the **–l** flag when you invoke **edrgy**. This section lists the commands that are valid for editing or viewing the local registry. Unless otherwise specified, all options are as described in the previous command descriptions.

v[iew] [*name* | *number*] [**–f**] [**–po**]
: View name entries. (The **–m** option is not valid.)

v[iew] [*account*] [**–f**]
: Display specified login accounts.

c[hange] [*account* [**–a** { **p** | **pg** | **pgo** }] [**–m** *misc*] [**–h** *homedir*] [**–anv**]
: Change one or more account entries. (The **–p**, **–s**, **–pnv**, **–pv**, **–x**, and **–av** options are not valid.)

del[ete] *account*
: Delete an account entry.

do[main] [**p** | **g** | **o** | **a**]
: Change or display the type of registry information being viewed or edited.

s[ite] [*//site*] [**–l**]
: Change or display the registry site being viewed or edited.

prop[erties]
: Change and/or display the registry properties and policies.

synch[ronize]
: Update the local registry to match the master registry.

co[py] [*account*]
: Copy information for the specified accounts from the master registry to the local registry.

def[aults]
: Change and/or display the default values that **edrgy** uses.

h[elp] [*command*]
: Display usage information for **edrgy**.

q[uit]
: Exit **edrgy**.

SEE ALSO

cvtrgy(1)

NAME

rgy_admin – registry server administrative tool

SYNOPSIS

/etc/rgy_admin

DESCRIPTION

The **rgy_admin** tool administers registry servers. It can view or modify the registry replica list, reinitialize replicas, delete replicas, stop servers, and change the registry master site.

Note that **rgy_admin** cannot add, delete, or modify data entries contained in the registry database, such as names and accounts; use **edrgy** to perform these tasks. To create a registry replica or to restart a server, use **rgyd**, the registry daemon.

Once invoked, **rgy_admin** enters an interactive mode in which it accepts the commands described in the next section.

COMMANDS

Most **rgy_admin** commands operate on a default host. You use the **set** command to establish the default host, which is remembered until changed by another **set**. In the following command descriptions, we identify the default host as *default_host*. If a command operates on a host other than the default, we identify this host as *other_host*.

Several of the **rgy_admin** commands require you to set the default host to the master registry site.

The host name you supply as a *default_host* or *other_host* takes the form *family*:*host* or *host*. The only currently supported *family* is **dds**, the Domain protocol family. You can specify a host in this family by its entry directory or by its network address. For example, **dds://clara**, **//clara**, **dds:#1234.abcd**, and **#1234.abcd** are all acceptable host names.

become [–master] [–slave] [–ro | –wr]

The **–master** option causes the replica at *default_host* (which must be a slave replica) to become the master. This operation can cause updates to be lost; the **change_master** command is the preferred means of designating a different master replica.

The **–slave** option causes the replica at *default_host* (which must be the master replica) to become a slave. This operation can cause updates to be lost; the **change_master** command is the preferred means of designating a different master replica.

The **–ro** option makes the replica list read-only. The *default_host* must be set to the master registry site.

The **–wr** option makes the replica list writable. The *default_host* must be set to the master registry site.

change_master –to *other_host*

Change the master replica of the registry from *default_host* to *other_host*. The *default_host* must be set to the master registry site.

The current master server copies its database to the replica at *other_host*, becomes a slave, then tells the replica at *other_host* to become the master.

delrep *other_host* [**–force**]

Delete the registry replica at *other_host*. The *default_host* must be set to the master registry site.

The master server marks the replica at *other_host* as deleted and propagates the deletion to all other replicas on its list. When it has actually delivered the delete request to the replica at *other_host*, the master server removes that replica from its own replica list.

The **–force** option causes a more drastic delete. It deletes *other_host* from the replica list at the master registry, which then propagates the delete request to the replicas at the hosts that remain on its list. Since this operation never communicates with the deleted replica, you should use **–force** only when the replica has died irrecoverably. If you use **–force** while the replica at *other_host* is still running, you should then **reset** the deleted replica.

help List the **rgy_admin** commands and show their allowed abbreviations.

info Get status information about the replica at *default_host*.

initrep *other_host*

Reinitialize the registry server at *other_host*. The *default_host* must be set to the master registry site. The *other_host* must be a slave site.

The master registry copies its entire database (or that of another up-to-date replica) to the replica at *other_host*.

lrep [**–state**] [**–na**]

List the registry replica sites as stored in the replica list at *default_host*.

The **–state** option shows the current state and update time on each host.

The **–na** option shows the network address of each host.

monitor [**–r** *m*]

Periodically list the registry replica sites as stored in the replica list at *default_host* and show the current state and update time at each site.

The **–r** option causes the sites to be listed every *m* minutes. If you omit this option, the period is 15 minutes.

quit Quit the **rgy_admin** session.

reprep *other_host*
: Replace the network address for *other_host* in the registry replica list. The *default_host* must be set to the master registry site.

  The master replica propagates the new network address for *other_host* to all other registry replica lists. Use this command only if a replica site's network number changes.

reset *other_host*
: Reset the registry replica at *other_host*. The registry server at *other_host* deletes its copy of the registry and stops running. This command does not delete *other_host* from any replica lists.

set [–h *host_name* **| –m]**
: Set the default host. Subsequent commands that do not specify a host will go to this host.

  The **–h** option specifies a replica to use as the default.

  The **–m** option causes the master replica to be the default.

  With no options, **set** locates a registry replica and sets it as the default.

site [*host_name*]
: If *host_name* is specified, the command sets the default host.

  If *host_name* is not specified, the command gets status information about *default_host*.

state –in_maintenance | –not_in_maintenance
: Put the master registry server into maintenance state or take it out of maintenance state. The *default_host* must be set to the master registry site.

  With the **–in_maintenance** flag, **state** causes the master registry server to save its database onto disk and refuse any updates.

  With the **–not_in_maintenance** flag, **state** causes the master registry server to reload its database from disk, return to its normal "in service" state, and (if its database and/or replica list are writable) start accepting updates.

stop
: Stop the registry server that is running at *default_host*.

EXAMPLES

Start **rgy_admin**, list the replicas and their states, then set the default host to the master replica:

```
$ /etc/rgy_admin
Default object: rgy  default host: dds://george
State: in service  slave
rgy_admin: lrep –st
  (master) dds://martha  state: in service  1987/11/16.12:46:59
           dds://george  state: in service  1987/11/16.12:46:59
           dds://thomas  state: in service  1987/11/16.12:46:59
rgy_admin: set –m
          Default object: rgy  default host: dds://martha
          State: in service  master  replica list is writeable
```

NAME

rgy_create – registry creation utility

SYNOPSIS

/install/tools/rgy_create

DESCRIPTION

The **rgy_create** tool creates a new registry database on the local node, initialized with reserved names and accounts. It ordinarily should be run only once at a site. Replicas of the database are created by running **rgyd** with the **–create** option.

You must be root to invoke **rgy_create**.

Note that to convert a pre-SR10 registry to SR10 format, you should run only the **cvtrgy** tool, and you will never need to use **rgy_create**.

NAME

rgy_merge – merge registry database

SYNOPSIS

rgy_merge –from *//site* [{ **–merge** | **–compare** } **–verbose**]

DESCRIPTION

The **rgy_merge** utility merges the contents of two registry databases, a source database and a target database. You typically use it when you are joining two networks that have been operating separately and you want to combine their registry databases.

You must invoke **rgy_merge** while logged in as root at the master registry node for the target registry. Use the required **–from** *//site* argument to specify the master registry node for the source registry. The merge takes less time if the source database is smaller than the target database.

After you invoke **rgy_merge**, the tool prompts you to "login" with an account that owns the source registry.

Without a **–merge** or **–compare** option, **rgy_merge** attempts to add each entry in the source database to a copy of the target database and reports any conflicts in names or UNIX numbers. If there are no conflicts or errors, the tool asks whether you want to actually update the target database. If you respond affirmatively, it performs the merge on the target database and makes all replicas of the source registry into slave replicas of the target registry.

If you specify **–merge**, **rgy_merge** performs the merge without querying you, provided there are no conflicts or errors. If you specify **–compare**, **rgy_merge** only checks for conflicts and does not perform a merge even if there are none.

The **–verbose** option causes **rgy_merge** to generate a verbose transcript of its activity.

NAME

import_passwd – create registry entries based on information in UNIX group and password files

SYNOPSIS

/etc/import_passwd **[–i]** **[–a | –f]** **[–c]** [**–o** *org*] **–s** *pathname* [**–v**]

DESCRIPTION

import_passwd is a mechanism for creating Domain/OS registry entries that are consistent with foreign password and group file entries. You should use **import_passwd** to ensure consistency between Domain/OS and foreign protection mechanisms when you

- Attach Apollo node(s) to an existing UNIX network
- Attach UNIX node(s) to an Apollo network
- Connect Apollo and UNIX networks

If the foreign password and group file entries do not exist in the Domain/OS registry, **import_passwd** will create them. If there are duplicate entries, **import_passwd** will follow your directions on how to handle them. (Note that reserved names and reserved UNIX IDs cannot be reassigned.)

The Process

The Domain/OS registry must exist before you can use **import_passwd.** If you are simply adding a few Apollo nodes to a foreign network, you can create a new, but empty, registry to meet this requirement. Once the registry exists, the registry server must be running, and you must be logged in as root.

As **import_passwd** processes, it

1. Examines the foreign group file and creates group entries in the registry.
2. Examines the foreign password file and creates person, organization, and account entries in the registry. The organization assigned is specified as input to **import_passwd.**
3. Re-examines the foreign group file and creates membership lists.

Conflicts

During this process, **import_password** may find conflicts in name strings (for example, in the foreign network, joe 102; in the registry, joe 555) and in UNIX IDs (for example, in the foreign network, joe 102; in the registry, ann 102). **import_passwd** provides a number of options to help resolve these conflicts.

The Favored Entry

The **–a** (favor registry entry) or **–f** (favor foreign entry) options specify which entry should be favored. A favored entry is retained as is. You are prompted to modify non-favored entries. (Note, however, that in some cases you may be prompted to modify a favored entry. For example, if the non-favored entry is a reserved name, you will be prompted to modify the favored entry.)

Name Conflicts

The **–i** option specifies that duplicate names are not in conflict but in fact, represent the same identity. Therefore, when duplicate names arise, no action is necessary. If you do not use the **–i** option, **import_passwd** resolves name conflicts by prompting for name strings for non-favored entries.

UNIX ID Conflicts

The resolution of UNIX ID conflicts is also determined by the favored entry. If a conflict exits, you are prompted for a new UNIX ID for the non-favored entry.

Other Registry Entries

Except for names and UNIX IDs, all other information stored in the registry for an existing identity is retained.

New registry entries created by **import_passwd** are assigned the following values:

For Person and Group Entries:

fullname = " (empty)

owner = Same as the owner of the organization specified with the **–o** option. If no organization is specified, then the owner is the organization named "none".

alias/primary = Primary for first entry; alias for subsequent ones.

projlist_ok = Yes.

passwd = For groups only, taken from the group file.

membership list = For new groups only, all persons listed in the group file, and all persons with accounts in the password file with that group.

For Account Entries:

abbreviation = Shortest possible abbreviation that does not conflict with pre-existing registry accounts.

acount_valid = True.

gecos = Same as UNIX password file.

homedir = Same as UNIX password file.

shell = Same as UNIX password file.

passwd = Same as UNIX password file. Note that you must modify or reset imported passwords before user authentication is possible and for the account to be usable in a pre-SR10 registry.

passwd_dtm = Date and time **import_passwd** was run.

passwd_valid = True.

OPTIONS

–i	Name strings are not in conflict, but represent the same identity.
–a (default)	Favor registry entries for conflicts.
–f	Favor foreign entries for conflicts.
–c	Run in check mode: Process the command, showing all conflicts, but make no requests for resolution.
–o *org*	*org* is the name of the organization to be assigned to all imported entries.
–s *pathname*	*pathname* is the path to the directory containing the foreign password and group files to be imported.
–v	Run in verbose mode: Generate a verbose transcript of **import_passwd** activity.

SEE ALSO

edrgy(8), rgy_admin(8), rgy_merge(8), rgyd(8)

NAME

/etc/rgy/passwd_override – override password, GECOS, home directory, and shell entries from the registry database.

DESCRIPTION

The **passwd_override** administrative file lets you override password, GECOS, home directory, and shell entries from the registry database. The **passwd_override** file is stored on each host machine participating in the Passwd Etc registry. Any changes you make to it are in effect for the local machine only, and have no effect on the centralized registry.

You may find **passwd_override** especially useful for excluding people from using certain machines, establishing local root passwords, or tailoring local user environments.

The override feature is enabled by policy information stored in the registry database. Using the **edrgy** command, you can control how much, if any, customization will be allowed.

PASSWD_OVERRIDE FILE FORMAT

The format of **passwd_override** entries is similar to entries in the UNIX password file. The format is

person.group.org:*passwd*:*uid*:*grp_id*:*GECOS*:*home_dir*:*shell*

Key Fields

The *person.group.org* and *uid* fields are key fields. You must enter one of them to identify the account for which you are making overrides. The following rules apply to entering these fields:

- If you enter *person.group.org*, *uid* is ignored.
- If you want to use *uid* as a key, enter *%.%.%* for the *person.group.org* subject identifier.

Field Descriptions

Each of the entries in the **passwd_override** file is described below.

person.group.org	Is the registry subject identifier that identifies accounts. The subject identifier is similar to a UNIX log-in name, but can contain a group and organization identifier. This field is a key field. You must enter it (or the UID) to identify the account.
passwd	Is the encrypted password. You can specify an override in this field. If you do, the password you enter here will be in effect for this local machine only. For no override, leave the field empty.

You can also specify "OMIT" in the *passwd* field to disallow login on the local machine. The use of OMIT in conjunction with an option to the **passwd_refresh** command prevents the inclusion of this user in the password file created by **passwd_refresh**. (See "Using OMIT," later in this command reference for details.)

uid Is the UNIX user ID. Like *person.group.org*, this entry is a key. You must enter it (or a subject identifier) to identify the account. Enter a uid when you want to apply the overrides to all a user's accounts and aliases (regardless of the group or organization). *uids* are especially useful for overrides to root (uid 0). If, for example, root has an alias of wizard, an override keyed by subject identifier applies only when root logs in as root, not when root logs in as wizard. If the override is keyed by *uid*, it applies when root logs in as root or wizard or any other alias.

grp_id Is the account's group ID. This field cannot be overridden and is ignored when the **passwd_override** file is processed. Normally you should leave this field empty.

GECOS Is the account's GECOS field. You can specify an override in this field. To keep it unchanged, leave it empty.

home_dir Is the account's home directory. You can specify an override in this field. To keep it unchanged, leave it empty.

login_shell Is the account's log-in shell. You can specify an override in this field. To keep it unchanged, leave it empty.

USING WILD CARDS TO IDENTIFY ACCOUNTS

You can use wild cards (%) for any part of the subject identifier. For example, the entry:

smith.%.%

says that overrides affect user **smith** regardless of the group and organization that **smith** logs in with. In other words, the overrides affect all accounts for user **smith**.

The entry:

smith.xproject.development

restricts the overrides to a specific account. If **smith** has other accounts, for example, **smith.newproject.development** or **smith.xproject.marketing**, these are not affected by the override.

The entry:

%.%.marketing

applies the overrides to all members of the organization **marketing**.

USING ABBREVIATIONS

If you do not completely specify the subject identifier, wild cards are assumed. For example, assume you are using **passwd_override** to change a user's home directory on a machine. To do this, you'd enter

smith:::::/node6/smith:

An entry of **smith.%.%** is assumed for the subject identifier.

USING OMIT

If you enter either the word "OMIT" or another invalid password string (such as a "*" or "NO GOOD") in the *passwd* field, the user (or set of users) will be unable to log in to the local machine. If you specify "OMIT" and run **passwd_refresh** with the **-x** option, the named user (or set of users) will not appear in the **/etc/passwd** file produced by **passwd_refresh**.

You should also be aware that if you have omitted users from the **/etc/passwd** file, information about those users will not be available to any programs that use the password file. For example, the **ls -l** and the **finger** commands both access the password file to obtain further information about a user identified by a user ID. If the user is omitted, no password entry will exist and no information will be available. For this reason, you should omit users only if your user community is very large and either of the following conditions occur:

- The **passwd** file is taking up too much space.
- User-ID-to-name mapping during, for example, **ls -l** is too slow.

OVERRIDE PRECEDENCE

When more than one override file entry applies to an acccount, the most specific entry is selected. User ID numbers and persons (*person.%.%*) are the most specific, followed by group (*%.group.%*), and organization (*%.%.org*).

For example, assume the override file contains entries with the following key fields:

- **marlena.%.%**
- **%.dds.%**

If a person logs on with the subject identifier of **marlena.dds.r_d**, the override keyed by **marlena.%.%** will be in effect. In this case, person (**marlena**) is more specific than group (**dds**). Now assume the override file also contains the entry **marlena.dds.%**. The override keyed by **marlena.dds.%** will be in effect because person and group are more specific than person alone (**marlena.%.%**) and more specific than group alone (**%.dds.%**).

EXAMPLES

1. To prevent users from the **evil_users** organization from logging in to a node, the entry in the **passwd_override** file would be:

   ```
   %.%.evil_users:AN_INVALID_STRING:::::
   ```

   Note that entries for members of the **evil_users** group will be present in the **/etc/passwd** file.

2. To prevent users in the **evil_users** organization from logging in to a node and to omit them from inclusion in the password file, first use the **passwd_override** command and enter **OMIT** in the *passwd* field:

   ```
   %.%.evil_users:OMIT:::::
   ```

   Then run the following **passwd_refresh** command with the **-x** option to cause these users to be omitted from **/etc/passwd** file

   ```
   /etc/rgy/passwd_refresh -x
   ```

3. To change the password on a specific machine for the entire marketing organization, the entry would be

   ```
   %.%.marketing:newpassword:::::
   ```

4. To change the password, home directory, and initial shell for the **smith.xproject.development** account, the entry would be

```
smith.xproject.development:newpassword:::/secret/xproject:/bin/sh
```

NOTES

Users can update their entries in the override file for the local host by using **chfn**, **chhd**, **chsh**, or **passwd**. Refer to the **passwd** manual page for details.

SEE ALSO

getpwent(3), login(1), crypt(3), passwd(1), group(5), finger(1), adduser(8), edrgy(8)

NAME

passwd_refresh – create local password, group, and organization files

SYNOPSIS

/etc/rgy/passwd_refresh -f [*dir_name*] **-x**

DESCRIPTION

The **/etc/rgy/passwd_refresh** command creates local password, group, and organization files from Domain/OS registry data. These files are used when the network registry is unavailable and by programs that have not been built with **librgy.a**. Use **passwd_refresh** to keep these local files consistent with the registry.

When **/etc/rgy/passwd_refresh** runs, it makes backup copies of the current password, group, and organization files, if they exist. The files are named, respectively, **passwd.bak, group.bak**, and **org.bak**.

By default, the backups are stored and the new files created in the **/etc** directory. You can override the default by supplying a directory name as an option to the command.

The **/etc/rgy/passwd_refresh** command keeps a record of when changes are made to the registry. If changes have not been made, **/etc/rgy/passwd_refresh** prints a message saying so and does not create new files. You can force the update with the **-f** option.

OPTIONS

–f	Update the password, group, and organization files, even if no changes have been made to the registry.
[*dir_name*]	Specifies the name of the directory to store the password, group, and organization files created by **passwd_refresh**. If you do not enter a directory name, the files are stored in the **/etc** directory.
-x	Do not create entries for users with "OMIT" as their encrypted password. See the **passwd_override** command manual page for further details about omitting users.

RUNNING PASSWD_REFRESH

passwd_refresh is commonly run via an entry in **/usr/lib/crontab**. For example, to update the files every hour, the entry would be

```
0 * * * * /etc/rgy/passwd_refresh
```

In large network environments, it is a good idea to stagger the times **passwd_refresh** is run.

NAME

/etc/rgy/pname_resolve

DESCRIPTION

The **/etc/rgy/pname_resolve** file contains a substitution string for pathames beginning with the characters "//" that are returned by registry calls.

For example, assume a user's home directory registry entry is the string **//mynode/home**. If the **/etc/rgy/pname_resolve** file contains the substitution string **/apollo**, then all registry calls on this user's machine that supply the user's home directory would return **/apollo/mynode/home**.

The Domain/OS distributed file system uses the string "//" to indicate the network- wide file system root. Other network file systems (notably NFS) do not accept this syntax. The **/etc/rgy/pname_resolve** file provides a substitution string for the Domain/OS // syntax.

The **/etc/rgy/pname_resolve** file is stored locally on each host. On NFS machines, this file should contain the pathname on which the Domain file system is mounted. For example, if the Domain/OS file system is mounted on **/apollo**:

```
% /etc/mount -p
apollo_host://      /apollo     nfs    soft ,rw, time=350   0  0
```

then the **/etc/rgy/pname_resolve** file should contain the string **/apollo**.

Index

Symbols are listed at the beginning of the index.

Symbols

%, use as wildcard character in subject identifiers, 2–4

%.locksmith.%. *See* locksmith

A

–a option, to **import_passwd** command, A–4

abbreviation type, 2–14, B–3

account valid flag, 2–14, B–3

accounts
- adding, 4–1 to 4–2, 7–5
  - *See also* **edrgy** command, use to add accounts
- changing, B–9
  - *See also* **edrgy** command, use to change accounts
- deleting, B–12
  - *See also* **edrgy** command, use to delete accounts
- displaying, B–14
- reserved, 2–7
- summary of information used to create, B–3

add, edrgy command
- account, 2–15, B–7, C–6
- groups and organizations, 2–12, B–7, B–8, C–3
- persons, 2–8, B–9, C–3

administrative control, 1–1, 2–4 to 2–5

administrative tasks, 1–1

adopt
- **edrgy** command, 2–5, C–5

aliases
- and access rights, 2–11
- adding, B–7
- creating. *See* **edrgy** command, use to create aliases
- deleting, B–12
- *See also* names, alias

B

become, rgy_admin command, C–11

C

–c option, to **import_passwd** command, A–6

cell configurations, 1–7

change
- account, C–7
- persons, groups, and organizations, C–3 to C–4

change_master, rgy_admin command option, C–12

chfn command, and overrides, 4–8

chgrp command, A–7

chhd command, and overrides, 4–8

chhdir command, 4–8 to 4–11

chown command, A–7

chpass command, 4–8 to 4–11

chsh command, and overrides, 4–8

client software, 1–2

configuration, 7–2 to 7–3

copy, **edrgy** command, C–8

cron command, and **passwd_refresh** command, 7–2

D

database
- consistency, 1–5, 1–6
- contents, 1–3
- creating, 7–4
- ensuring consistency with UNIX password and group files. *See* **import_passwd command**
- and password and group files, 1–4
- procedures to back up, 4–9 to 4–11
- replication, 1–5
- saves, 1–5
- storage, 1–4
- updates, 1–7
  - propagation, 1–7
- *See also* registries

default, **edrgy** command, B–9 to B–10, C–9

delete
- account, C–8
- person, group, or organization, C–5

deletion, forced, 6–5

delrep, **rgy_admin** command option, C–12

domain, **edrgy** command, C–8

Domain/OS, and server software, 1–3

domains, changing, B–10 to B–11

drm_admin command, 6–1

E

edrgy command, 1–10
- and local and network registry, 4–8 to 4–11
- changing defaults in, B–9 to B–10
- changing domains in, B–10 to B–11
- exiting, B–18
- functions, 2–6
- invoking, B–18 to B–19
- use to add accounts, 2–15, B–7
- use to add and delete members from groups, 2–17 to 2–18, B–13
- use to add groups and organizations, 2–12
- use to add members to groups, B–8
- use to add organizations, B–8
- use to add persons, 2–8, B–9
- use to change accounts, 2–15, B–9
- use to change groups, 2–12 to 2–13, B–11
- use to change persons, 2–9 to 2–10, B–11 to B–12
- use to create aliases, 2–9, B–7
- use to create groups, B–7 to B–8
- use to delete accounts, 2–16, B–12
- use to delete aliases, B–12 to B–12
- use to delete groups, 2–13, B–13
- use to delete organizations, 2–13, B–13
- use to delete persons, 2–10, B–14
- use to display accounts, B–14
- use to display groups and organizations, B–15 to B–16
- use to display persons, B–16 to B–17
- use to display policies, 3–5 to 3–6, B–17 to B–18
- use to display properties, 3–5 to 3–6, B–17 to B–18
- use to edit local registry, 1–9
- use to set organization policy, 3–7 to 3–8, B–22
- use to set override policies, 3–7, B–20 to B–21
- use to set policies, 3–6, B–19 to B–21
- use to set properties, 3–6, B–19 to B–21

edrgy command reference, C–2 to C–10

/etc/group file, 1–8 to 1–9

/etc/org file, 1–8 to 1–9

/etc/passwd file, 1–8 to 1–9

/etc/rgy/pname_resolve file, 7–2, 7–7

/etc/rgy/rgy_data directory, 1–5

/etc/rgyd, 1–2
- *See also* server software

/etc/syncids command. *See* **syncids** command

F

-f option, to **import_passwd** command, A-4

find command, 2-10, B-12

free space required for registry servers, 7-2

fullname. *See* names, fullname

G

glbd, 7-3

global location broker daemon. *See* **glbd**

group file, making consistent with registry. *See* **passwd_refresh** command and **import_passwd** command

groups
- adding, B-7
  - *See also* **edrgy** command, use to add groups
- changing, B-11 to B-12
  - *See also* **edrgy** command, use to change groups
- deleting, B-13
  - *See also* **edrgy** command, use to delete groups
- displaying, B-15 to B-16
- and passwords, 2-11
- and project lists, 2-11
- summary of information used to create, B-2

H

hardware requirements for registry servers, 7-2

help
- **edrgy** command, C-9
- **rgy_admin** command, C-12

home directory, 2-14, B-3
- overriding on a specific machine, 4-6 to 4-7

I

-i option, to **import_passwd** command, A-3

import_passwd command, 1-10
- changes made in registry, A-2
- check mode. *See* -c option
- conflict summary, A-5
- conflicts resolved, A-3, C-17, C-18
- data entry conventions, A-6
- general, 7-6, A-1
- options, C-19
- processing, A-2
- processing requirements, A-7
- registries entries created by, A-2 to A-3, C-18
- sample invocation, A-9 to A-10
- sample session, A-7 to A-14
- syntax, A-5
- types of conflicts found, A-1

import_passwd command reference, C-17 to C-18

info, **rgy_admin** command, C-12

initrep, **rgy_admin** command option, C-12

L

lb_admin, 6-1

llbd, 7-3

local location broker daemon. *See* **llbd**

local registry, **edrgy** commands for, C-9

location broker, 7-1

locksmith
- and overrides, 3-3
- and ownership rights, 2-3

login, prohibiting on a specific machine, 4-6

log-in shell, 2-14, B-3
- overriding for a specific machine, 4-6 to 4-7

lrep command
- use to verify master registry site, 5-2
- use to verify slave server deletion, 5-3

lrep, **rgy_admin** command option, C-12

lrep -state, **rgy_admin** command, 7-6

M

master registry. *See* master server

master server
- changing site of, 5-1 to 5-2
- defined, 1-7
- restoring to new site, 6-4 to 6-6
- restoring to original site, 6-4
- steps to start, 7-4
- turning into a slave, 6-3

master server site, changing
    general, 5–1
    procedure, 5–2
    procedure requirements, 5–1

member, edrgy command option, C–5

members
    adding and deleting from groups. *See* **edrgy** command, use to add and delete members from groups
    adding members to groups, B–8
    deleting from groups, B–13
    effects of creating accounts, 2–17
    effects of deleting from membership lists, 2–16

membership lists, 2–16
    default members, 2–16
    See also members

memory required by registry servers, 7–2

miscellaneous information in accounts, 2–14, B–3

monitor, rgy_admin command option, C–12

mount point
    establishing name for, 7–7
    See also **pname_resolve** file

N

NCS. *See* Network Computing System

Network Computing System
    and registry configurations, 7–3
    and the registry, 1–1
    servers required for the registry, 7–3

name conflicts, A–1

names
    alias, 2–2
    fullname, 2–2
    in the registry, 2–1
    primary, 2–2
    reserved, 2–7

newgrp command, 2–11

O

omit, in password overrides, C–22

organization file, making consistent with registry. *See* **passwd_refresh** command

organizations
    adding, B–8
        See also **edrgy** command, use to add organizations
    deleting, B–13
        See also **edrgy** command, use to delete organizations
    displaying, B–15 to B–16
    setting policy for. *See* **edrgy** command, use to set organization policy
    summary of information used to create, B–2

overrides, 1–10
    and abbreviations, C–22
    allowing, 3–3
    defined, 4–2
    entries, 4–4 to 4–5
    examples, C–23
    format, 4–3 to 4–4, C–20
    home directory on a specific machine, 4–6 to 4–7
    and local machines, 4–2 to 4–3
    login shell on a specific machine, 4–6 to 4–7
    multiple entries in, 4–7 to 4–8
    omit password override, 4–7, C–22
    override precedence, C–22
    overriding passwords on specific machines, 4–6
    policy. *See* policies
    prohibiting login on a specific machine, 4–6
    reference, C–20 to C–23
    setting policy. *See* **edrgy** command, use to set override policy
    types, 4–2
    and wildcards, 4–5 to 4–6, C–21 to C–22

owners
    defaults assigned by edrgy, 2–7
    defined, 2–3
    of a group, 2–4
    of domains, 2–4
    and little administrative control, 2–4
    and moderate administrative control, 2–5
    of an organization, 2–4
    of a person, 2–4
    of the registry, 2–4
    and tight administrative control, 2–5

P

passwd command, and overrides, 4–8

passwd_override file
- entries, 4–4 to 4–5
- format, 4–3 to 4–4
- multiple entries in, 4–7 to 4–8
- wildcard entries in, 4–5 to 4–6

passwd_refresh command, 1–10
- and **cron** command, 7–2
- format, 4–10 to 4–11
- function, 4–10
- and the omit override, 4–7

passwd_refresh command reference, C–25

password, 2–14, B–3
- for groups, 2–11
- overriding on a specific machine, 4–6
- setting, 7–5

password file
- ensuring consistency with registry. *See* **import_passwd** command
- making consistent with registry. *See* **passwd_refresh** command
- omitting users from, 4–7

password–valid flag, 2–14, B–3

percent character. *See* %

persons
- adding, B–9
  - *See also* **edrgy** command, use to add persons; names
- changing, B–11 to B–12
  - *See also* **edrgy** command, use to change persons
- deleting, B–14
  - *See also* **edrgy** command, use to delete persons
- displaying, B–16 to B–17
- naming. *See* names
- summary of information used to create, B–2

pgo. *See* subject identifier

pgo triplet. *See* subject identifier

pname_resolve
- file, 1–10, 7–2, 7–7
- reference, C–26

policies
- defined, 3–1
- displaying, B–17 to B–18
  - *See also* **edrgy** command, use to display policies
- effects of changing, 3–5
- override, 4–3
- registry vs. organization, 3–4 to 3–8
- setting, 7–5
  - *See also* **edrgy** command, use to set policies
- summarized, B–4

primary names. *See* names

project lists
- and access rights, 2–11
- and PROJLIST environment variable, 2–11
- and prohibiting inclusion on, 2–11
- changing. *See* **edrgy** command, use to change groups
- defined, 2–11

prop, edrgy command, 7–5

propagation. *See* database updates

properties
- defined, 3–1
- displaying. *See* **edrgy** command, use to display properties
- **edrgy** command, C–8
- setting, 7–5
  - *See also* **edrgy** command, use to set properties
- summarized, B–5

Q

quit
- **edrgy** command, C–9
- **rgy_admin** command, C–12

R

registries
- adding accounts, 7–5
- adding names and accounts, 7–5
- configurations, 7–2 to 7–3
- defined, 1–1
- deleting, 6–5
- ensuring consistency with UNIX password and group files. *See* **import_passwd** command
- entries created by **import_passwd**, A–2 to A–3
- entries modified by **import_passwd**, A–2
- local, 1–9
  - setting up, 1–9

merging, general, 5–3 to 5–4
See also **rgy_merge** command
restoring master to original site, 6–4
restoring to a new site, 6–4
steps to set up, 7–1 to 7–2
turning master into a slave, 6–3
turning slave into a master, 6–2 to 6–3
See also database

registry
benefits, 1–2
defined, 1–1
and software releases, 7–2

registry client. *See* client software

registry daemon, 7–3
recreating, 6–2 to 6–3

registry policies. *See* policies

registry properties. *See* properties

registry replicas, creating slave registry replicas, 7–5 to 7–6

registry server
restarting, 7–7
See also server software

registry servers, hardware requirements, 7–2

replica list
defined, 1–8
updating, 5–8

reprep, rgy_admin command, C–13

reset, rgy_admin command option, C–13

rgy_admin command, 1–10
lrep –state, 6–2
use to backup registry database, 4–9 to 4–11
use to delete registry replica, 6–5 to 6–6
use to delete slave server, 5–3
use to reverse master and server sites, 5–2
use to turn master into a slave, 6–3
use to turn slave to a master, 6–2 to 6–3
use to verify servers are running, 7–6

rgy_admin command reference, C–11 to C–13

rgy_create, command reference, C–15

rgy_create command, 7–4

rgy_merge, command reference, C–16

rgy_merge command, 1–10
compare mode, 5–5 to 5–6
conflicts, 5–5
general, 5–4
source, 5–6
syntax, 5–5
target, 5–6
using, 5–6 to 5–8

rgyd, 7–3
recreating. *See* registry daemon
restarting, 7–7

root, and overrides, 3–3

root.%.%, and ownership rights, 2–3

S

SID. *See* subject identifier

server software, 1–2
changing master server site, 5–1 to 5–2
deleting, 5–3
recreating, 6–1 to 6–2
recreating a replica, 6–1 to 6–2

servers, restarting, 7–7

set, rgy_admin command option, C–13

site
edrgy command, C–8
rgy_admin command, C–13

slave registry. *See* slave server

slave server
creating, 7–5
defined, 1–7
turning into a master, 6–2 to 6–3

state, rgy_admin command option, C–13

stop, rgy_admin command option, C–13

subject identifier, 2–14, B–3
defined, 2–1

synch, edrgy command option, C–8

syncids command, 2–10, 5–7 to 5–8, 7–6, A–7, B–12

U

UID. *See* unique identifier

UNIX ID conflicts, A–1

UNIX IDs, synchronizing with registry, 5–7 to 5–8, A–7

UNIX numbers, 2–5 to 2–6
    making consistent, 7–6

unique identifier, 2–5
    recreating association with registry object
        name, 6–5 to 6–6

V

view
    account, C–6
    edrgy command, B–14 to B–18
    persons, groups, and organizations, C–2

Reader's Response

Please take a few minutes to give us the information we need to revise and improve our manuals from your point of view.

Document Title: *Administering the Domain/OS Registry*
Mfg. No.: 015363-A00

User Profile

Your Name ______________________ Title ______________________

Company ______________________

Address ______________________

Telephone number (____) ______________

When you use the Apollo system, what **job(s)** do you perform?

☐ Programming ☐ Application End User ☐ Hardware Engineering
☐ System Administration ☐ Other (describe) ______________________

Characterize your level of **experience** in using the Apollo system:

☐ Experienced user (2+ yrs.) ☐ New user (6 mos. or less)
☐ Moderately experienced user (6 mos.–2 yrs.)

What programming **languages** do you use with the Apollo system?

____

Distribution

How do you know what manuals are **available** to support the products you're using or want to use?

____

What is a major concern for you in **ordering** books?

____

How would you **evaluate** this book?

	Excellent		**Average**		**Poor**
Completeness	1	2	3	4	5
Accuracy	1	2	3	4	5
Usability	1	2	3	4	5

Additional Comments: ______________________

____

____

____

No postage necessary if mailed in the U.S.

cut or fold along dotted line

fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

Apollo Systems Division
A Subsidiary of Hewlett–Packard Company
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

fold

HEWLETT
PACKARD

Order No. 015363-A00

015363-A00