Trusted Solaris Administration Overview

Sun Microsystems Federal, Inc.
A Sun Microsystems, Inc. Business
901 San Antonio Road
Palo Alto, CA 94303
U.S.A.

Part No: 805-8024-10
Revision A, August 1998

DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Please
Recycle

Adobe PostScript

Contents

Figures

Tables

Preface

The *Trusted Solaris Administration Overview* is an introduction to administering the Trusted Solaris™ environment. As prerequisites, you should be familiar with basic system administration in the UNIX environment, understand security policy concepts, and should read the *Trusted Solaris User's Guide*.

Related Materials

The Trusted Solaris documentation set is supplemental to the Solaris 2.5.1 documentation set. You should obtain a copy of both sets for a complete understanding of Trusted Solaris. The Trusted Solaris documentation set consists of:

- *Trusted Solaris Documentation Roadmap* shows all volumes in the documentation set.
- *Trusted Solaris 2.5.1 Release Notes* presents information regarding the hardware requirements for installing Trusted Solaris, features included in the release, any known problems, and interoperability with previous versions.
- *Trusted Solaris Installation and Configuration* describes the process of planning for, installing, and configuring a new or upgraded Trusted Solaris system.
- *Trusted Solaris Global Index* provides an index with entries covering the entire Trusted Solaris documentation set.

- *Trusted Solaris User's Guide* describes basic features of the Trusted Solaris environment from the end user's point of view.

Note – *Trusted Solaris User's Guide* contains a glossary that applies to the entire documentation set.

- *Trusted Solaris Administrator's Procedures* provides detailed information for performing specific administration tasks.
- *Trusted Solaris Audit Administration* describes the auditing system for system administrators.
- *Trusted Solaris Label Administration* provides information on specifying label components in the label encodings file.
- *Trusted Solaris Reference Manual* is a printed version of the man pages available in the Trusted Solaris environment.
- *Compartmented Mode Workstation Labeling: Encodings Format* describes the syntax used in the label encodings file for enforcing the various rules concerning well-formed labels for a system.
- *Trusted Solaris 2.5.1 Transition Guide* provides an overview of the differences between Trusted Solaris 1.x and Trusted Solaris 2.5.

How This Guide is Organized

Chapter 1, "Introduction to Administration," provides an overview of basic concepts needed to administer Trusted Solaris.

Chapter 2, "Controlling File Access: An Example," provides an example that demonstrates how Trusted Solaris mechanisms control access to files.

Chapter 3, "Quick Tour of the Admin Tools" presents an overview of the tools available in the Trusted Solaris environment, how they are accessed, and the databases on which they operate.

Chapter 4, "Administering Users," describes how to administer users and roles in the Trusted Solaris environment.

Chapter 5, "Administering Trusted Networking," provides an overview of how networking is implemented in the Trusted Solaris environment and discusses the tools for administering networking.

Chapter 6, “Administering Auditing,” describes the basics of performing auditing in the Trusted Solaris environment.

Chapter 7, “Other Trusted Solaris Utilities,” introduces tools for administering labels, file systems, devices, execution profiles, and other elements in the Trusted Solaris environment.

Typographic Changes and Symbols

The following table describes the type changes and symbols used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	**Meaning**	**Example**
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `system% You have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`system%` **`su`**
		`Password::`

AaBbCc123	Command-line placeholder or variable name. Replace with a real name or value	To delete a file, type `rm` *filename*. The *errno* variable is set.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User’s Guide*. These are called *class* options. You *must* be root to do this.
Code samples are in `code font` and may display the following:		
`%`	UNIX C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	`system$`
`#`	Superuser prompt, all shells	`system#`

Introduction to Administration 1

This chapter introduces you to system administration in Trusted Solaris. It begins with a quick review of Trusted Solaris concepts from the *Trusted Solaris User's Guide* and goes on to explain some advanced concepts necessary for Trusted Solaris administrators.

≡ 1

Basic Concepts Review

Trusted Solaris is an enhanced version of Solaris that incorporates configurable security policy into the system. The concepts in this section are basic to understanding the Trusted Solaris environment, both for users and administrators. They are briefly covered here and are discussed in more depth in the *Trusted Solaris User's Guide*.

How Trusted Solaris Protects Against Intruders

Trusted Solaris protects access to the system by providing accounts requiring user names with passwords. Passwords can be created by users or system-generated, according to site policy. You can also require that passwords be changed regularly. In addition, users must enter sensitivity information at login that determines which (if any) information they are allowed to access.

Trusted Solaris provides an unmistakable, tamper-proof emblem that appears at the bottom of the screen indicating to users when they are using security-related parts of the system.

As administrator, you should make it a policy never to send instructions via email for users to take actions without personally verifying these instructions. You should tell users never to follow instructions without verification. The purpose of this policy is to avoid such situations as imposters posing as administrators and sending email to users to try to get passwords to accounts or other sensitive information.

How Trusted Solaris Enforces Access Control Policy

Trusted Solaris protects information and other resources through *discretionary access control*—the traditional UNIX permission bits and access control lists set at the discretion of the owner—and *mandatory access control*—a mechanism enforced by the system automatically that controls all transactions by checking the sensitivity labels of processes and files in the transaction.

Sensitivity labels (SLs) represent the sensitivity level at which the user is permitted to and chooses to operate. They determine which information the user is allowed to access. Both mandatory and discretionary access controls can be overridden by special permissions called *privileges*, which are granted to programs. In some cases, users may need *authorizations* as well, which are granted to users (and roles) by the administrator.

As administrator, you need to train users on proper procedures for securing their files and directories, according to your site's security policy. Furthermore, you should instruct any users allowed to upgrade or downgrade labels on when it is appropriate to use these privileges.

How Trusted Solaris Indicates Information Sensitivity

Trusted Solaris provides an option for *information labels (ILs*). Information labels advise users of the sensitivity and handling of data, processes, and devices. In contrast to sensitivity labels, information labels are advisory and not related to access control. They notify users of the security levels for processes and files. Trusted Solaris ensures that whenever a transaction takes place involving data or processes with different information labels that any resulting data files have an appropriate information label.

How Trusted Solaris Implements Administration

Trusted Solaris divides up the system administration responsibilities to help ensure that no single user can compromise the system's security. By default, Trusted Solaris provides four predefined roles for performing administration tasks:

- **Security administrator (secadmin)** – responsible for security tasks and decisions, such as setting up and assigning sensitivity labels and auditing user activity. The security administrator assigns the security-related aspects of all user and role accounts (except for the security administrator's own account). The security administrator also evaluates and installs new software that can impact security and assigns needed privileges to the new software.
- **System administrator (admin)** – performs standard UNIX system administration tasks such as setting up the non-security-relevant portions of user accounts.
- **System operator (oper)** – does system backups, performs printer administration, and mounts removable media.
- **Root** – used primarily for installing commercial software when a real UID of 0 is required. The root role in Trusted Solaris is more limited than the traditional root user in other UNIX systems.

If your site reconfigures the predefined administrative roles, make sure all users know who is performing each set of duties.

Understanding Labels

Sensitivity labels (SLs) and clearances are the heart of mandatory access control in Trusted Solaris. They determine which users can access which files and directories. Information labels (ILs) help users keep track of the sensitivity of the information contained in documents. Note that information labels are an optional feature.

Note – Sensitivity labels and information labels are packaged together in structures called CMW labels, which are primarily of importance to programmers. For general purposes, you can think of sensitivity labels and information labels as separate entities.

This section discusses relationships between labels, the `label_encodings` file (which is the source of all labels for a system), and the various factors that determine which labels are available to a user.

Dominance Relationships Between Labels

Trusted Solaris mediates all attempted security-related transactions. It first compares the sensitivity labels of the accessing entity and the entity being accessed, and then permits or disallows the transaction depending on which label is *dominant* (as described below). Secondly, Trusted Solaris compares the information labels (if implemented) of the two entities and floats (raises) the information label of the resulting document, if necessary. (See “Monitoring Information Transactions” in Chapter 1, “Introduction to Trusted Solaris,” in the *Trusted Solaris User’s Guide*.)

One entity’s label (sensitivity or information) is said to *dominate* another’s if the following two conditions are met:

- The classification component of the first entity’s sensitivity label is equal to or higher than the second entity’s classification. (The security administrator assigns numbers to classifications in the `label_encodings` file; these numbers are compared when determining dominance.)

- The set of compartments (and markings if information labels) in the first entity includes all of the second entity's compartments (and markings).

Two labels are said to be *equal* if they have the same classification and the same set of compartments (and markings if information labels). If they are equal, they dominate each other so that access is permitted. If one label has a higher classification or includes all of the second label's compartments or both, the first label is said to *strictly dominate* the second label. Two labels are said to be *disjoint* or *noncomparable* if neither label dominates the other.

Table 1-1 presents examples of label comparisons for dominance.

Table 1-1 Examples of Label Relationships

Label 1	**Relationship**	**Label 2**
Top Secret A B	(strictly) dominates	Secret A
Top Secret A B	(strictly) dominates	Secret A B
Top Secret A B Eyes-only	(strictly) dominates	Secret A B Eyes-only
Top Secret A B	(strictly) dominates	Top Secret A
Top Secret A B	dominates (equals)	Top Secret A B
Top Secret A B	is disjoint with	Top Secret C
Top Secret A B	is disjoint with	Secret C
Top Secret A B	is disjoint with	Secret A B C

Label Encodings Files

All label components for a system, that is, classifications, compartments, markings, and the associated rules are stored in a file called `label_encodings` (located in `/etc/security/tsol/`). The security administrator sets up the `label_encodings` file for the site. A label encodings file contains

- **component definitions** – definitions of classifications, sensitivity labels, clearances, and information labels, including rules for required combinations and constraints
- **accreditation range definitions** – definitions of the range boundaries for the entire system and for normal (non-administrative) users

- **printing specifications** – identification and handling information for print banners, trailers, headings, footers, and other security features for printouts
- **customizations** – local definitions including label color codes, alternative names for classifications, compartments, and markings in the graphical interface, and other items

For more information on the `label_encodings` file, see the man page for `label_encodings(4TSOL)` and the manuals, *Trusted Solaris Label Administration* and *Compartmented Mode Workstation Labeling: Encodings Format*.

Label Ranges

Since there are multiple labels in a system, it is useful to think in terms of ranges of labels, defined by a minimum, maximum, and other constraints. A *label range* is the set of potentially usable sensitivity labels at which a user or a class of users can operate. A range is not quite as simple as all combinations of labels that fall between a maximum and minimum label. There may be rules in the label encodings file that disqualify certain combinations. A label must be *well-formed*, that is, permitted by all applicable rules in the label encodings file, in order to be included in a range. On the other hand, a clearance does not have to be well-formed. Suppose, for example, that a label encodings file prohibits any combination of compartments A, B, and C in a sensitivity label. TS A B C would be a valid clearance but not a valid sensitivity label; as a clearance, it would let a user access files labeled TS A, TS B, and TS C.

Administration Labels

Trusted Solaris provides two special administration labels (used as sensitivity labels, information labels, and clearances): ADMIN_HIGH and ADMIN_LOW. (You can rename these two labels in the `label_encodings` file if you choose.) These labels are intended for administrators rather than normal users.

ADMIN_HIGH is the highest possible label in the system and is used to protect system data, such as administration databases or audit trails, from being read. You need to work at the ADMIN_HIGH label (typically in an administrative role) or have the privilege to read up from your current sensitivity label to read data labeled ADMIN_HIGH.

ADMIN_LOW is the lowest sensitivity label in a system. Mandatory access control does not permit users to write data to files with sensitivity labels lower than the subject’s sensitivity label. Thus, applying ADMIN_LOW, the lowest sensitivity label, to a file ensures that normal users cannot write to it although they can read it. ADMIN_LOW is typically used to protect public executables and configuration files to prevent them from being modified, since only a user working at ADMIN_LOW or with the privilege to write down would be able to write to these files.

Accreditation Ranges

An accreditation range is a label range for a class of user. Accreditation ranges are approved by the security administrator as part of an organization’s security policy. There are two accreditation ranges defined in the `label_encodings` file:

- system accreditation range
- user accreditation range

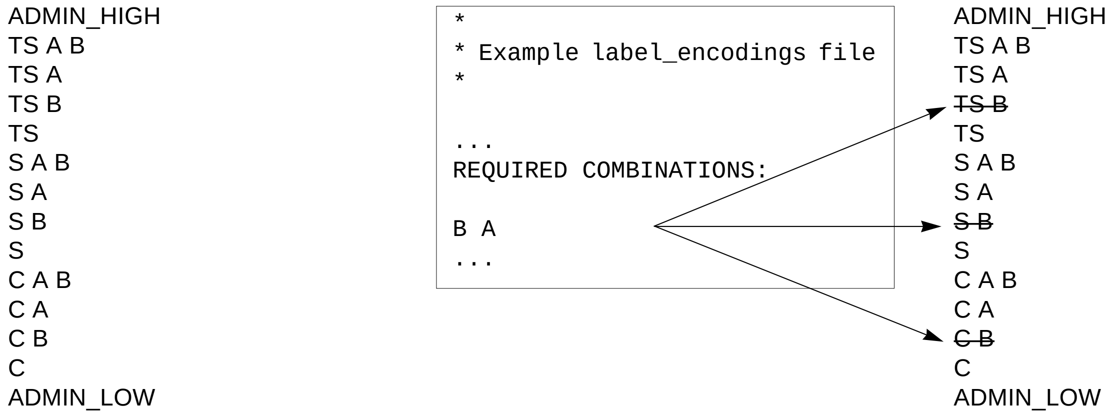
System Accreditation Range

The *system accreditation range* is the complete set of potentially usable sensitivity labels and information labels intended for administrators. The system accreditation range includes ADMIN_HIGH and ADMIN_LOW; it is constrained by the rules in the `label_encodings` file. The rules for the system accreditation range are used to disqualify label combinations that will never be permitted on the system.

Figure 1-1 presents an example of how rules constrain the labels permitted in a system accreditation range.

Figure 1-1 (a) shows all possible combinations given the classifications, TS (TOP SECRET), S (SECRET), and C (CONFIDENTIAL), and the compartments, A and B.

Figure 1-1 (b) shows a typical rule in the REQUIRED COMBINATIONS section of the `label_encodings` file and its effects. The arrows point to the labels disqualified by the rule, which appear with lines through them. The syntax B A means that any label that has B as a compartment must also contain A. (Note that the converse is not true; compartment A is not required to be combined with any other compartments.) Since compartment B is only permitted in combination with A, the labels TS B, S B, and C B are not well-formed and hence not in the system accreditation range.



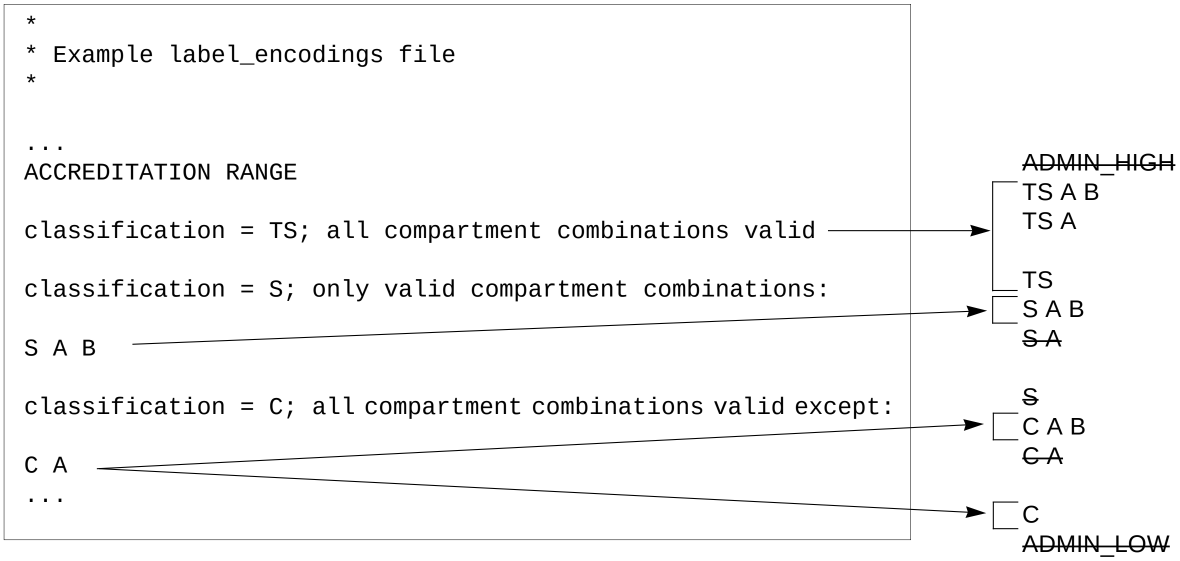
(a) Set of Potential Combinations

(b) Rule in label_encodings File and its Effect on the System Accreditation Range

Figure 1-1 How System Accreditation Range Is Constrained By Rules

User Accreditation Range

The *user accreditation range* is the largest set of labels (within the system accreditation range) that a single user could potentially access. It is a subset of the system accreditation range; it excludes ADMIN_HIGH and ADMIN_LOW and is further constrained by a set of rules located in the ACCREDITATION RANGE portion of the `label_encodings` file. The rules for the user accreditation range disqualify label combinations that are permitted for administrators only. The user accreditation range in Figure 1-2 continues the example showing three different types of rules in the ACCREDITATION RANGE section and their effect on the user accreditation range. The arrows point to the well-formed labels permitted by the particular rule.



```
*
* Example label_encodings file
*

...
ACCREDITATION RANGE

classification = TS; all compartment combinations valid

classification = S; only valid compartment combinations:

S A B

classification = C; all compartment combinations valid except:

C A
...
```

Figure 1-2 ACCREDITATION RANGE Portion of label_encodings File

As shown in Figure 1-2, the user accreditation range excludes ADMIN_HIGH and ADMIN_LOW. It includes all TS combinations except TS B, which is overruled by the REQUIRED COMBINATIONS rule B A mentioned earlier (for the same reason, S B and C B are not permitted). S A B is the only valid combination for the S classification. All C combinations except C A are valid (remember that C B was overruled earlier).

Other label_encodings File Constraints

The `label_encodings` file imposes additional constraints on the labels available to users. The *minimum clearance* defines the lowest default clearance that the administrator can assign to any user. The (accreditation) *minimum sensitivity label* defines the lowest well-formed sensitivity label that the administrator can assign to any user for operation in a Trusted Solaris session (this minimum sensitivity label is applied on a system-wide basis; do not confuse it with the minimum sensitivity label assigned to individual accounts). Typically but not always, the minimum clearance and minimum sensitivity label are set to the same value. The `label_encodings` file also contains rules regarding other required label combinations and constraints.

Account Label Range

The *account label range* is the effective range of sensitivity labels available to an individual user or role. It governs which label selections are available to the user in the Session Sensitivity Label and Clearance dialog boxes when the user first logs in (see “Setting the Session Level” in Chapter 2, “Accessing and Leaving the Trusted Solaris Environment,” in the *Trusted Solaris User’s Guide*). The labels available in the account label range are a function of

- the definition of the user accreditation range – a user cannot use sensitivity labels disqualified for the user accreditation range
- the (accreditation) minimum sensitivity label from the `label_encodings` file – defines an absolute minimum on sensitivity labels that can be assigned to users
- the user clearance from the `tsoluser` database – defines the top of the account label range.

Note – The `tsoluser` database contains security attributes for users and roles and is edited by the administrator using the User Manager.

- the (account) minimum sensitivity label from the `tsoluser` database – sets the bottom of the account range, unless it is overridden by the (accreditation) minimum sensitivity label from the `label_encodings` file.

The account label range is a subset of the user accreditation range and is also constrained by the minimum sensitivity label from the `label_encodings` file. An example account label range is shown in Figure 1-3 based on the accreditation examples from the previous sections.

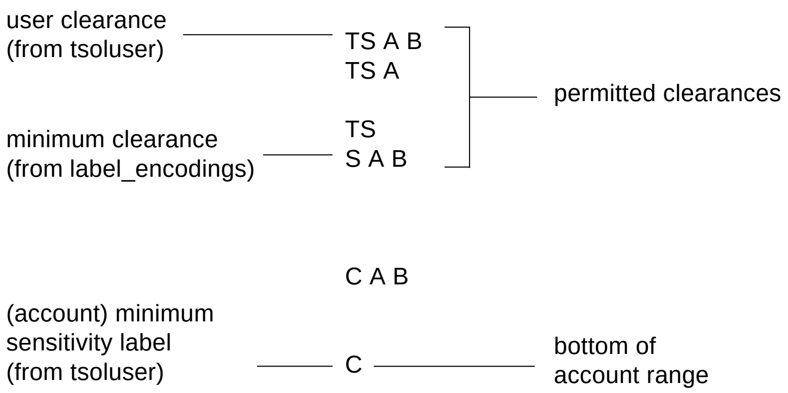


Figure 1-3 Constraints on an Account Label Range

The user in this example has an account range bounded by TS A B, the user clearance, at the top and C, the (account) minimum sensitivity label, at the bottom. As a result of these definitions, the user is constrained to logging in at TS A B, TS A, TS, or S A B. The user's (account) minimum sensitivity label is C, which happens to coincide with the minimum sensitivity label from the label_encodings file; if these two minimums were different, the higher of the two would set the bottom of the account range.

Note – If you set the user's clearance to be the same as the user's (account) minimum sensitivity label, you are effectively forcing the user into single-label sessions at this sensitivity label.

Session Range

The *session range* is the set of sensitivity labels available to a user during a Trusted Solaris session. It is a function of

- the user's account label range
- the user's choice of session mode (single- or multilabel)
- the value the user enters in the Session Sensitivity Label dialog box (if single-label session) or the Clearance dialog box (if multilabel session)
- the label range for the user's workstation

The choice of session clearances appearing in the Clearance dialog box range from the user clearance down to the higher of the (accreditation) minimum clearance and the (account) minimum sensitivity label, subject to any additional required combinations or constraints from the clearance rule definitions in the `label_encodings` file. If the user selects a single-label session, the user has the same range of labels to select from, subject to any required combinations or constraints from the sensitivity rule definitions in the `label_encodings` file.

Note – It is also possible to impose a range on a login device. This is done by specifying a maximum and minimum sensitivity label in the `device_allocate` file. For more information, see "How Trusted Solaris Controls Device Access" on page 34.

In the example, the user can specify a session clearance using any well-formed label in the Figure 1-3 between S A B and TS A B. If the user's clearance does not dominate the minimum clearance, the user cannot log in. If the user's (account) minimum sensitivity label is less than the (accreditation) minimum sensitivity label, then the (accreditation) minimum sensitivity label defines the bottom of the session range.

Figure 1-4 (a) continues the example showing the range of sensitivity labels available if the user selects a multilabel session with a session clearance of S A B. Since the other potential labels between S A B and C have been disallowed, effectively the user can only work at S A B, C A B, or C.

Figure 1-4 (b) shows the range of labels if the user chooses a single-label session with a session sensitivity label of C A B. Note that C A B is below the minimum clearance but is accessible because the user is selecting a session sensitivity label, not a clearance. Since this is a single-label session, the user can work at only one label; in this example, the user specified C A B, although S A B or C could have been chosen instead.

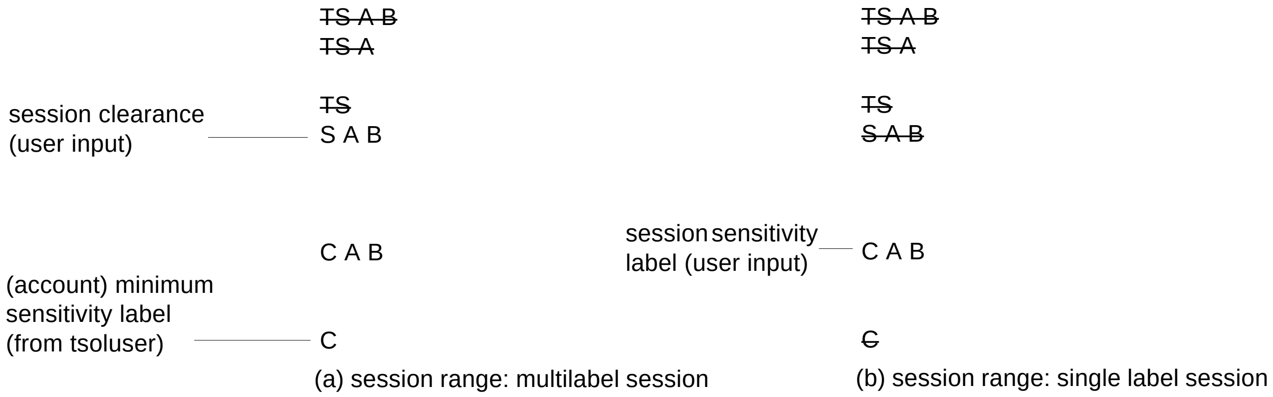

~~TS A B~~
~~TS A~~
~~TS~~
S A B
session clearance (user input)
C A B
(account) minimum sensitivity label (from tsoluser)
C
(a) session range: multilabel session
~~TS A B~~
~~TS A~~
~~TS~~
~~S A B~~
session sensitivity label (user input)
C A B
~~C~~
(b) session range: single label session

Figure 1-4 Comparison of Session Ranges

Figure 1-5 summarizes the progressive eliminations of available sensitivity labels in this example. The eliminated sensitivity labels are shown with a line through them in the range where they are filtered out and are not shown in subsequent ranges.

(a) Set of Potential Combinations	(b) System Accreditation Range	(c) User Accreditation Range	(d) Account Label Range	(e) Multilabel Session Range Using S A B
ADMIN_HIGH	ADMIN_HIGH	~~ADMIN_HIGH~~		
TS A B	TS A B	TS A B	TS A B	~~TS A B~~
TS A	TS A	TS A	TS A	~~TS A~~
TS B	~~TS B~~			
TS	TS	TS	TS	~~TS~~
S A B	S A B	S A B	S A B	S A B
S A	S A	~~S A~~		
S B	~~S B~~			
S	S	~~S~~		
C A B	C A B	C A B	C A B	C A B
C A	C A	~~C A~~		
C B	~~C B~~			
C	C	C	C	C
ADMIN_LOW	ADMIN_LOW	~~ADMIN_LOW~~		

Figure 1-5 Cumulative Effect of Constraints on a Session Range

Label Availability in Trusted Solaris Sessions

Table 1-2 shows session label limitations and availability based on users' session choices; it continues the example. The left column identifies the types of label settings used in sessions. The middle two columns apply to multilevel sessions and the right two columns apply to single-level sessions. The columns labeled General Case show how the label types are determined. The columns marked Example show a typical user's session selections at login.

Table 1-2 Labels in Trusted Solaris Sessions

	Multilevel Session		**Single-level Session**	
	General Case	**Example #1: Multilevel with clearance of [SECRET A B]**	**General Case**	**Example#2: Single-level with session sensitivity label of [SECRET A B]**
Initial workspace SL	Lowest sensitivity label in account label range.	[CONFIDENTIAL]	Session sensitivity label specified by user	[SECRET A B]
Available workspace SLs	Any sensitivity label in account label range up to the session clearance	[CONFIDENTIAL] [CONFIDENTIAL A B] [SECRET A B]	Session sensitivity label specified by user	[SECRET A B]
Initial Input IL	Lowest sensitivity label in user accreditation range.	UNCLASSIFIED	Lowest sensitivity label in user accreditation range.	UNCLASSIFIED
Maximum information label (Input IL and floating maximum)	Highest permitted sensitivity label in current workspace with markings.	SECRET A B <markings> (in [S A B] workspace)	Highest permitted sensitivity label in current workspace with markings.	SECRET A B <markings>

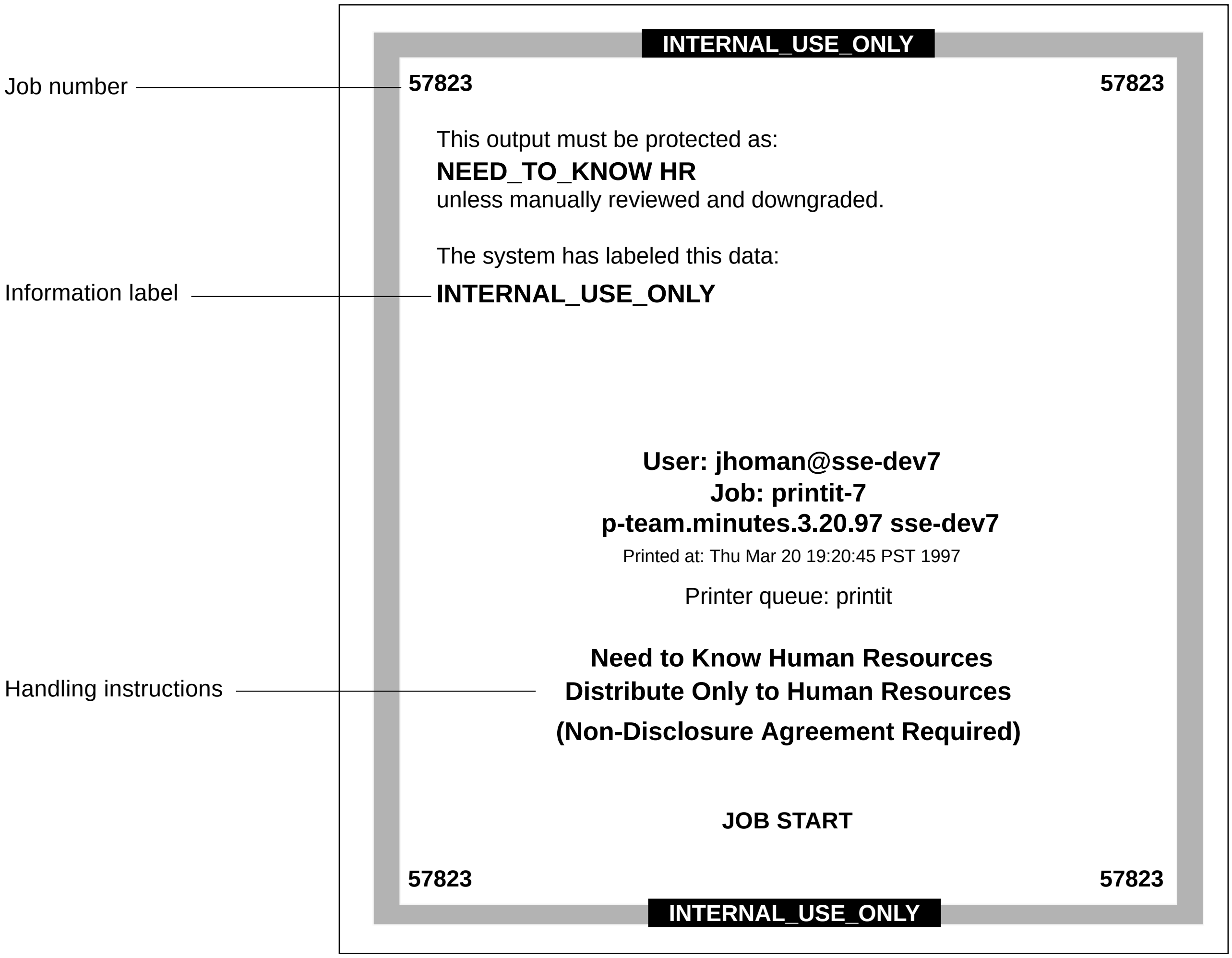
In Example #1, the initial workspace is set [CONFIDENTIAL], the sensitivity label at the bottom of the user's account label range. The user can work at a sensitivity label of [CONFIDENTIAL], [CONFIDENTIAL A B], or [SECRET A B] (users switch sensitivity labels by changing the sensitivity label of a workspace and clicking its button).

The user's initial Input IL is set to the minimum sensitivity label in the user accreditation range, with the assumption that any new data entered is considered to be non-sensitive information unless the user makes a conscious effort (by selecting Change Input IL from the Trusted Path menu) to raise the information label of the information. Raising the Input IL or the information label of a document through floating is limited to the sensitivity label of the workspace plus any valid markings.

In Example #2, the user's initial workspace SL is [SECRET A B]. Since this is a single-level session, the only available workspace SL is [SECRET A B]. As in multilevel sessions, the user's initial Input IL is set to the minimum sensitivity label in the user accreditation range and maximum information label is equal to the workspace sensitivity label with markings.

Applying Labels to Printed Output

You can cause sensitivity labels, information labels, and handling information to print out automatically on all printers as well as configure other security features to be printed. Figure 1-6 shows a typical banner page. For more information on configuring printing in Trusted Solaris, see the "Managing Printing" chapter in *Trusted Solaris Administrator's Procedures* and also *Trusted Solaris Label Administration*.



Job number

Information label

Handling instructions

INTERNAL_USE_ONLY

57823 57823

This output must be protected as:
NEED_TO_KNOW HR
unless manually reviewed and downgraded.

The system has labeled this data:
INTERNAL_USE_ONLY

User: jhoman@sse-dev7
Job: printit-7
p-team.minutes.3.20.97 sse-dev7
Printed at: Thu Mar 20 19:20:45 PST 1997

Printer queue: printit

Need to Know Human Resources
Distribute Only to Human Resources
(Non-Disclosure Agreement Required)

JOB START

57823 57823

INTERNAL_USE_ONLY

Figure 1-6 Typical Print Banner Page

Understanding Single- and Multilevel Directories

To help prevent the inadvertent mixing of files with different labels, the Trusted Solaris environment provides two special types of directories: multilevel directories and single-level directories.

Multilevel Directories (MLDs)

Multilevel directories (MLDs) are directories that have the ability to store files and directories with different sensitivity labels transparently. Home directories are typically multilevel directories.

Multilevel directories have a hidden string, ".MLD." (referred to as an *adornment*) appended to the beginning of the directory name. The adornment is not visible using standard UNIX commands; you can view it using special adornment commands (see Table 1-3).

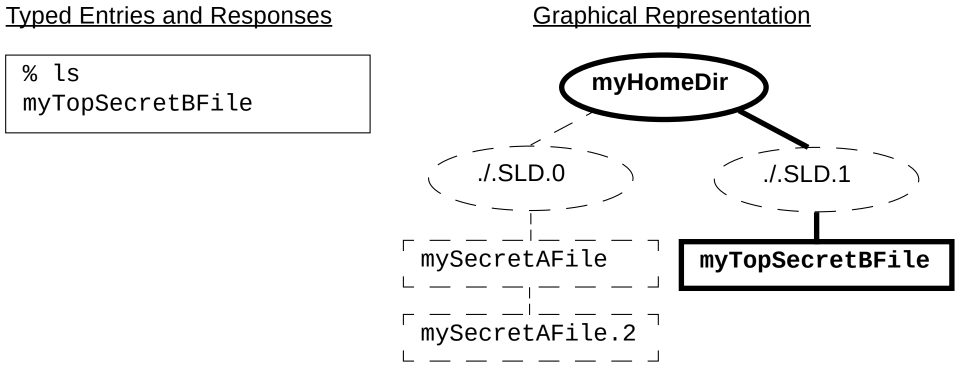
Single-level Directories (SLDs)

Single-level directories (SLDs) are hidden directories that store files and directories having the same sensitivity label only. When you create or move a file or directory into a multilevel directory, the new file or directory is automatically stored in the single-level directory corresponding to its sensitivity label. If a single-level directory corresponding to the sensitivity label does not yet exist, the environment creates one automatically.

The adornment for single-level directories is the string, ".SLD.". The single-level directories are named `.SLD.0`, `SLD.1`, and so on, in order of their creation. They are not normally visible except through the special commands described in Table 1-3 on page 19.

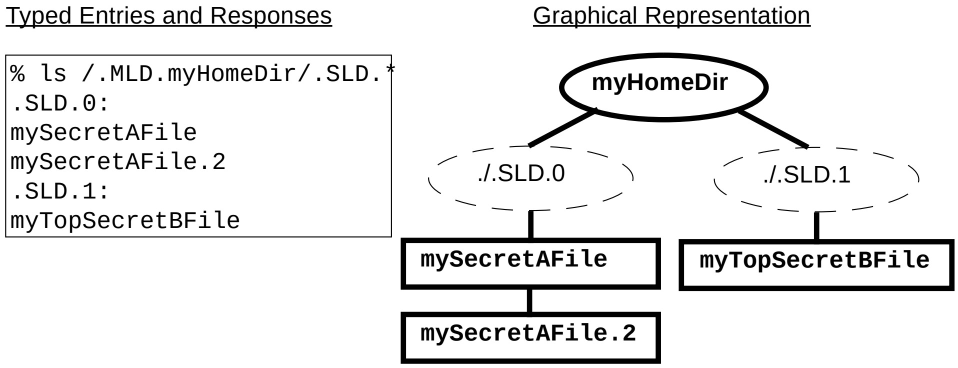
Viewing Contents of Single-level Directories

One can view the contents of a hidden directory by explicitly specifying the adornments to the path. For example, while working at the TOP SECRET A B sensitivity label, the user can type **`ls`** to view the contents of the single-level directory for TS A B files and directories (see Figure 1-7). If the user types **`ls /.MLD.myHomeDir/.SLD.*`** (and has the appropriate privileges), the user sees all hidden directories in the multilevel directory (see Figure 1-8). The left side of these figures show the commands the user enters; the right side illustrates the directory structure, depicting directories as ovals, files as rectangles, visible items with solid lines and bolding, and hidden items with dashed lines and normal font.



```
% ls
myTopSecretBFile
```

Figure 1-7 Normal Viewing of a Directory



```
% ls /.MLD.myHomeDir/.SLD.*
.SLD.0:
mySecretAFile
mySecretAFile.2
.SLD.1:
myTopSecretBFile
```

Figure 1-8 Viewing the Contents of Multiple SLDs

Commands for Working in Single- and Multilevel Directories

The Trusted Solaris environment provides special commands for viewing the adornments on single- and multilevel directories. These commands are described in Table 1-3.

Table 1-3 Adornment Commands

Command Name	Description
`adornfc(1TSOL)`	The adornfc(1TSOL) command lets you display the specified directory pathname with the final component adorned, that is, the strings .MLD. or .SLD. used to identify whether the directory is multilevel or single-level.
`getmldadorn(1TSOL)`	The `getmldadorn`(1TSOL) command lets you display the MLD adornment of the filesystem on which the specified pathname resides.
`getsldname(1TSOL)`	The `getsldname`(1TSOL) command lets you display the single-level directory name associated with the sensitivity label of the current process within the multilevel directory referred to by pathname.
`mldpwd(1TSOL)`	The `mldpwd`(1TSOL) command lets you display the pathname of the current working directory, including any MLD adornments and SLD names.
`mldrealpath(1TSOL)`	The `mldrealpath`(1TSOL) command lets you display the canonicalized absolute pathname, including any MLD adornments and SLD names. It expands all symbolic links and resolves references to special characters (/. and /..) and translations in pathnames. The resulting path has no special characters, unadorned multilevel directories, or any hidden SLD names.

Understanding Trusted Software Administration

In standard UNIX systems, root (superuser) is all-powerful, with the ability to read and write to any file, run all programs, and send kill signals to any process. In the Trusted Solaris environment, root's powers to override system protections are separated into discrete permissions—*authorizations*, which are assigned to users, and *privileges*, which are assigned to applications. Applications that use privileges, authorizations, or effective UIDs/GIDs are called *trusted applications*. Trusted applications, as well as all other applications, are assigned to users and roles through a bundling mechanism called an *execution profile*. Execution profiles can include applications, authorizations, privileges, and effective UIDs / GIDs. To run a particular trusted application requires the right combination of authorizations and privileges.

Figure 1-9 illustrates how users and roles gain access through execution profiles to trusted applications in the Trusted Solaris environment. A user can access profiles either directly or through a role. Profiles have names and include some combination of CDE actions, commands, authorizations, privileges, and effective UIDs / GIDs. These concepts are covered in more depth in the sections that follow.

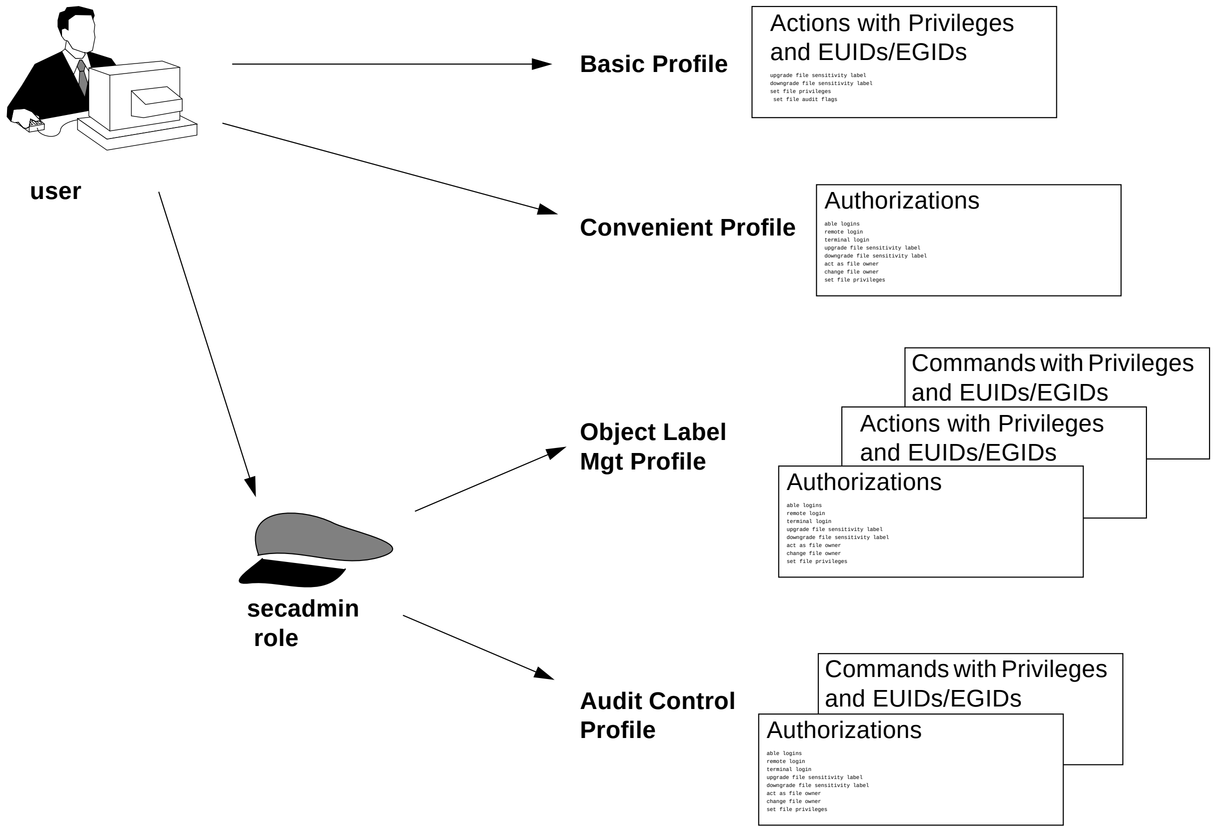


Figure 1-9 How Trusted Applications Are Allocated in Trusted Solaris

Understanding Execution Profiles

An *execution profile* is a bundling mechanism that serves as a building block for assigning trusted programs and capabilities to individual users or roles. An execution profile may contain

- Authorizations
- CDE actions with
  - specified inheritable privileges
  - label ranges defined by maximum and minimum sensitivity labels
  - effective UIDs and GIDs
- Commands with
  - specified inheritable privileges
  - label ranges defined by maximum and minimum sensitivity labels
  - effective UIDs and GIDs

The main purpose of an execution profile is to isolate authorizations and applications that exercise privileges, effective UIDs / GIDs, or both so that only users who need to use them can access them. You edit profiles with a tool called the Profile Manager (see "Using the Profile Manager" on page 120).

Profiles Available in Trusted Solaris

Trusted Solaris provides the execution profiles shown in Table 1-4, which also shows their assignment to the four default roles.

Table 1-4 Execution Profiles with Assignment to Default Roles

Profile Name	**Purpose**	**Security Admin**	**System Admin**	**System Oper**	**Root**
All	Provides access to all executables but without privileges.				Y
All Actions	Provides access to all actions but without privileges.				
All Authorizations	Provides all authorizations. For testing.				
All Commands	Provides access to all commands but without privileges.				
Audit Control	For managing the audit subsystem but without ability to read files.	Y			

Table 1-4 Execution Profiles with Assignment to Default Roles

Profile Name	**Purpose**	**Security Admin**	**System Admin**	**System Oper**	**Root**
Audit Review	For reading the audit trail.		Y		
Basic Actions	Provides access to the applications on the Front Panel with the necessary privileges.	Y	Y	Y	Y
Basic Commands	Provides access to rudimentary commands necessary for all roles.	Y	Y	Y	Y
boot	For starting and shutting down the system.				
Convenient Authorizations	Provides authorizations for normal users.				
cron	Provides root with commands needed for cron jobs.				Y
Cron Management	For managing cron and at jobs.		Y		
Cron Security	For managing cron and at jobs for administrative roles.	Y			
Custom Admin Role	This is an empty profile for adding security attributes to the default Admin role.		Y		
Custom Oper Role	This is an empty profile for adding security attributes to the default Oper role.			Y	
Custom Root Role	This is an empty profile for adding security attributes to the default Root role.				Y
Custom Secadmin Role	This is an empty profile for adding security attributes to the default Secadmin role.	Y			
Device Management	For allocating and deallocating devices, and correcting error conditions.		Y		Y
Device Security	For managing and configuring devices.	Y			
dtwm	For using the window manager.				
Enable Login	Provides the authorization for allowing yourself and other users to log in after boot.		Y		
File System Management	For managing file systems.		Y		Y
File System Security	For managing file system labels and other security attributes.	Y			
inetd	For programs executed by the inetd daemon.				

Table 1-4 Execution Profiles with Assignment to Default Roles

Profile Name	Purpose	Security Admin	System Admin	System Oper	Root
Mail Management	For configuring sendmail, modifying aliases, and checking mail queues.		Y		
Maintenance and Repair	Provides commands needed to maintain or repair a system.		Y		
Media Backup	Backup files.			Y	
Media Restore	Restore files from backup.		Y		
Network Management	For managing the host and network configuration.		Y		Y
Network Security	For managing network and host security, with authorizations for modifying trusted network databases.	Y			Y
NIS+ Administration	Provides access to NIS+ scripts/commands that are not security-related.		Y		
NIS+ Security Administration	Provides access to NIS+ security-related scripts/commands.	Y			Y
Object Access Management	For changing ownership and permissions on files.	Y			
Object Label Management	For changing labels of files and setting up system-wide labels.	Y			
Object Privilege Management	For changing privileges on executable files.	Y			
Outside Accred	Operate outside system accreditation range.	Y	Y	Y	Y
Printer Security	For managing printer devices.	Y			Y
Privileged Shells	For developers to run Bourne, Korn, and C shells with all privileges. NOT intended for secure environments.				
Process Management	For managing current processes, including cron and at jobs.	Y	Y		

Table 1-4 Execution Profiles with Assignment to Default Roles

		Security	**System**	**System**	
Profile Name	**Purpose**	**Admin**	**Admin**	**Oper**	**Root**
Software Installation	For adding application software to the system.		Y		Y
User Management	For creating and modifying users but without the ability to modify self (as a security measure).		Y		Y
User Security	For creating and modifying users' security attributes but without the ability to modify self (as a security measure).	Y			Y

To see the contents of the execution profiles, refer to Appendix A, "Profile Summary Tables" in *Trusted Solaris Administrator's Procedures*.

Complementary Profile Pairs

Notice in Table 1-4 that the following pairs of execution profiles are complementary, that is, they are logically related but are split up in the Trusted Solaris environment for security purposes:

- Audit Control and Audit Review
- Media Backup and Media Restore
- NIS+ Administration and NIS+ Security Administration
- System Management and System Security
- User Management and User Security

Reconfiguring Execution Profiles

As an administrator, you need to know which trusted programs are available, their sensitivity label range, the privileges they need to perform tasks, and which profiles they are in. With this information, you can devise a strategy for assigning profiles to users and roles. For a complete listing of the default profiles and their contents, see `tsolprof(4TSOL)` or Appendix A, "Profiles," in *Trusted Solaris Administrator's Procedures*.

Reconfiguring Roles

If your site does not use the four default roles, you can reassign the profiles to different roles using the Profiles dialog box in the User Manager (see "Specifying Execution Profiles for Users" on page 78 in this manual and Chapter 4, "Managing Roles" in *Trusted Solaris Administrator's Procedures*).

How Users Access Applications in Execution Profiles

Users can access CDE actions in profiles through the Front Panel, the Application Manager, and the File Manager. Users access commands in profiles through a version of the Bourne shell called the *profile shell*, which is modified to limit users and roles to applications in their profiles. You assign a profile shell to a user or role through the User Manager (see "Specifying Execution Profiles for Users" on page 78). A profile shell can be used to *enable* users, that is, give them access to commands, privileges, and authorizations not available to normal users; or to *restrict* users, that is, to limit them to a specific set of commands (this might be appropriate for unsophisticated users). Profile shells are required when you are setting up role accounts or users with profiles containing privileges or authorizations.

Operating as a user or assuming a role gives a user access to those applications and security attributes available through that user's or role's profiles. Note that two profiles may access the same application but with different levels of capability, according to their privileges and authorizations. For example, a user accessing the File Manager from the Basic executable profile has no extra privileges or authorizations. A user accessing the File Manager from the Object Label Management profile can exercise the file_mac_write privilege to override MAC protections when writing to a file or can exercise the file_dac_read privilege to read files without having the basic UNIX permissions.

Understanding Roles

A *role* is a special user account that is generally used to give a user access to certain applications and the authorizations and privileges necessary for running them. All users who can assume the same role have the same role home directory, operate in the same environment, and have access to the same files. Users cannot log in directly to a role; they must log into their user account prior to assuming a role (this requirement ensures that the user's real

UID is recorded for auditing). Each role has its own workspace, which is accessed by a button in the Front Panel. Users are required to authenticate themselves by providing a role password prior to assuming the role.

You may wish to create new roles in addition to the three predefined administrative roles (system operator is actually a non-administrative role). The main reason for creating a role is to define an explicit job responsibility that can use special commands and actions and any necessary privileges, that needs to be isolated from normal users, and that uses a shared home directory, files, and environment. (If you need to isolate commands and privileges with separate home directories and files for different users, then you should create a special execution profile instead of a role. See "Understanding Execution Profiles" on page 21.)

There are two types of roles: administrative and non-administrative. *Administrative roles* are used for security-related tasks. Administrative roles are assigned to sysadmin group 14, are privileged NIS+ principals, and can launch processes containing the *trusted path attribute*, all of which are required for running most administrative applications. *Non-administrative roles* are used for tasks that are not related to security and that can take advantage of shared files and directories. A task with a rotating ownership would be a good application of non-administrative roles.

Understanding Authorizations

An *authorization* is a discrete right granted to a user or role to perform an operation that would otherwise be prohibited by Trusted Solaris. For example, users are not normally allowed to paste information from one window to another window whose sensitivity label strictly dominates the first window's sensitivity label. The *paste to upgraded window* authorization lets a user paste the information in this situation.

Trusted Solaris provides more than 40 authorizations that administrators can assign. The authorizations provided fall into the categories shown in Table 1-5.

Table 1-5 Authorization Categories

Authorization Category	Example Authorizations in the Category
login	*enable logins* – lets user enable logins after a reboot *remote login* – lets user log in remotely using such programs as Telnet or FTP
file control	*upgrade file sensitivity label* – lets user upgrade a file's sensitivity label *set file audit flags* – lets user set audit flags for a file
device control	*allocate device* – lets user allocate a device, its sensitivity label, and its information label
window control	*paste to downgraded window* – lets user paste information to a downgraded window *occupy a different SL's workspace* – lets user move an application window to a workspace with a different sensitivity label
label control	*use all defined labels* – lets user use any label in the system accreditation range
file management	*bypass view of file contents on drag and drop* – lets user view file contents on drag and drop *set application search path* – lets user change the locations for loading applications for CDE executable actions
admin tools	*set user identity* – lets user set user identity information *set user profiles* – lets user set execution profiles

For a complete list of authorizations, see the `auth_desc(4TSOL)` man page. Authorizations are assigned to execution profiles using the Profile Manager, which is described in "Using the Profile Manager" on page 120.

Understanding Privileges

A *privilege* is a right granted to a process to perform an operation that would otherwise be prohibited by Trusted Solaris. For example, processes cannot normally open data files unless they have the proper file permission. In the Trusted Solaris environment, the file_dac_read privilege gives a process the ability to override the UNIX file permissions for reading a file.

Trusted Solaris determines which privileges a process can exercise based on privileges assigned to the application's executable file and privileges associated with the application process or parent process. To make a privilege available to an application, you assign it to two or more of the following sets, depending on how you want it made available to users:

- **Allowed set** – is associated with the application's executable file. An *allowed privilege* is a privilege that can be used with an application provided that other conditions are met (see "How a Process Acquires Privileges" below). The allowed set is the most general factor that determines if a process can exercise a privilege. Excluding a privilege from the allowed set means that no user can ever exercise this privilege with this application. You specify allowed privileges using either the File Manager or the command `setfpriv`. The command `getfpriv` lets you see which privileges are currently in the allowed set.
- **Forced set** – is associated with the application's executable file. A *forced privilege* is a privilege that is enabled unconditionally when the application is executed by any user with access to it. You specify forced privileges using the File Manager or `setfpriv`, in similar fashion to the allowed privileges. Note that a privilege cannot be added to the forced set unless it is also in the allowed set.
- **Inheritable set** – is associated with the application process and is a combination of privileges assigned to the application in its execution profile and privileges inherited from the process's parent. An *inheritable privilege* is a privilege that is enabled when the process is launched (provided that the privilege is also in the application's allowed set). You can assign a privilege directly to the process's inheritable set using the Profile Manager. A process can also acquire inheritable privileges from its parent process. If the parent process launches the child using `exec`, the child's allowed set limits the privileges it can inherit.

Note – Forced privileges are not inheritable by child processes except in applications that have been customized especially for the Trusted Solaris environment to have that specific capability.

How a Process Acquires Privileges

A process must meet the following conditions to be able to exercise a privilege:

- The privilege must be included in the executable file's (or script interpreter's) set of allowed privileges.
- If any user with access to the application should be able to exercise this privilege, then the privilege must be included in the executable file's set of forced privileges.
- If only users or roles with a specific execution profile are to exercise this privilege, then the privilege must be included in the execution profile's set of inheritable privileges or in the inheritable privilege set of a parent process that can launch the application.

Default Privileges Supplied by Trusted Solaris

Trusted Solaris provides more than 80 privileges that you can apply to applications to override security policy. For a complete list of privileges, see the `priv_desc(4TSOL)` man page. The privileges provided fall into the categories shown in Table 1-6.

Table 1-6 Privilege Categories

Privilege Category	**Summary**	**Example Privileges in the Category**
file system security	Overrides file system restrictions for user and group IDs, access permissions, labeling, ownership, and file privilege sets	*file_dac_chown* – lets a process change the owner user ID of a file.
System V Interprocess Communication (IPC) security	Overrides restrictions for message queues, semaphore sets, or shared memory regions	*ipc_dac_read* – lets a process read a System V IPC message queue, semaphore set, or shared memory region whose permission bits or ACL do not allow process read permission
Network security	Overrides restrictions for reserved port binding or binding to a multilevel port, sending broadcast messages, or specifying security attributes (such as labels, privileges on a message, or network endpoint defaults)	*net_broadcast* – lets a process send a broadcast packet on a specified network

Table 1-6 Privilege Categories

Privilege Category	Summary	Example Privileges in the Category
Process security	Overrides restrictions for auditing, labeling, covert channel delays, ownership, clearance, user IDs, or group IDs	*proc_mac_read* – lets a process read another process where the reading process's sensitivity label is dominated by the other process's sensitivity label
System security	Overrides restrictions for auditing, workstation booting, workstation configuration management, console output redirection, device management, file systems, creating hard links to directories, increasing message queue size, increasing the number of processes, workstation network configuration, third-party loadable modules, or label translation	*sys_boot* – lets a process halt or reboot a Trusted Solaris workstation
Window security	Overrides restrictions for colormaps, reading to and writing from windows, input devices, labeling, font paths, moving data between windows, X server resource management, or direct graphics access (DGA) X protocol extensions	*win_selection* – allows a process to request inter-window data moves without the intervention of selection arbitrator

Allowed and Forced Privilege Assignment

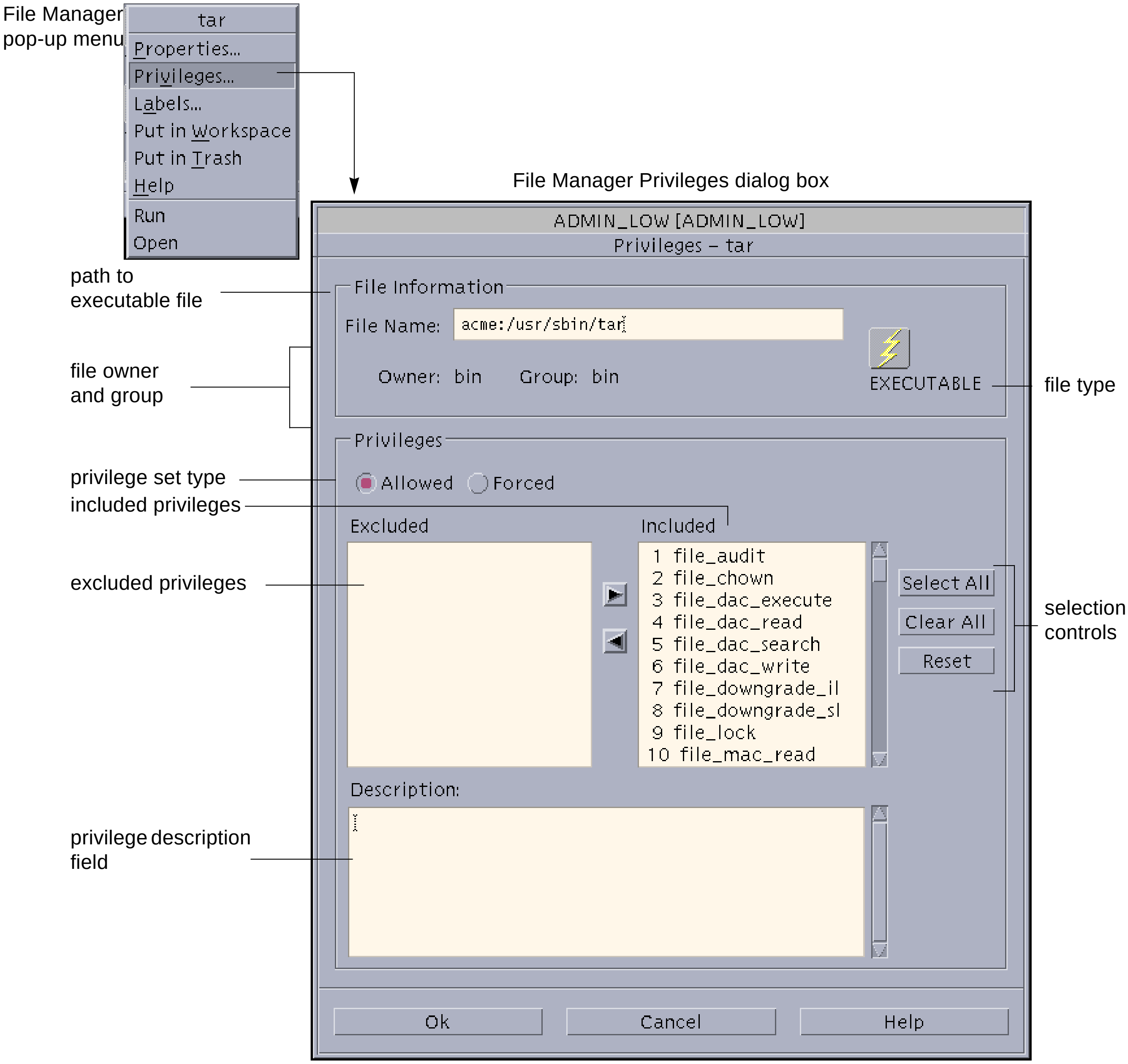
You assign allowed and forced privileges to an executable file through the File Manager. In practice, you generally include all privileges in the allowed set. If you have a privilege that should never be exercisable for this application, exclude it from the allowed set. Generally, you use forced privileges only when they are essential for the application. A privilege that is allowed but not forced can only be used if the same privilege is in the process's inheritable set.

Selecting Change Privileges in the File Manager's pop-up menu displays the File Manager Privileges dialog box for the selected application icon (see Figure 1-10). The Privileges dialog box identifies the executable file's path, owner, group, and file type (executable or script), lets you select the type of privilege set (allowed or forced) and provides two list fields for moving privileges in and out of the excluded set. The Description field describes the selected privilege. The three selection controls let you specify the entire group of privileges.

Inheritable Privilege Assignment

You assign inheritable privileges to CDE actions and commands within an execution profile using the Profile Manager. A privilege in an application's inheritable set within a profile is only available for use if it is also in the allowed set for the corresponding executable file. The application process can pass this privilege, along with other inheritable privileges (if they are allowed), to child processes that the application forks.

Note – The same application can be contained by different profiles with different sets of inheritable privileges.



ADMIN_LOW [ADMIN_LOW]

Privileges – tar

File Information

File Name: acme:/usr/sbin/tar

Owner: bin Group: bin

EXECUTABLE

Privileges

● Allowed ○ Forced

Excluded

Included

1 file_audit
2 file_chown
3 file_dac_execute
4 file_dac_read
5 file_dac_search
6 file_dac_write
7 file_downgrade_il
8 file_downgrade_sl
9 file_lock
10 file_mac_read

Description:

Figure 1-10 Assigning Privileges to a File

Privilege Availability Example

Table 1-7 presents an example of how privileges are made available to processes. It shows the allowed (A = allowed; N = not allowed), forced (marked F), and inheritable (marked I) privilege sets for a hypothetical application.

Table 1-7 Privilege Sets for an Example Application

	Privilege	**Allowed**	**Forced**	**Inheritable**
Not available because not allowed	file_mac_write	N		I
	file_upgrade_sl	N		I
Available because allowed and forced. Inheritable is redundant.	win_dga	A	F	I
	win_fontpath	A	F	I
	win_colormap	A	F	I
Available because allowed and inheritable	file_dac_search	A		I
	file_dac_read	A		I
Not available because neither forced nor inheritable	file_chown	A		
	file_dac_execute	A		

Here is how to interpret the example:

- **Allowed set** – Privileges that are not in the Allowed set are not available at all (for example, file_mac_write and file_upgrade_sl); this is a good way to ensure that powerful privileges do not become available inadvertently. Privileges that are allowed but not forced will be available for use only if they are included in a profile's inheritable set of privileges for this application (file_dac_search and file_dac_read). Privileges that are allowed but neither forced nor inheritable are unavailable (for example, file_chown and file_dac_execute).
- **Forced Set** – Privileges that are forced are available unconditionally to users who can run the application; a privilege cannot be forced without being allowed (win_dga, win_fontpath, and win_colormap are forced).
- **Inheritable Set** – Inheritable privileges are included in an execution profile by assignment. Notice that all the privileges are shown as inheritable except file_chown and file_dac_execute. Being inheritable is not sufficient for being available; those privileges that are inheritable but not allowed are not available (for example, file_mac_write and file_upgrade_sl).

How Trusted Solaris Controls Device Access

Because devices provide a means for the import and export of data to and from a Trusted Solaris system, they must be controlled to properly protect the data. (A *device* is either a physical peripheral that is connected to a Trusted Solaris system or a software-simulated device called a pseudo-device.) Trusted Solaris lets you control data flowing through devices through device allocation and device label ranges.

For information on the tools related to device allocation, see “Devices and Drivers” on page 143.

Device Allocation

Device allocation provides a way to control data when it is imported and exported and prevents unauthorized users from access to the information. In a Trusted Solaris system the administrator decides which devices, if any, each user can use to import and export data and sets those devices to be allocatable. The administrator then assigns to selected users the authorization needed to allocate a device. Users authorized to use a device must allocate the device before using it and deallocate the device when finished. Between its allocation and deallocation the user has exclusive use of the device.

Device Label Ranges

Each allocatable device has an associated sensitivity label range that is assigned by an administrator. To use an allocatable device, the user must be currently at a process sensitivity label within the device’s label range; if not, allocation is denied. The user’s process sensitivity label is applied to data imported or exported while the device is allocated to the user. The sensitivity label and information label of exported data are displayed when the device is deallocated so that the user can physically label the medium containing the exported data.

Examples of devices that have label ranges are frame buffers, tape drives, diskette and CD-ROM drives, and printers.

For more information on managing devices, see “Devices and Drivers” on page 143.

Controlling File Access: An Example 2

To gain access to various parts and resources of the system, users need permissions, possibly applications with privileges, and, in many cases, authorizations. This chapter provides an extended example of how data is protected in the Trusted Solaris environment. It demonstrates how security attributes associated with the user's account, the executable file, the process, and the data file combine to permit or deny access to processes attempting to interact with data files.

Overview: Security Attributes

To enforce data protection, Trusted Solaris uses a combination of

- **discretionary access control (DAC)** – UNIX permissions and access control lists set by users
- **mandatory access control (MAC)** – security clearances and information sensitivity rules for your site
- **authorizations and privileges** – rights granted by the security administrator

The elements of these controls are broadly classified as *security attributes*. A security attribute is the property of an entity (file, directory, process, device, or network interface) in the Trusted Solaris environment related to security. Security attributes can be classified according to where they are applied:

- user account security attributes
- file security attributes
- process security attributes

Figure 2-1 presents the big picture of where security attributes are used, stored, and maintained in the Trusted Solaris environment. The left column of the diagram shows where security attributes are used in transactions. The middle column shows the databases that store security attributes (and other information). The right column shows the graphical tools used by administrators to maintain this information.

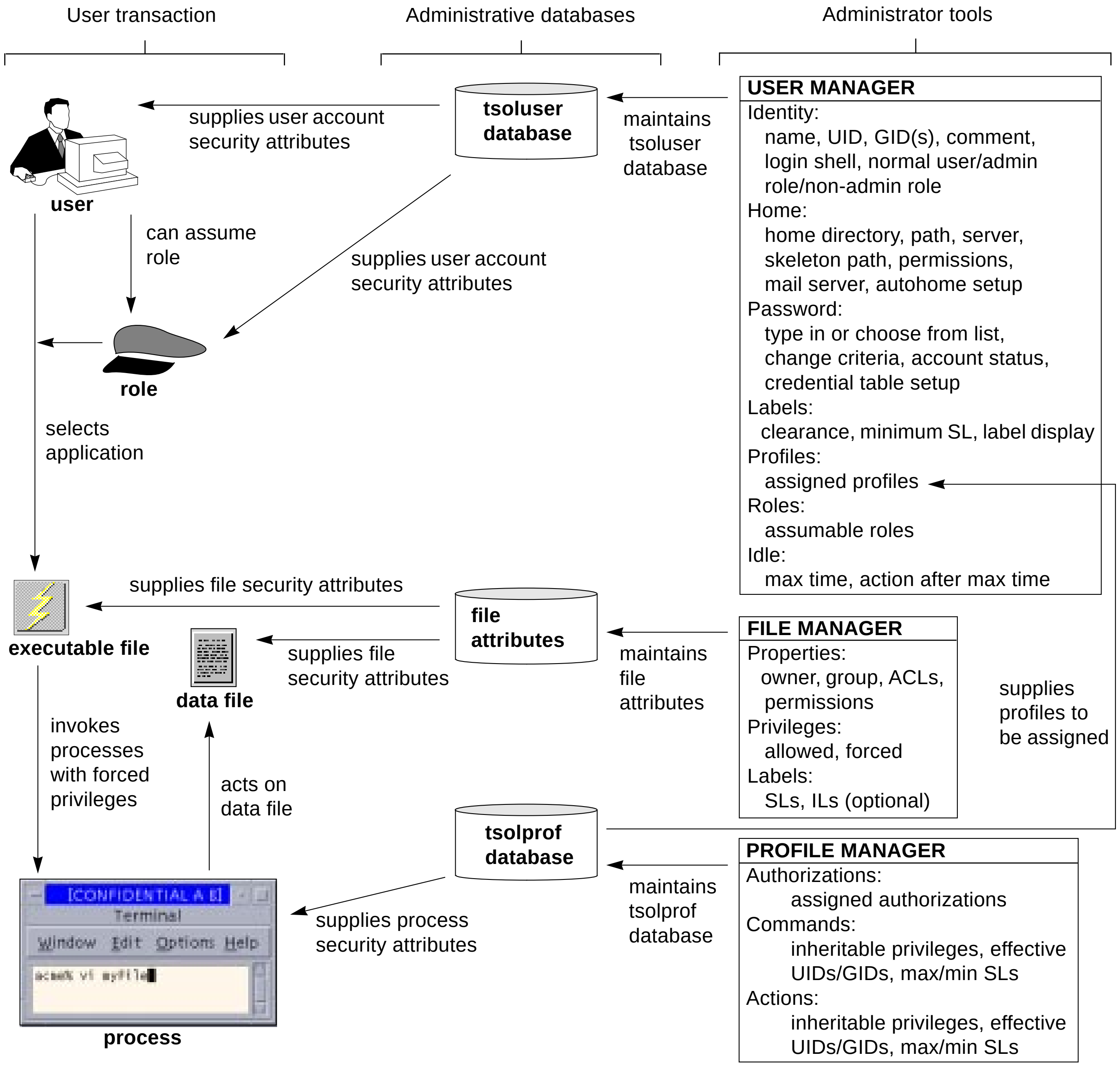

User transaction
Administrative databases
Administrator tools
user
supplies user account security attributes
tsoluser database
maintains tsoluser database
USER MANAGER
Identity:
name, UID, GID(s), comment,
login shell, normal user/admin
role/non-admin role
Home:
home directory, path, server,
skeleton path, permissions,
mail server, autohome setup
Password:
type in or choose from list,
change criteria, account status,
credential table setup
Labels:
clearance, minimum SL, label display
Profiles:
assigned profiles
Roles:
assumable roles
Idle:
max time, action after max time
can assume role
supplies user account security attributes
role
selects application
supplies file security attributes
executable file
file attributes
maintains file attributes
data file
supplies file security attributes
FILE MANAGER
Properties:
owner, group, ACLs,
permissions
Privileges:
allowed, forced
Labels:
SLs, ILs (optional)
supplies profiles to be assigned
invokes processes with forced privileges
acts on data file
tsolprof database
maintains tsolprof database
PROFILE MANAGER
Authorizations:
assigned authorizations
Commands:
inheritable privileges, effective
UIDs/GIDs, max/min SLs
Actions:
inheritable privileges, effective
UIDs/GIDs, max/min SLs
[CONFIDENTIAL A B]
Terminal
Window Edit Options Help
acme% vi myfile
supplies process security attributes
process

Figure 2-1 Security Attribute Relationships in the Trusted Solaris Environment

User Account Security Attributes

Security attributes for the user, along with other account information, are stored in the `tsoluser` database, which is maintained through a tool called the User Manager. User account information falls into seven categories, each of which has a separate dialog box accessed from the User Manager:

- Identity – identifies users
- Home – assigns home directory information
- Password – defines password selection method and change criteria
- Labels – sets user’s label range and label displays
- Profiles – determines the applications and associated security attributes that the user may access
- Roles – grants user access to accounts with special capabilities
- Idle – sets maximum time a workstation may be unattended and action taken when the maximum is exceeded

A role is simply a special user account that a user assumes after logging in. A role can have no other role assigned to it. When the user assumes a role, the user gets a different set of security attributes.

File Security Attributes

The executable file’s security attributes are maintained through the File Manager. The File Manager’s Selected menu and pop-up menu each have commands for changing a file’s properties, privileges, and sensitivity labels (SL).

When the user selects an application, either as a normal user or from a role, the file security attributes of the executable file play a role in the transaction. For example, the user cannot access the executable file without the proper permissions and a dominating SL (unless the user has overriding privileges).

The executable file can also have allowed and forced privileges assigned to it. Privileges that are not allowed can never be used with this application; allowed privileges can be used if they are assigned as forced privileges or if they are included as inheritable privileges assigned to the application in one of the user’s execution profiles. (See “Understanding Privileges” on page 27 for an explanation of privileges.)

The data file acted on in the transaction also has file security attributes. In this case, the permissions and SL determine whether the process invoked by the user is permitted access to the data file. Allowed and forced privileges are available to data files but have no real meaning, just the same as an executable permission (x) has no meaning for a data file.

Process Security Attributes

The process attributes come mainly from the user's execution profiles. Forced privileges are the exception and come from the executable file's file properties. The user's profile contains the command or action that can invoke the process and any associated security attributes. Execution profiles are stored in the `tsolprof` database and are maintained through the Profile Manager. The Profile Manager has three views in which authorizations and commands and actions, with their associated security attributes, can be included in execution profiles.

How Security Attributes are Applied in Transactions

Figure 2-2 shows a transaction involving a process acting on data and indicates the sources of specific security attributes.

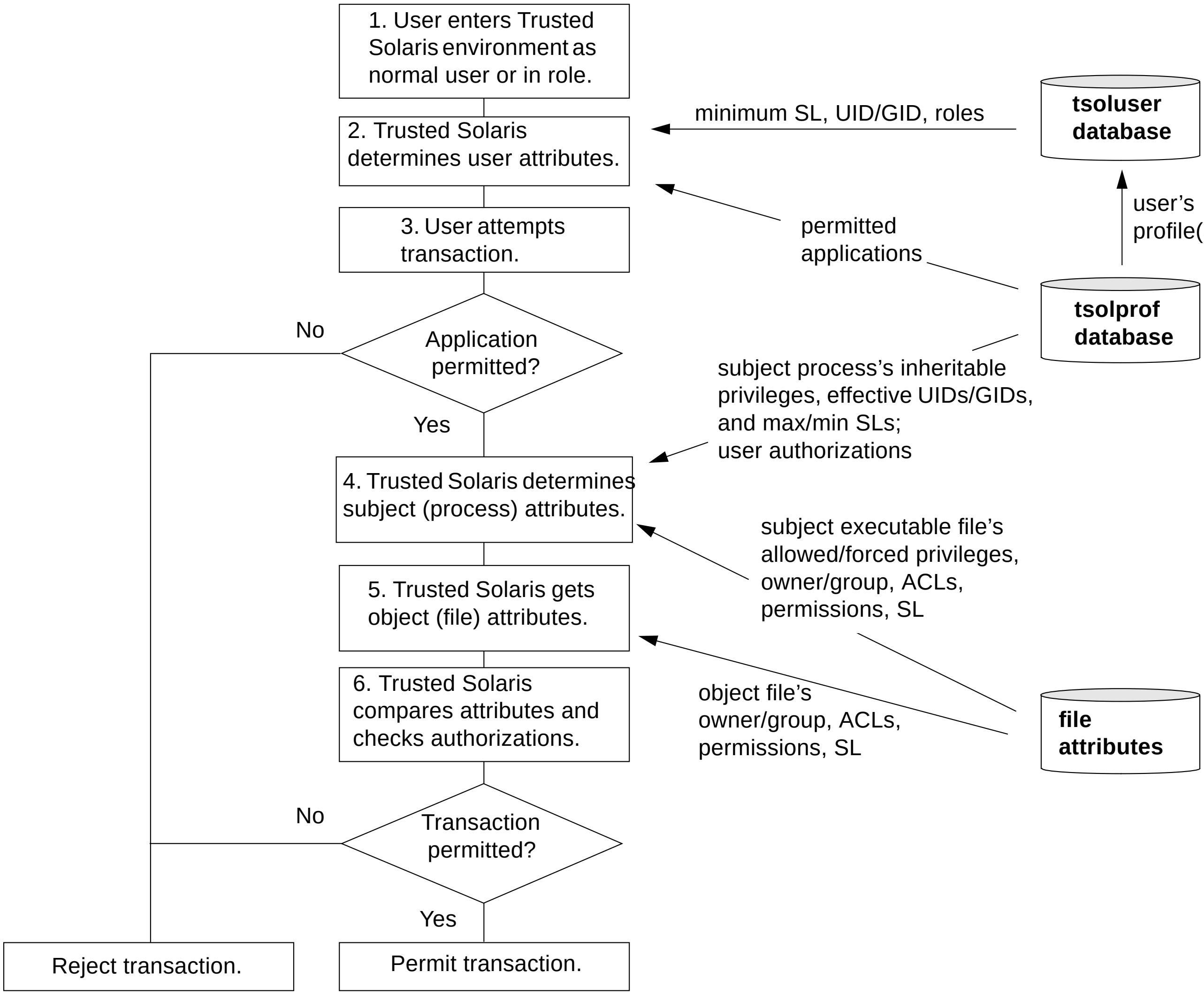

1. User enters Trusted Solaris environment as normal user or in role.
2. Trusted Solaris determines user attributes.
3. User attempts transaction.
Application permitted?
No
Yes
4. Trusted Solaris determines subject (process) attributes.
5. Trusted Solaris gets object (file) attributes.
6. Trusted Solaris compares attributes and checks authorizations.
Transaction permitted?
No
Yes
Reject transaction.
Permit transaction.
tsoluser database
minimum SL, UID/GID, roles
user's profile(s)
tsolprof database
permitted applications
subject process's inheritable privileges, effective UIDs/GIDs, and max/min SLs; user authorizations
subject executable file's allowed/forced privileges, owner/group, ACLs, permissions, SL
object file's owner/group, ACLs, permissions, SL
file attributes

Figure 2-2 How Trusted Solaris Mediates Transactions

An explanation of the diagram follows:

- **1. User enters Trusted Solaris environment.** – When the user enters the Trusted Solaris environment, the user has to supply his or her user name and password. These entries are checked against the `passwd` and `shadow` files. If the user assumes a role, the user must enter the password for that role as well. For auditing purposes, the user's audit UID is always set to the user's personal UID rather than the role's UID.
- **2. Trusted Solaris determines user attributes.** – Upon entering the Trusted Solaris environment, the user is provided with a workspace set to the appropriate SL for the session type—single- or multilabel. If the user has selected a multilabel session, the user can change the workspace SL to any SL from the minimum SL assigned to the user account up to the highest SL permitted in the session. The user's UID and GID determine which files the user can access (without the use of privileges). The account's profile(s) determine the set of applications and security attributes permitted to the user. If the user assumes a role, the user gets a different set of applications and security attributes.
- **3. User attempts transaction.** – If a site assigns the All execution profile to all users, then any user can access any command or action although not necessarily with privileges. In more restrictive sites that do not assign the All profile, attempts to run prohibited commands return the error message: `command not in profile and` prohibited action icons do not appear in the interface or, if they appear, they return an error message. If Trusted Solaris cannot find the application in a profile, the transaction request is rejected.
- **4. Trusted Solaris determines subject (process) attributes.** – When the user selects an application, the account security attributes, in combination with the application's executable file's security attributes, determine the capabilities of the process. The profile containing the application defines the inheritable privileges, effective UID/GID, and maximum and minimum sensitivity label security attributes for the process. Authorizations are available from this profile and any other profiles assigned to this UID. The executable file provides the allowed and forced privileges.
- **5. Trusted Solaris gets object (file) attributes.** – The security attributes protecting the access to the data file are owner, group, ACLs, permissions, and the data file SL.

- **6. Trusted Solaris compares attributes and checks authorizations.** – Trusted Solaris compares the subject (process) attributes with the object (file) attributes to determine if the transaction is permitted. These are some typical tests:
  - Does the process's sensitivity label dominate the file's directory label? If not, does the process have the file_mac_search privilege for accessing directories without dominating them?
  - Does the process's permissions let it access the file's directory? If not, does the process have the file_dac_search privilege for accessing directories without the right permissions?
  - Does the process's sensitivity label dominate the file's sensitivity label? If not, does the process have the file_mac_read privilege for accessing files without dominating them?
  - Does the process's permissions let it access the file? If not, does the process have the file_dac_read privilege for accessing files without the right permissions?
  - Does the process require any special authorizations?

Examples: Security Attributes in Transactions

In these two examples, a user named Sam attempts to run `tar` to save a file called `testFile`. Sam is allowed to run `tar` and has the permissions necessary for reading and writing `testFile`. In example #1, Sam attempts the transaction as a normal user and is unsuccessful. In example #2, Sam assumes the oper role, and the transaction successfully completes.

Example #1: Failed Transaction by Normal User

In Figure 2-3, Sam's user account has been assigned the All execution profile, which gives him access to `tar` but without any privileges. Since Sam is thus permitted to use `tar`, the transaction proceeds to steps #3 through #5 in which Trusted Solaris gathers and compares the subject's (the `tar` process) and the object's (the `testFile` file) security attributes. The conditional tests in this case are as follows:

- **Does the process's SL dominate the SL for the file and its directory?** – The `tar` process's SL is C due to Sam's session selection when he began the session. Since `testFile` and its directory are also at SL = C, the process's

SL dominates (in this case is equal), and the process does not need the file_mac_search privilege for reading the directory or the file_mac_read privilege for reading the files.

- **Does the process's permissions let it access the file's directory?** – Sam owns the directory and file, so he does not need the file_dac_search privilege for accessing the directory nor the file_dac_read privilege for reading the file.
- **Does the process require any special authorizations?** – Using `tar` requires an available magnetic tape device. Sam works in a secure environment where only administrators can allocate devices. Since Sam does not have the allocate device authorization, the transaction cannot be completed.

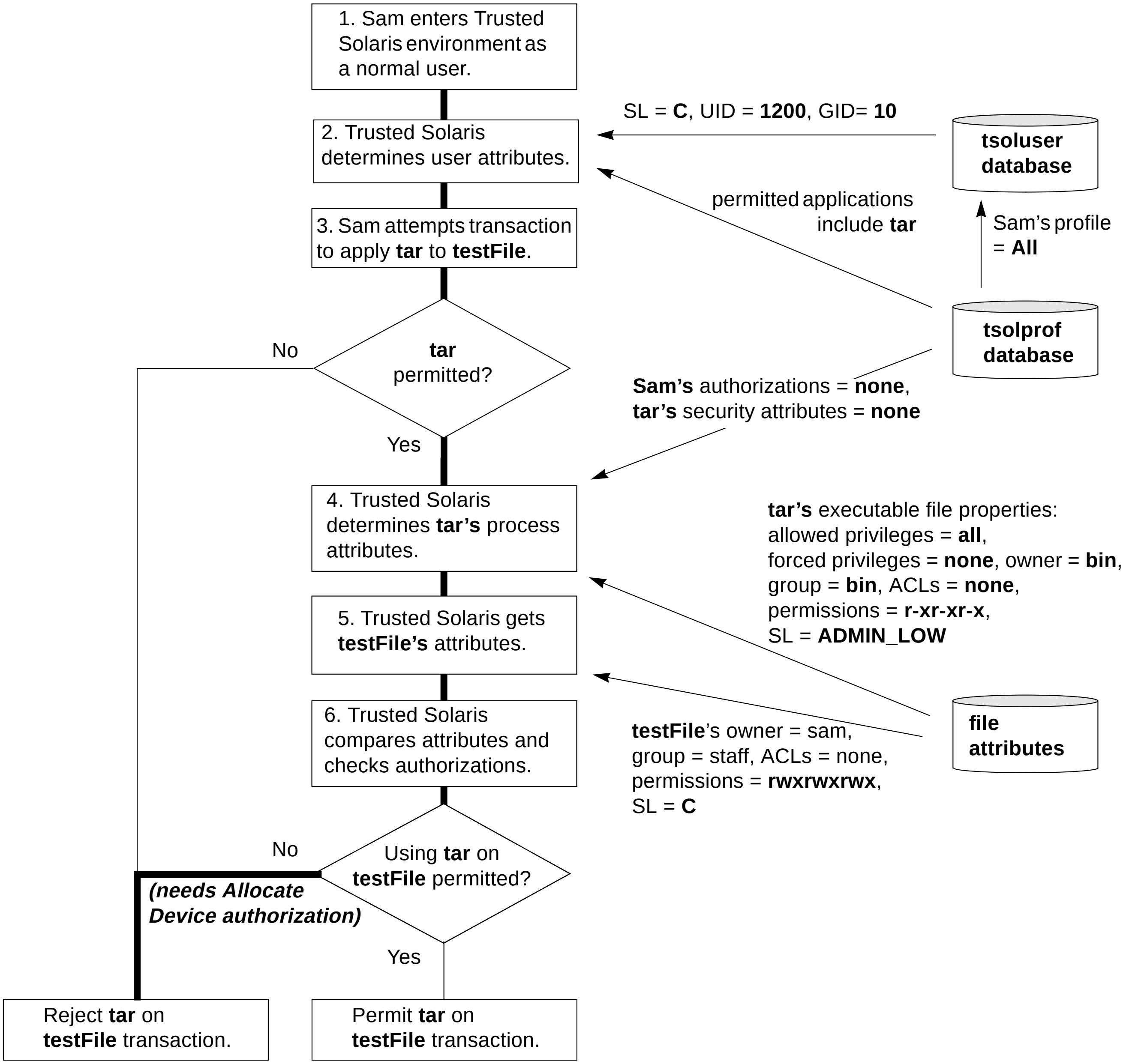

1. Sam enters Trusted Solaris environment as a normal user.
2. Trusted Solaris determines user attributes.
3. Sam attempts transaction to apply **tar** to **testFile**.
tar permitted?
No
Yes
4. Trusted Solaris determines **tar's** process attributes.
5. Trusted Solaris gets **testFile's** attributes.
6. Trusted Solaris compares attributes and checks authorizations.
Using **tar** on **testFile** permitted?
No
Yes
(needs Allocate Device authorization)
Reject **tar** on **testFile** transaction.
Permit **tar** on **testFile** transaction.
SL = **C**, UID = **1200**, GID= **10**
tsoluser database
permitted applications include **tar**
Sam's profile = **All**
tsolprof database
Sam's authorizations = **none**, **tar's** security attributes = **none**
tar's executable file properties: allowed privileges = **all**, forced privileges = **none**, owner = **bin**, group = **bin**, ACLs = **none**, permissions = **r-xr-xr-x**, SL = **ADMIN_LOW**
file attributes
testFile's owner = sam, group = staff, ACLs = none, permissions = **rwxrwxrwx**, SL = **C**

Figure 2-3 Example #1: Unsuccessful Transaction as Normal User

Example #2: Successful Transaction from oper Role

When Sam assumes the oper role, the transaction succeeds, as shown in Figure 2-4. The oper account has the Basic Media execution profile assigned to it, among others. The Basic Media execution profile includes the `tar` command with the file_dac_search, file_dac_read, file_mac_read, and other privileges and with a range of SLs from ADMIN_LOW to ADMIN_HIGH. Since `tar` is permitted in this case, too, the transaction proceeds to steps #3 through #5. This time the transaction passes the conditional tests as follows:

- **Does the process's SL dominate the SL for the file and its directory?** – The tar process's SL is ADMIN_LOW, which is the default SL when Sam assumes the oper role. Since `testFile` and its directory have an SL of C, the `tar` process does not dominate their SLs and must exercise the file_mac_search and file_mac_read privileges.
- **Does the process's permissions let it access the file's directory?** – Since oper does not own the directory or file, the process uses the file_dac_search and file_dac_read privileges.
- **Does the process require any special authorizations?** – In contrast to Sam's failed attempt as a normal user, the Basic Media execution profile includes the allocate device authorization. As oper, Sam can allocate the magnetic tape from the Device Manager and perform the `tar` transaction.

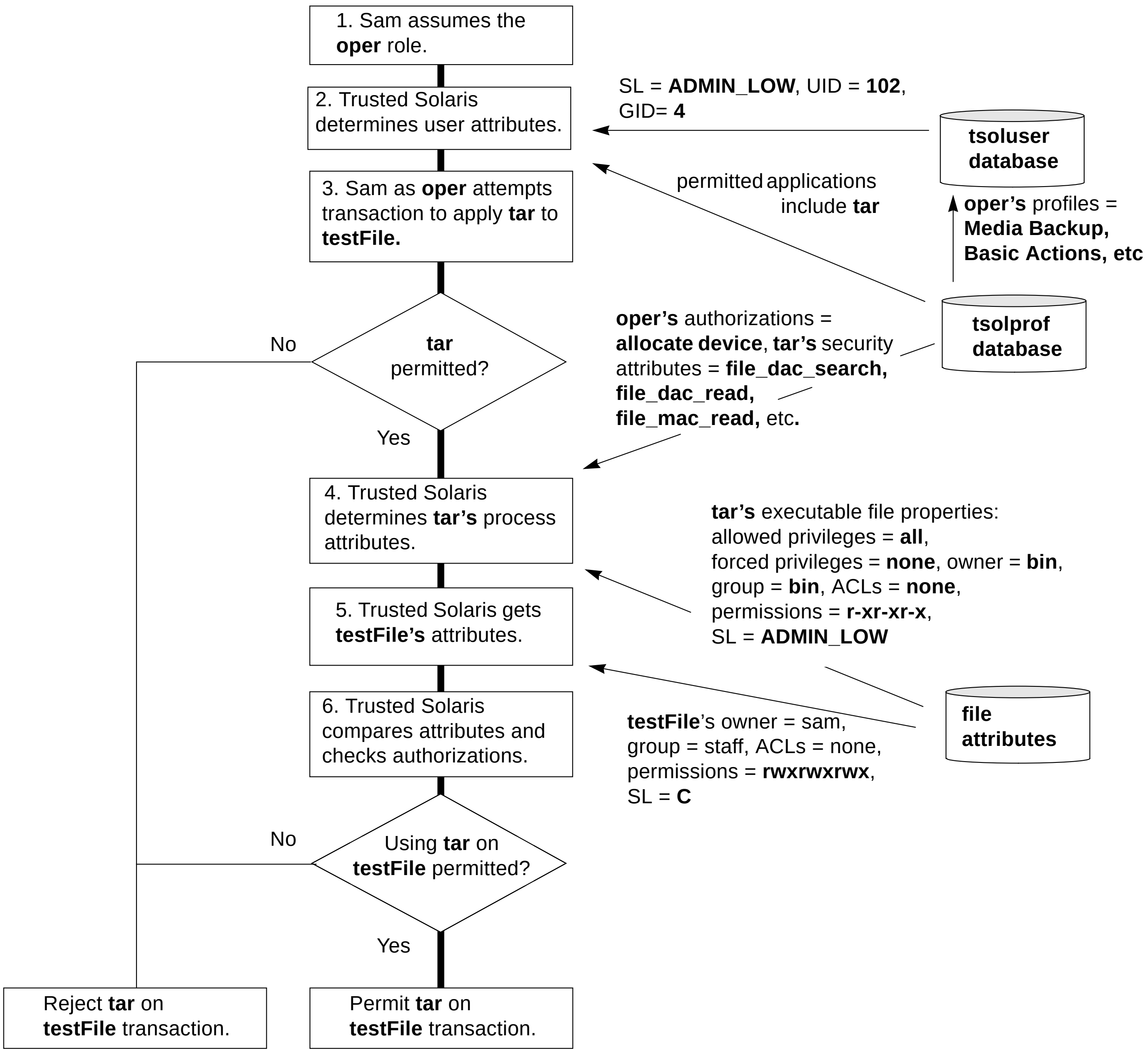

1. Sam assumes the **oper** role.
2. Trusted Solaris determines user attributes.
SL = **ADMIN_LOW**, UID = **102**, GID= **4**
tsoluser database
permitted applications include **tar**
oper's profiles = **Media Backup, Basic Actions, etc**
3. Sam as **oper** attempts transaction to apply **tar** to **testFile.**
tsolprof database
oper's authorizations = **allocate device**, **tar's** security attributes = **file_dac_search, file_dac_read, file_mac_read,** etc**.**
No
tar permitted?
Yes
4. Trusted Solaris determines **tar's** process attributes.
tar's executable file properties: allowed privileges = **all**, forced privileges = **none**, owner = **bin**, group = **bin**, ACLs = **none**, permissions = **r-xr-xr-x**, SL = **ADMIN_LOW**
5. Trusted Solaris gets **testFile's** attributes.
file attributes
testFile's owner = sam, group = staff, ACLs = none, permissions = **rwxrwxrwx**, SL = **C**
6. Trusted Solaris compares attributes and checks authorizations.
No
Using **tar** on **testFile** permitted?
Yes
Reject **tar** on **testFile** transaction.
Permit **tar** on **testFile** transaction.

Figure 2-4 Example #2: Successful Transaction from oper Role

Quick Tour of the Admin Tools 3

This chapter presents an overview of the tools available in the Trusted Solaris environment, how they are accessed, and the databases on which they operate.

3

Accessing the Administrator Tools: Overview

The graphical administrator tools in the Trusted Solaris environment are accessed from the Front Panel and the Application Manager, as shown in Figure 3-1.

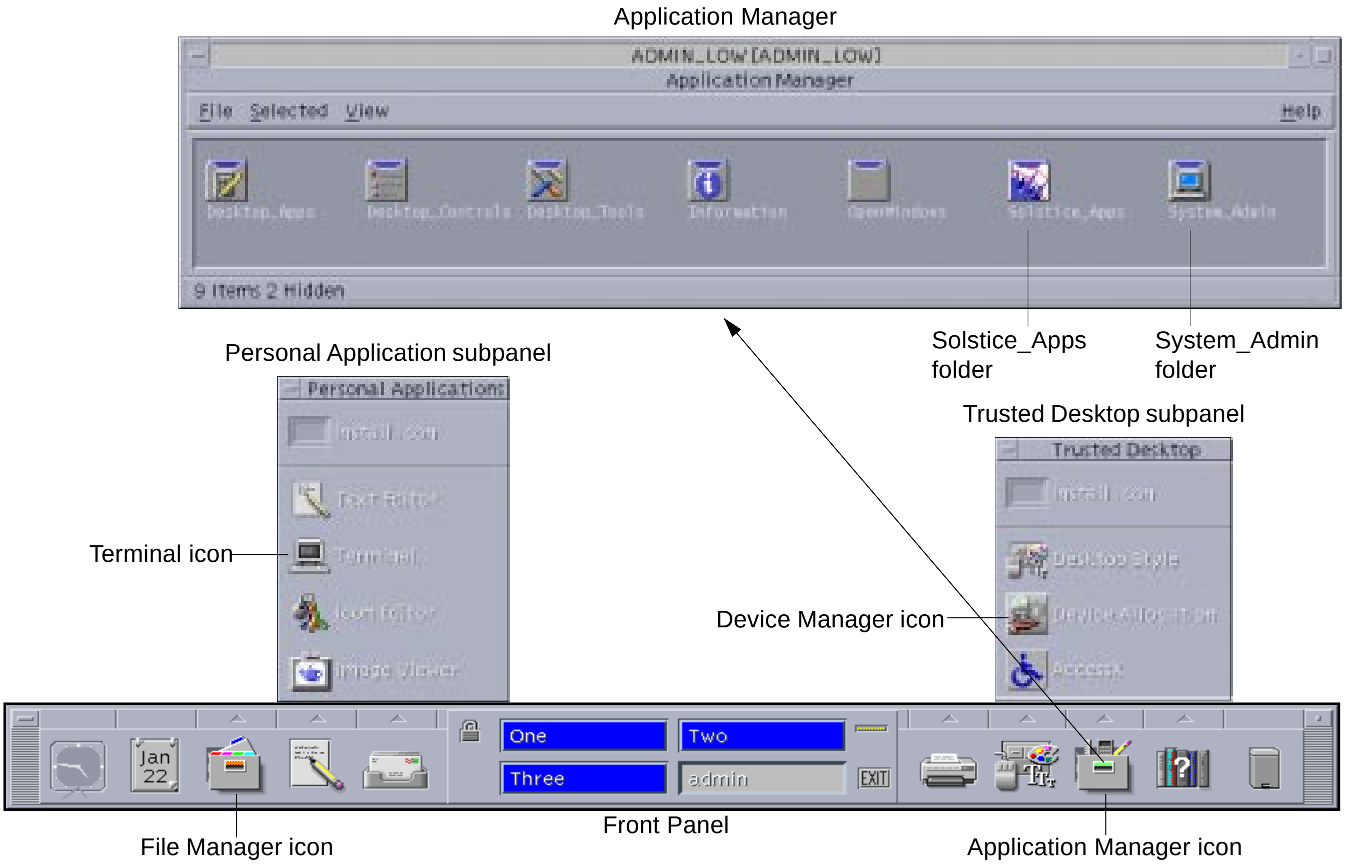


Figure 3-1 Accessing the Administrator Tools

Accessing the File Manager

The File Manager icon appears on the left side of the Front Panel. The File Manager permits all users to see and operate on their own files and directories. The basic File Manager operations are documented in the base Solaris documentation.

Users with privileges can perform limited operations on file and directory labels. This is covered in more detail in Chapter 5, “Managing Files and Directories,” in the *Trusted Solaris User’s Guide*.

Administrators can change the privileges and labels on files and directories. This is described in “Using the File Manager to Change Privileges and Labels” on page 126.”

Accessing the Device Allocation Manager

The Device Allocation Manager icon is accessible from the Trusted Desktop subpanel (see Figure 3-1). You can also access it from the Allocate Device menu option in the Trusted Path menu. The Device Allocation Manager is described in “Administering Devices through the Device Allocation Manager” on page 143.

Accessing the Application Manager

The Application Manager icon is accessed from the right side of the Front Panel. It operates in similar fashion to the base Solaris Application Manager, that is, applications can be launched from its folders. In the Trusted Solaris environment, the Application Manager provides major graphical tools in the Solstice_Apps folder and special text editors linked to system databases in the System_Admin folder.

Accessing Command Line Tools

Command line tools are directly available from terminal windows for users in the system or security administrator role. Users in the root role must first type **pfsh** to enter the profile shell belonging to root and then enter the desired shell type: **sh**, **csh**, **ksh**, etc. The commands available vary according to the profiles assigned to the role.

Solstice_Apps Folder

The Solstice_Apps folder in the Application Manager provides access to the major Trusted Solaris graphical tools. Figure 3-2 shows the complete contents of the Solstice_Apps folder. Note that the security administrator and the system administrator roles can only access a subset of these tools, for security purposes.

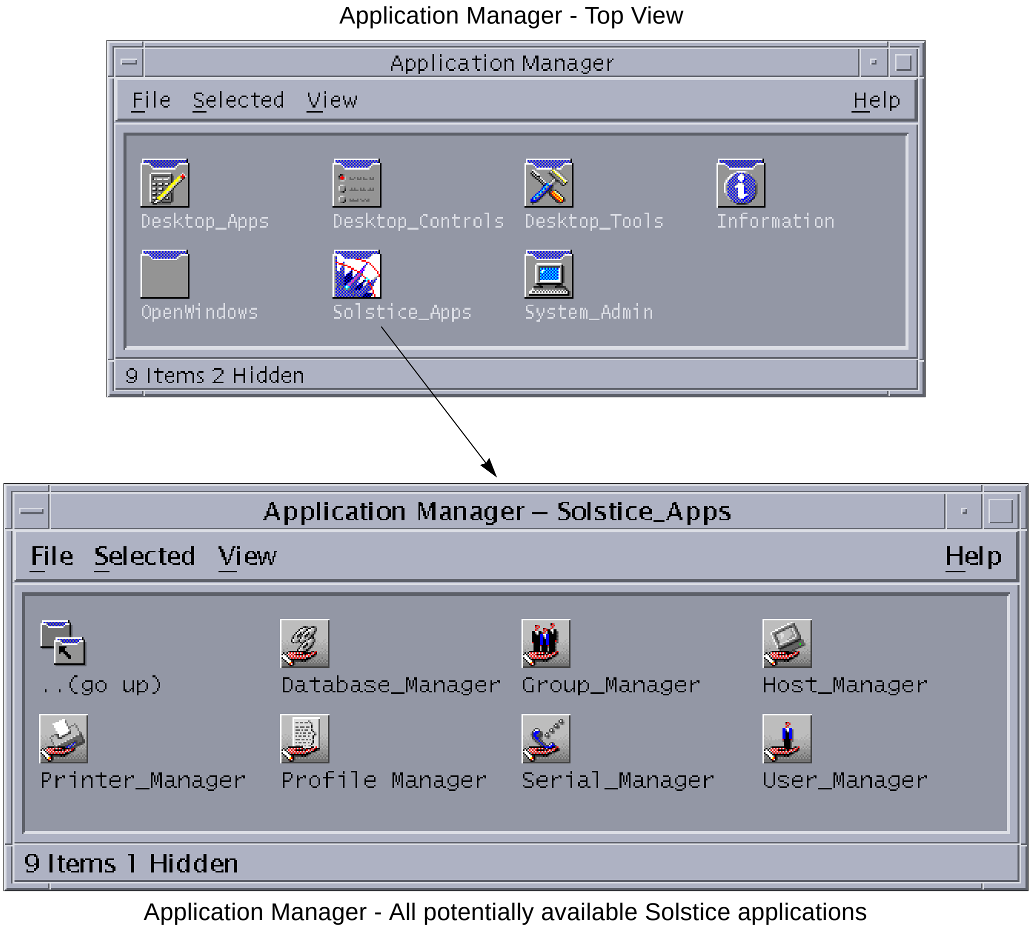


Figure 3-2 Solstice Application Folder

Solstice_Apps Folder Tools Available to the System Administrator

Figure 3-3 shows the tools in the Solstice_Apps folder that can be accessed by users in the system administrator role.

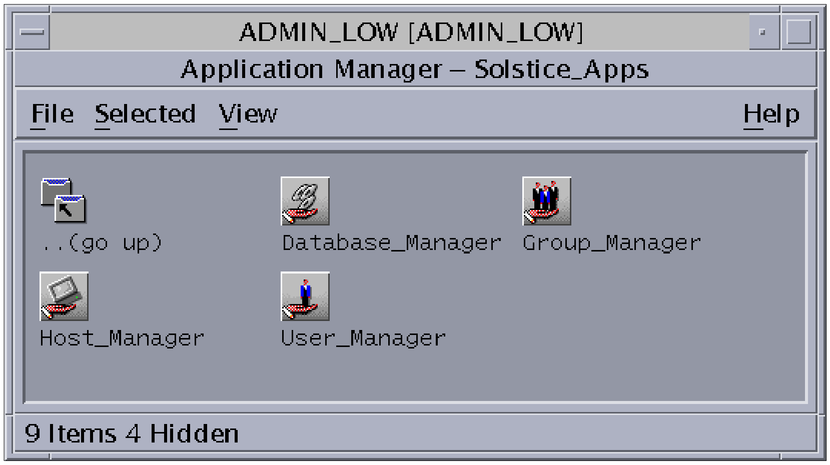


Figure 3-3 Solstice Applications Accessible by the System Administrator

The applications available to the system administrator in the Solstice_Apps folder are as follows. Those items without descriptions are the same as in base Solaris.

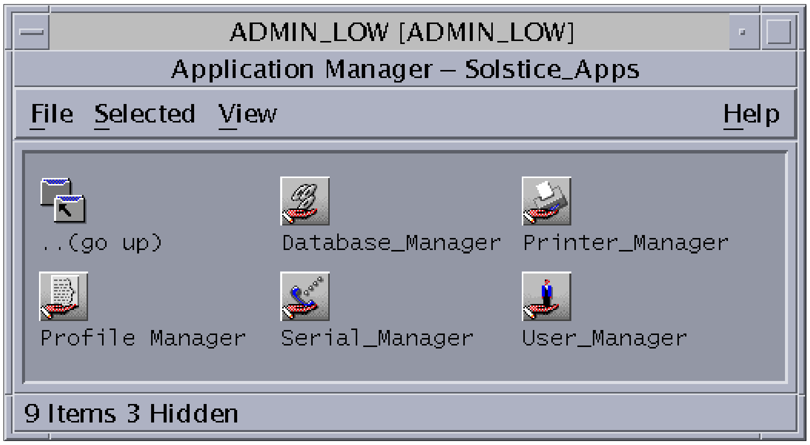
- **Database Manager** – lets the system administrator edit these databases:
    - Aliases
    - Bootparams – contains minor modifications for the Trusted Solaris environment.
    - Ethers
    - Group
    - Hosts
    - Locale
    - Netgroup
    - Netmasks
    - Networks
    - Protocols
    - RPC
    - Services

  - Timezone
  - Tnidb – a special Trusted Solaris database that holds information on network interfaces. It is managed on the local host. See “The tnidb Database” on page 96.

- **Host Manager**
- **Group Manager** – permits the system administrator to manage groups. Groups can also be managed using the Database Manager. In either case, all groups (local and network) must have a unique group ID (GID) for that network. The uniqueness of a GID must be manually checked by an administrator. Before a group can be deleted, the system administrator must check that all objects with this GID are deleted from the system or reassigned to another group and that any users assigned to this group as their primary group are reassigned to another group. (For auditing purposes, you never reuse a group name or GID from a deleted group).
- **User Manager** – lets the system administrator edit the `tsoluser` database, which holds user and role account information. Specifically, the system administrator can edit identity and home directory information. See Chapter 4, “Administering Users.”

Solstice_Apps Folder Tools Available to the Security Administrator

Figure 3-4 shows the tools in the Solstice_Apps folder that can be accessed by users in the security administrator role.



ADMIN_LOW [ADMIN_LOW]

Application Manager – Solstice_Apps

9 Items 3 Hidden

Figure 3-4 Solstice Applications Accessible by the Security Administrator

The applications available to the security administrator in the Solstice_Apps folder are as follows. Those items without descriptions are the same as in base Solaris.

- **Database Manager** – lets security administrator edit these databases as follows:
  - Auto_home (security administrator only)
  - Tnrhdb – a special Trusted Solaris database that holds networking information concerning remote hosts. See "The tnrhdb Database" on page 90. (security administrator only)
  - Tnrhtp – a special Trusted Solaris database that holds network security templates that can be applied to remote hosts. See "The tnrhtp Database" on page 93. (security administrator only)
- **Printer Manager**
- **Profile Manager** – lets you edit `tsolprof(4TSOL)`, the database that holds execution profile information. See "Using the Profile Manager" on page 120.
- **Serial Manager**
- **User Manager** – lets the security administrator edit the `tsoluser` database, which holds user and role account information. Specifically, the security administrator can edit password, label, profile, role, and idle directory information. See Chapter 4, "Administering Users."

System_Admin Folder

The System_Admin folder provides special actions for performing minor system administration tasks (see Figure 3-5). The actions that affect security are available only to the security administrator and actions not relevant to security are available only to the system administrator.

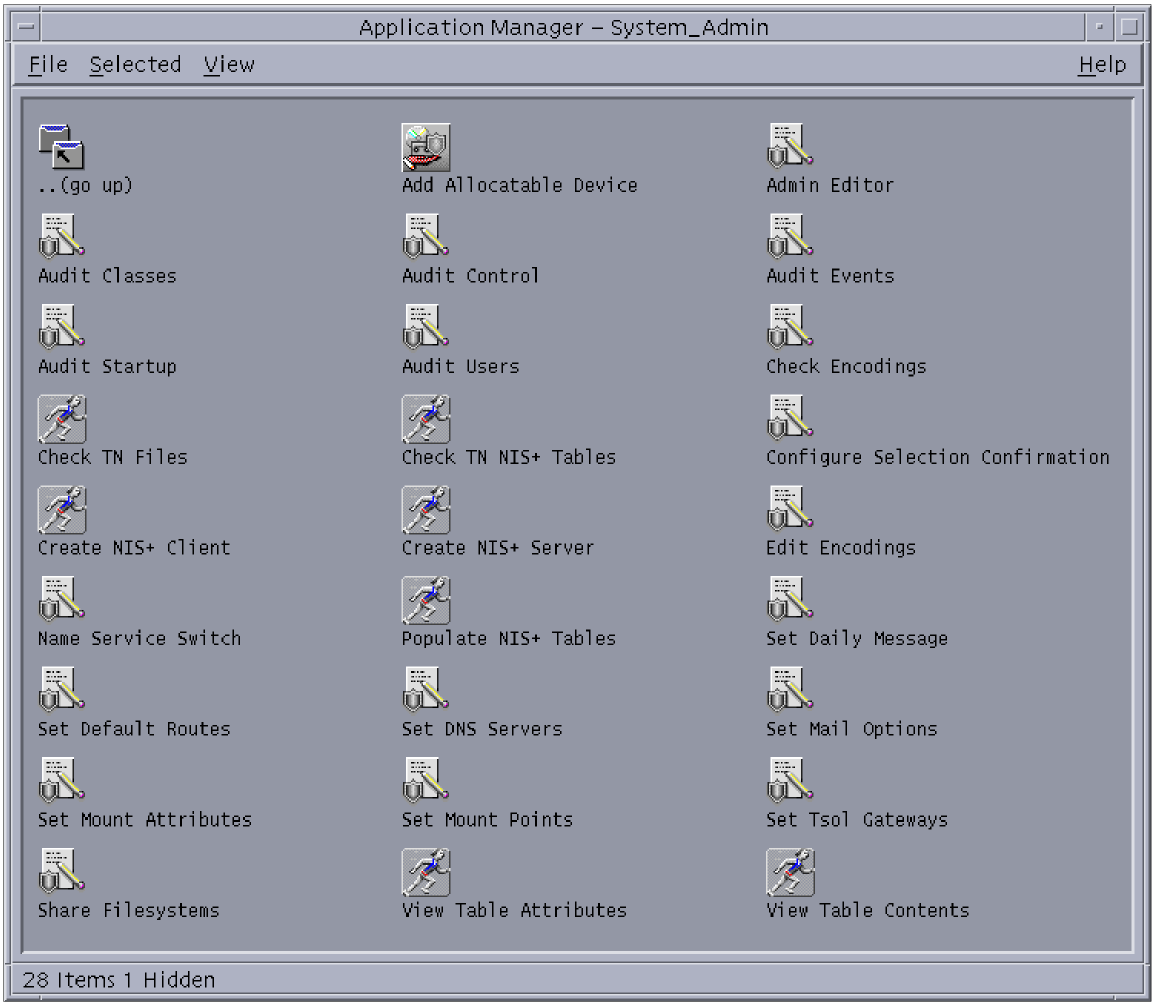


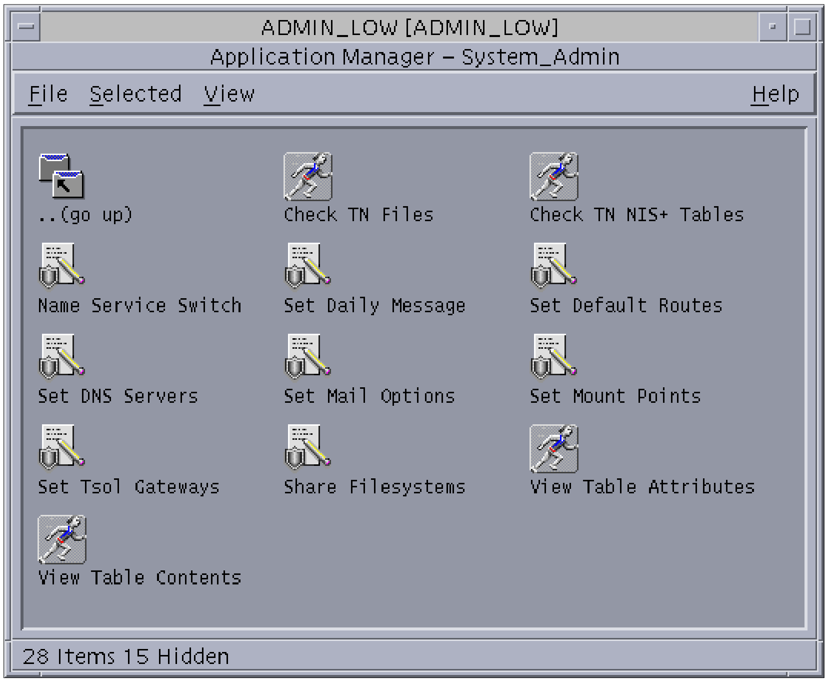
Figure 3-5 Application Manager System_Admin Folder

Most of these actions apply a special version of the vi editor, `adminvi(1MTSOL)`, to one of the system databases by default; you can substitute the dtpad editor as well. These special file actions are restricted; they cannot save a file to a different name, create a new file, or be used to escape to shell. These restrictions prevent you from inadvertently creating copies of the database that the system will not recognize. The editors also conform with mandatory access control and the local security policy.

System_Admin Folder Tools Available to the System Administrator

The system administrator can run the following actions in the System_Admin folder (see Figure 3-6):

- **Check TN Files** – checks the consistency of the local `tnrhdb` and `tnrhtp` files.
- **Check TN NIS+ Tables** – checks the consistency of the NIS+ `tnrhdb` and `tnrhtp` tables.
- **Name Service Switch** – edits the `/etc/nsswitch.conf` file for designating the search order for name services.
- **Set Daily Message** – edits the `/etc/motd` file for setting the message of the day.
- **Set Default Routes** – edits the `/etc/defaultrouter` file for setting up basic static routing.
- **Set DNS Servers** – edits the `/etc/resolv.conf` file, the configuration file for name server routines.
- **Set Mail Options** – edits `/etc/sendmail.cf`, the file for defining the mail environment.
- **Set Mount Points** – edits `/etc/vfstab`, the file for specifying the base mounting attributes.
- **Set Tsol Gateways** – specifies routes for static routing.
- **Share Filesystems** – edits `etc/dfs/dfstab`, the file containing commands for sharing resources across a network.
- **View Table Attributes** – opens a terminal window and applies the `niscat` command with the `-o` option to the specified NIS+ table.
- **View Table Contents** – opens a terminal window and applies the `niscat` command to the specified NIS+ table.



ADMIN_LOW [ADMIN_LOW]

Application Manager – System_Admin

28 Items 15 Hidden

Figure 3-6 System Administrator's View of the System_Admin Folder

System_Admin Folder Tools Available to the Security Administrator

The security administrator can run the following actions in the System_Admin folder (see Figure 3-7):

- **Add Allocatable Device** – adds an alloatabe device to the system databases.
- **Admin Editor** – applies the special version of the `dtpad` text editor directly to a file specified by the user.
- **Audit Classes** – edits the `/etc/security/audit_class` file.
- **Audit Control** – edits the `/etc/security/audit_control` file.
- **Audit Events** – edits the `/etc/security/audit_events` file.
- **Audit Startup** – edits the `/etc/security/audit_startup` file.
- **Audit Users** – edits the `/etc/security/audit_user` file.
- **Check Encodings** – checks the syntax of the specified label encodings file and displays the results in a window.
- **Check TN Files** – checks the consistency of the local `tnrhdb` and `tnrhtp` files.

- **Check TN NIS+ Tables** – checks the consistency of the NIS+ `tnrhdb` and `tnrhtp` tables.
- **Configure Selection Confirmation** – specifies upgrade/downgrade policy when data is moved.
- **Create NIS+ Client** – specifies a host as a NIS+ client.
- **Create NIS+ Server** – specifies a host as a NIS+ server.
- **Edit Encodings** – edits the specified label encodings file and automatically runs the Check Encodings action immediately after the file is saved.
- **Populate NIS+ Tables** – populate NIS+ tables from files.
- **Set Mount Attributes** – edits `/etc/vfstab_adjunct`, the file for specifying the security-related mounting attributes.
- **View Table Contents** – opens a terminal window and applies the `niscat` command to the specified NIS+ table.

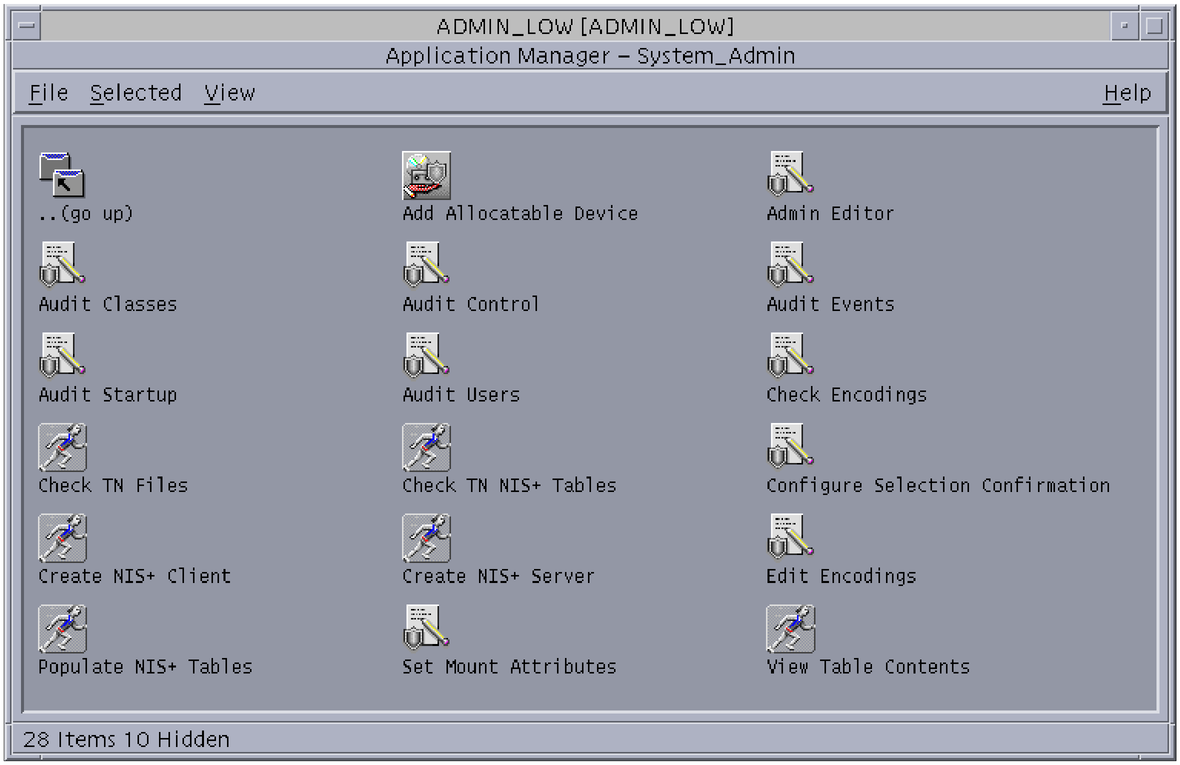


Figure 3-7 Security Administrator's View of the System_Admin Folder

3

Command Line Tools Summary

The commands available to administrators that are unique to the Trusted Solaris environment or that have been modified for the environment are listed in Table 3-1. For complete descriptions of these commands, see their man pages.

Table 3-1 User and Administrator Commands

adminvi(1MTSOL)
adornfc(1TSOL)
allocate(1MTSOL)
atohexlabel(1MTSOL)
chk_encodings(1MTSOL)
deallocate(1MTSOL)
device_clean(1MTSOL)
devpolicy(1MTSOL)
dminfo(1MTSOL)
getfacl(1)
getfattrflag(1TSOL)
getfpriv(1TSOL)
getfsattr(1MTSOL)
getlabel(1TSOL)
getmldadorn(1TSOL)
getsldname(1TSOL)
hextoalabel(1MTSOL)
ipcrm(1TSOL)
ipcs(1TSOL)
list_devices(1MTSOL)
mldpwd(1TSOL)
mldrealpath(1TSOL)
netstat(1MTSOL)
newsecfs(1MTSOL)
pattr(1TSOL)
pclear(1TSOL)
pfsh(1MTSOL)
plabel(1TSOL)
ppriv(1TSOL)
pprivtest(1TSOL)
route(1MTSOL)
rpc.getpeerinfod(1MTSOL)
runpd(1MTSOL)
setfacl(1)
setfattrflag(1TSOL)
setfpriv(1TSOL)
setfsattr(1MTSOL)
setlabel(1TSOL)
sysh(1MTSOL)
tar(1TSOL)
testfpriv(1TSOL)
tnchkdb(1MTSOL)
tnctl(1MTSOL)
tnd(1MTSOL)
tninfo(1MTSOL)
tokmapctl(1MTSOL)
tokmapd(1MTSOL)
uname(1TSOL)
writeaudit(1MTSOL)

To find out in which profiles the commands are located, refer to Appendix B, in *Trusted Solaris Administrator's Procedures.*

Administering Users 4

This chapter introduces you to the User Manager, the Trusted Solaris tool for administering user and role accounts. It shows how you to set up and maintain users. The chapter is divided into two parts. The first part explains how to access the User Manager and view lists of users. The second part tells you how to enter user data.

Loading and Viewing the User List

The User Manager is a graphical interface for viewing and editing user and role account information in the `tsoluser` database. The following figure illustrates the information that can be maintained through the User Manager. (In this section, the term *user* refers to both users and roles unless explicitly noted.)

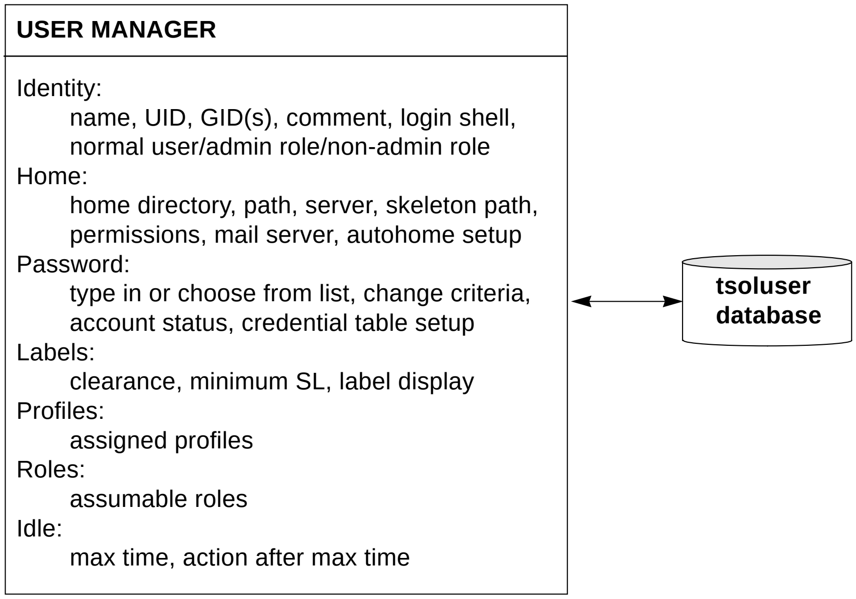
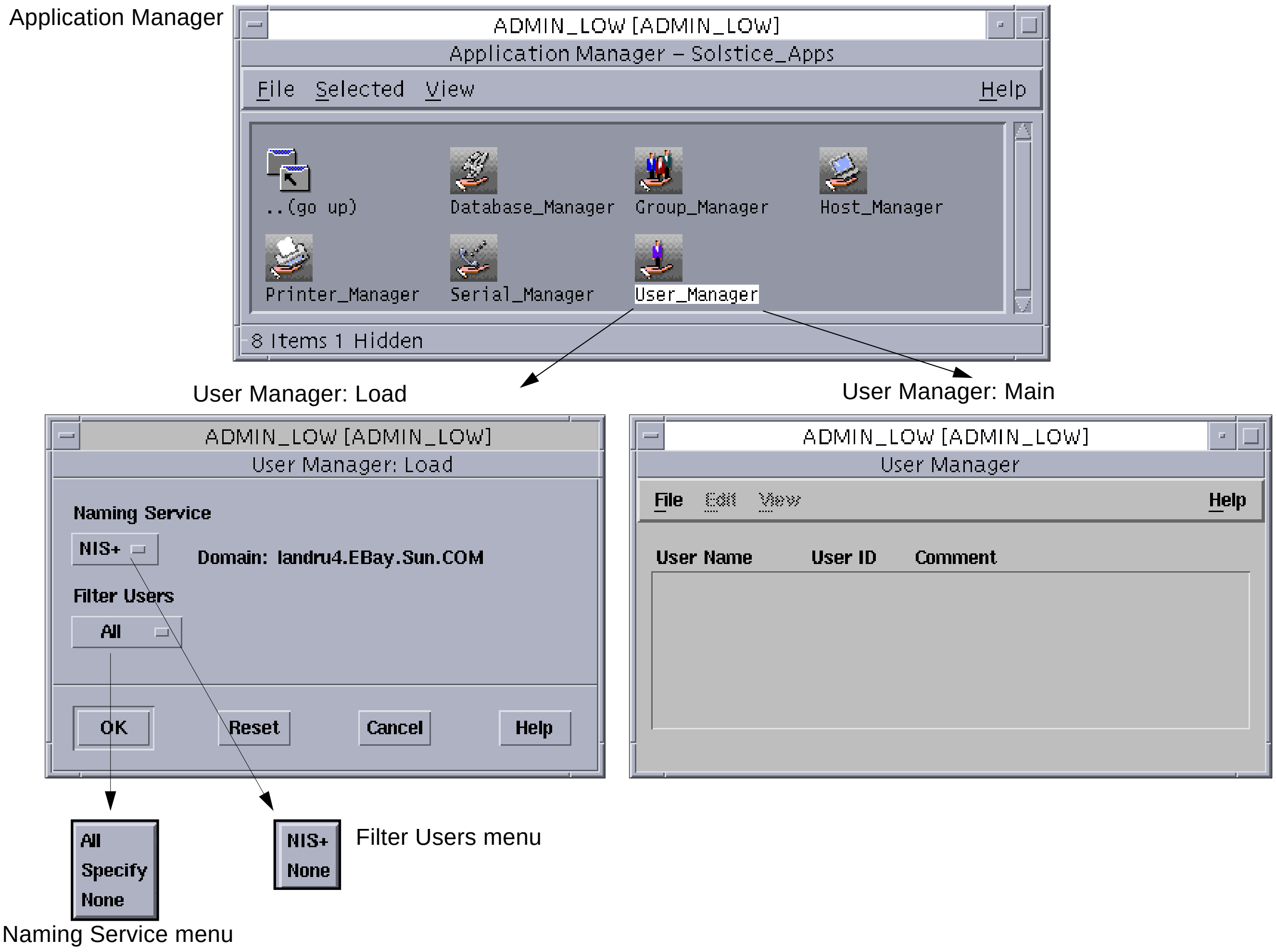

USER MANAGER
Identity:
name, UID, GID(s), comment, login shell,
normal user/admin role/non-admin role
Home:
home directory, path, server, skeleton path,
permissions, mail server, autohome setup
Password:
type in or choose from list, change criteria,
account status, credential table setup
Labels:
clearance, minimum SL, label display
Profiles:
assigned profiles
Roles:
assumable roles
Idle:
max time, action after max time
tsoluser
database

Figure 4-1 How User Information is Maintained

Launching the User Manager

To access the User Manager, you click the CDE Application Manager icon in the front panel. The User Manager icon is accessed from the Solstice_Apps folder icon in the Application Manager (see Figure 4-2). Clicking the User

Manager icon displays the User Manager: Load dialog box with the main User Manager window in an empty state (no users displayed). The User Manager: Load dialog box lets you specify a set of users to view.



Naming Service

NIS+ Domain: landru4.EBay.Sun.COM

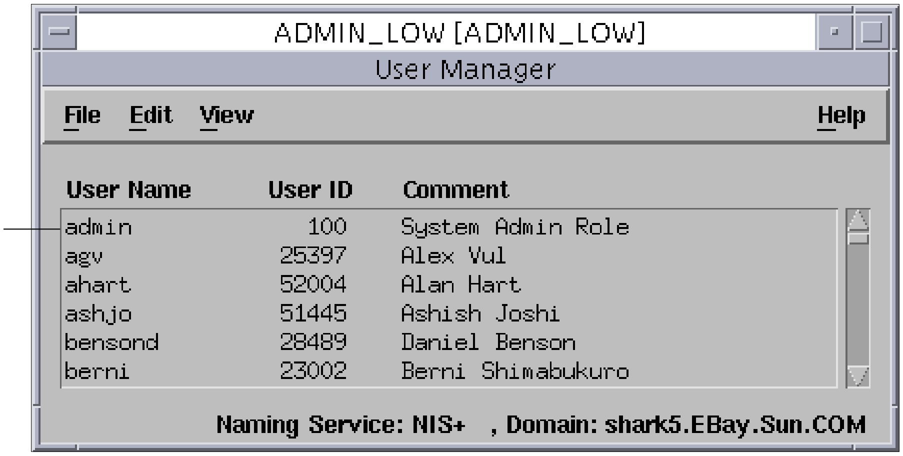
Filter Users

All

Figure 4-2 Launching the User Manager

The Main User Manager Window

The main User Manager window lets you see user and role names and their associated user IDs and comments. It lets you change how users are displayed and get access to tools for viewing and editing account data.



User Name	User ID	Comment
admin	100	System Admin Role
agv	25397	Alex Vul
ahart	52004	Alan Hart
ashjo	51445	Ashish Joshi
bensond	28489	Daniel Benson
berni	23002	Berni Shimabukuro

Naming Service: NIS+ , Domain: shark5.EBay.Sun.COM

Figure 4-3 User Manager: Main Window and Menus

The File menu lets you perform general functions, such as loading a list of users or exiting the User Manager. The Edit menu provides access to dialog boxes for editing user data (or entering it for the first time). The View menu lets you find users in the list; sort the list by user name, user ID, or comment; and rebuild the list to adjust for any new, deleted, or modified users.

Changing User Data

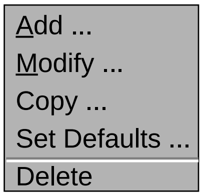
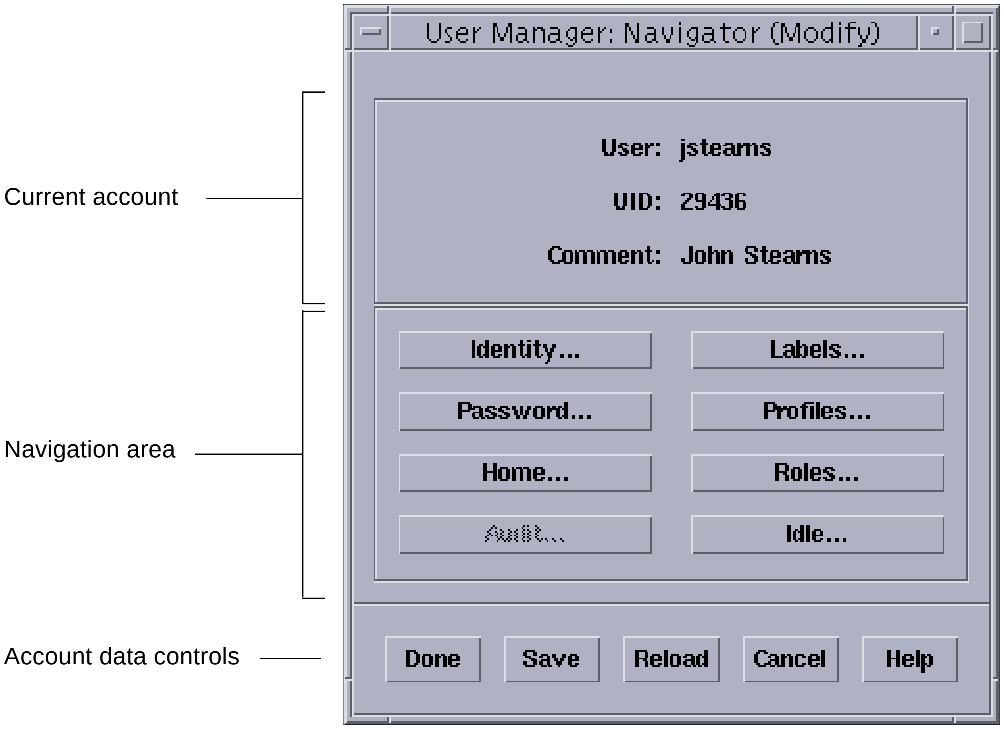


Figure 4-4 User Manager Edit

You make changes to user data through the User Manager Edit menu (see Figure 4-4). Trusted Solaris provides a family of dialog boxes for editing user data. Selecting any of the Edit menu items causes the User Manager Navigator dialog box to be displayed. The User Manager Navigator dialog box provides access to the different categories of user information.

Selecting Type of Data to Modify

The User Manager Navigator dialog box is displayed initially when you make any selection from the Edit menu (see Figure 4-5). The User Manager Navigator dialog box lets you access the dialog boxes containing the different types of user data.



User Manager: Navigator (Modify)

User: jsteams

UID: 29436

Comment: John Steams

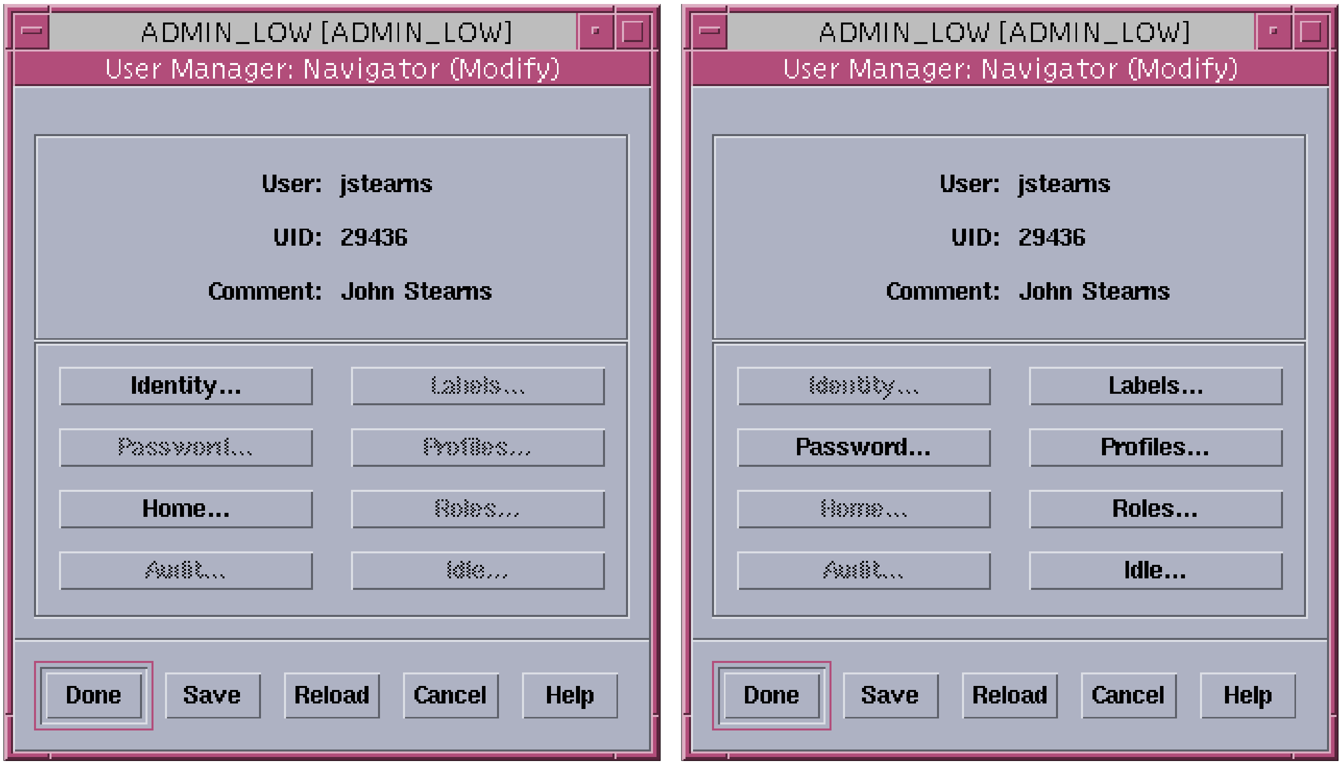
Figure 4-5 User Manager: Navigation Dialog Box

Note – The figure shows all buttons enabled (except Audit which is not yet available); normally a single role will not have the authorizations to perform these tasks. For security purposes, the responsibilities for setting up users are split by default between the security administrator, who handles the security aspects of the user's account, and the system administrator, who handles the general aspects. This helps ensure that a user cannot modify his or her own configuration in order to break the security of the system. If your security policy is less stringent, you can combine the responsibilities for setting up users into a single role. See “Alternatives to Two-Role Administration” in *Trusted Solaris Administrator’s Procedures*.

Clicking any of the buttons in the data entry navigation area displays the corresponding dialog box. The buttons are:

- **Identity** – displays the Identity dialog box for entering user identification information including login shell and user type (normal user, administrative role, or non-administrative role.
- **Password** – displays the Password dialog box for specifying password type, password change requirements, and current account state.
- **Home** – displays the Home dialog box for specifying automatic home directory creation, home directory permissions, mail server, and automounting.
- **Audit** – The Audit button is not functional in this release.
- **Labels** – displays the Labels dialog box for entering the user’s clearance and minimum sensitivity label and specifying how and if labels display.
- **Profiles** – displays the Profiles dialog box for assigning execution profiles to the user.
- **Roles** – displays the Roles dialog box for making roles available to the user.
- **Idle** – displays the Idle dialog box for specifying security measures if no operations are performed at a workstation for a set period.

Under the default configuration of roles, the system administrator has exclusive access to the Identity and Home dialog boxes; the security administrator has exclusive access to the Password, Audit, Labels, Profiles, Roles, and Idle dialog boxes (see Figure 4-6).



ADMIN_LOW [ADMIN_LOW]
User Manager: Navigator (Modify)

User: jstearns
UID: 29436
Comment: John Stearns

System administrator facilities

ADMIN_LOW [ADMIN_LOW]
User Manager: Navigator (Modify)

User: jstearns
UID: 29436
Comment: John Stearns

Security administrator facilities

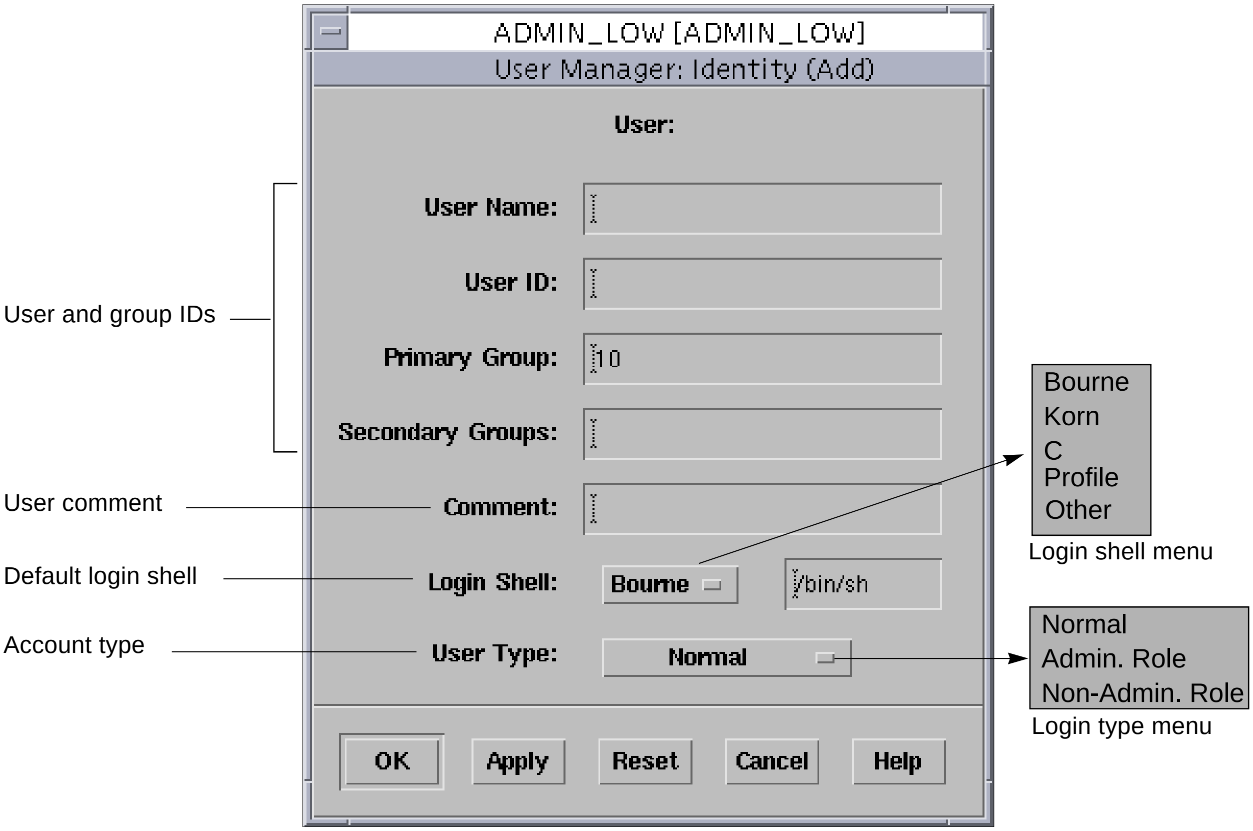
Figure 4-6 User Manager Facilities for the Security and System Administrators

Each dialog box registers its data as part of the current account record. Use the Done or Save buttons to actually save the record (or partial record). If you use Done or Save and no password information has been saved, the account will stay locked.

Editing Account Identification Information

The Identity dialog box (see Figure 4-7) lets you specify

- user and group IDs
- user comment
- default login shell
- type of account



ADMIN_LOW [ADMIN_LOW]

User Manager: Identity (Add)

User:

User Name:

User ID:

Primary Group: 10

Secondary Groups:

Comment:

Login Shell: Bourne /bin/sh

User Type: Normal

Figure 4-7 User Manager: User Identity Dialog Box

User and Group IDs

The Identity dialog box lets you edit the user’s (login) name, user ID, primary group, and secondary groups. These items are associated with every process that acts on behalf of the user and with any files or directories that the user creates. This identification information is also used in discretionary access control (DAC) to determine if the user can access files and directories created by other users. The user name is requested at login as part of the identification and authentication process. The UID is also used to identify the user for auditing purposes.

You must select a unique user name and UID when creating a new local or network user. Before creating the new user, check the user name and UIDs of all the current and deleted users on the network. (For auditing purposes, you never reuse a user name or UID from a deleted user).

Warning – Never create a user with the same name or UID as an existing role account; that user will not be able to log in.

User Comment

The User Identity dialog box also lets you enter a comment for the user. Comments contain such items as the user's real name, job title, telephone number, or in informal organizations, a humorous pseudonym. The comment appears in user lists in the main User Manager window where it can be used as a key to sort the list. The comment also displays in the From: line when the user sends email and when the `finger` command is invoked with the user as the argument.

Login Shell

The Login Shell menu lets you enter the user's type of login shell: Profile, Bourne, Korn, C, or another type that you specify. A *profile shell* is a special version of the Bourne shell that gives users and roles access to the commands and privileges specified in their assigned profiles (see "Understanding Execution Profiles" on page 21). A profile shell can be used to *enable* users, that is, give them access to commands and privileges not available to normal users, or to *restrict* users, that is, to limit them to a specific set of commands. Profile shells are required when you are setting up role accounts for users with profiles containing privileges. The other shells give users access to any commands on the system but without privileges.

Account Type

The User Type menu lets you specify the type of user account being created: normal user, administrative role, or non-administrative role. The main reason to create a new role is to define an explicit job responsibility that requires special actions, commands, privileges, and/or authorizations and that needs to be isolated from normal users (see "Understanding Roles" on page 25).

In general, your administrative needs should be satisfied by the predefined administrative roles (security administrator, system administrator, system operator, and root) supplied with Trusted Solaris, which can be modified if needed. If however you need to group administrative tasks differently, as in combining predefined roles into a superset role or defining a narrow set of tasks, then you have to create a new administrative role. Administrative roles are assigned to sysadmin group 14, are privileged NIS+ principals, and contain the *Trusted Path Attribute*, which is required for running most administrative applications. (Note that you can change the new role's group and NIS+ status if you need to.)

Create a non-administrative role when you wish to set up a non-security-related job responsibility where shared ownership of directories and files is useful. As mentioned earlier, non-administrative roles are well suited to tasks requiring rotating ownership.

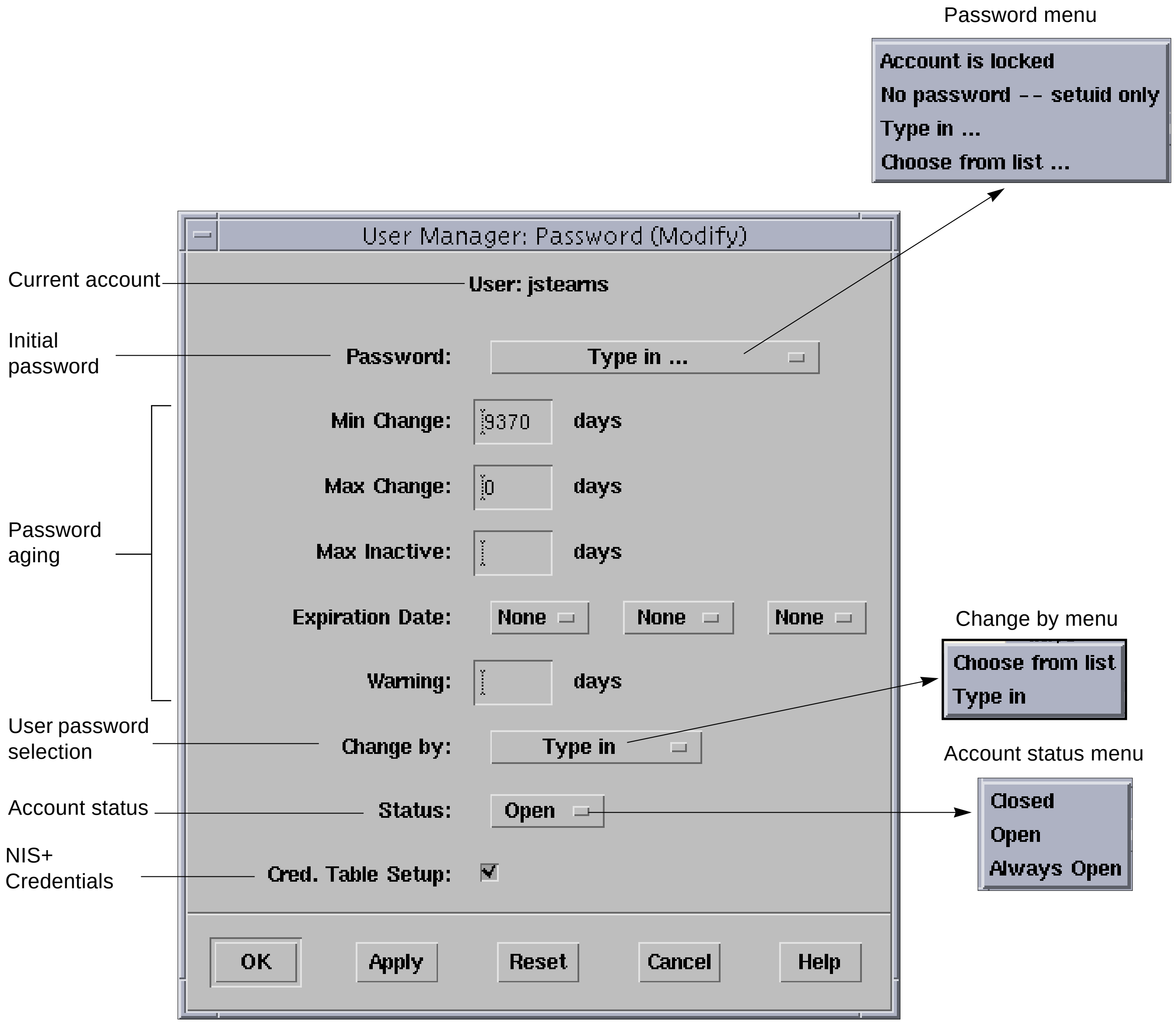
Note – Role accounts have their own mailboxes just like user accounts.

Warning – Never create a role with the same name or UID as an existing user; that user will not be able to log in.

Specifying Password Information

The Password dialog box (see Figure 4-8) is displayed when you click the Password button in the Edit Navigation dialog box. The current account is displayed read-only at the top of the dialog box. The password dialog box lets you specify

- initial password
- password aging
- password selection method
- account state
- NIS+ credential table update



User Manager: Password (Modify)

User: jstearns

Password: Type in ...

Min Change: 9370 days

Max Change: 0 days

Max Inactive: days

Expiration Date: None None None

Warning: days

Change by: Type in

Status: Open

Cred. Table Setup: ☑

Figure 4-8 User Manager: Password Dialog Box

Initial Password Creation

The Password menu lets you set the user's initial password (see Figure 4-9). The Password menu provides the following options:

Note – For security reasons, only the Type in and Choose from list menu items are recommended for user and role accounts.

- **Account is locked** – bars the user from accessing the account
- **No password -- setuid only** – for specialized system accounts, such as lp or uucp, that can be accessed through the su command but cannot be logged into directly. These accounts use the same UID regardless of user.
- **Type in ...** – lets you enter the user's initial password directly (see Figure 4-9). Manually created passwords should adhere to the rules in Table 4-1. If you specify a password that breaks these rules, your site will not conform with evaluated security requirements; however, these rules will be enforced by the system for subsequent password changes.

Table 4-1 Password Rules for Manually Created Passwords

Rules for Manually Created Passwords
The number of characters in the password must be exactly 8.
The password must contain at least two alphabetic characters.
The password must contain at least one numeric or special character.
The password must differ from the user's login name and any reverse or circular shift of that login name. (For this comparison, upper case letters and lower case letters are considered to be equal.)
A new password must have at least three characters different from the old. (For this comparison, upper case letters and lower case letters are considered to be equal.)

- **Choose from list ...** – lets you select a system-generated password for the user from a list of system-generated passwords in the Password Selection dialog box (see Figure 4-9)

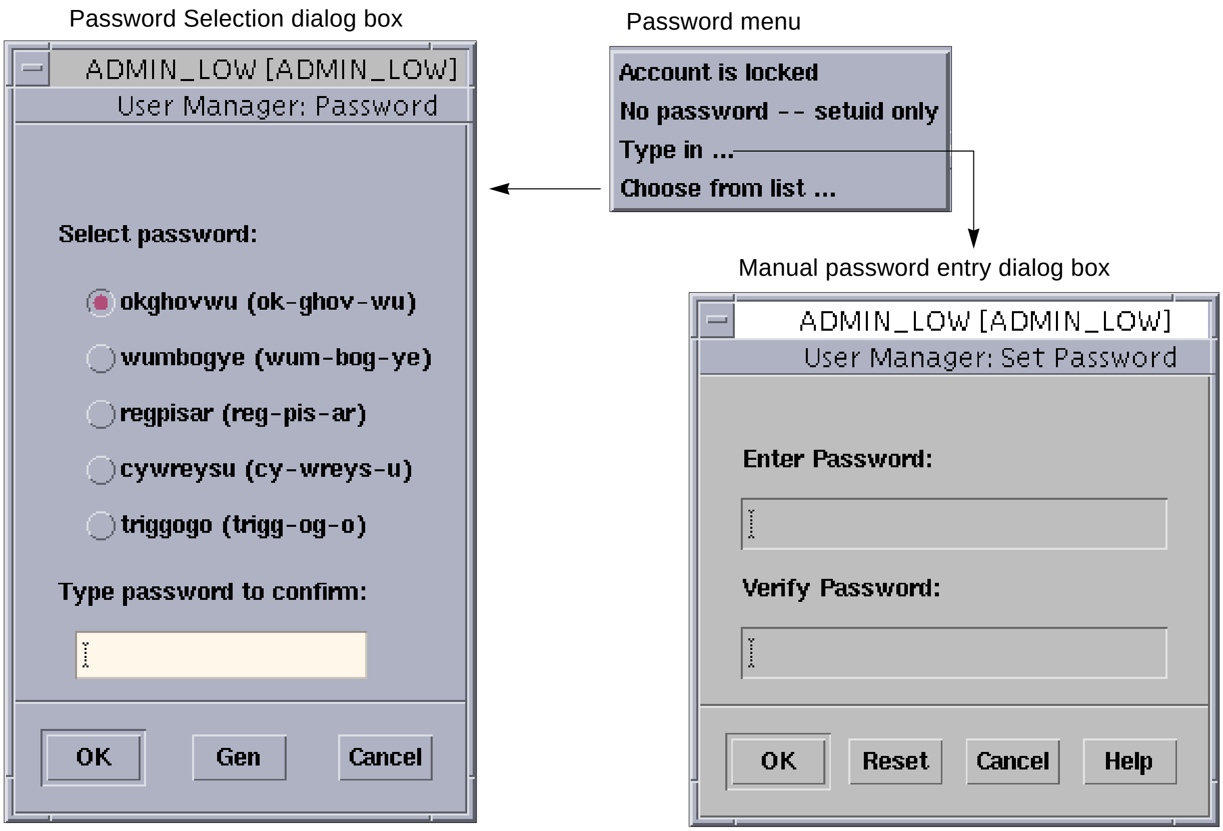


Figure 4-9 Initial Password Creation Options

The Password Selection dialog box provides you with a choice of five system-generated passwords for the user. The system-generated passwords do not use the rules in Table 4-1—they contain 8 lower-case alphabetic characters, are pronounceable, and contain no numeric or special characters. The pronunciation mnemonic shown in parentheses to the right of each password divides the password into syllables to make it easier to remember. The Gen button generates five new passwords to choose from.

Warning – Take precautions when providing users with passwords to ensure that the password is not overheard or discovered inadvertently. If there is any suspicion that the user's password has been captured by another person, change the password immediately.

Setting Up Password Aging

The next five fields in the Password dialog box are for password aging (see Figure 4-8). The password change options limit damage by intruders who have guessed or stolen passwords. The password aging options are:

- **Min Change** – sets a minimum number of days after a password change before the user can change the password on the account again. This prevents users from reverting to their old passwords.
- **Max Change** – sets the maximum number of days that a user can use the same password on an account. This forces the user to change the password periodically.
- **Max Inactive** – sets the maximum number of days that an account can be inactive before the user is locked out automatically.
- **Expiration Date** – sets the date by which the user must change the password.
- **Warning** – reminds the user to set a new password the specified number of days prior to the password expiration (by date or maximum period).

Note – When requiring a password change after a maximum period or expiration date, be sure to enable the warning message.

Setting User Password Choice

The Change By menu in the Password dialog box lets you specify how users change their passwords. If you select Type in, the user types in a new password directly into the manual password entry dialog box. If you select Choose from list, the Password Selection dialog box is displayed whenever the user changes passwords. The Password Selection dialog box provides five system-generated passwords to choose from at a time. See Figure 4-9.

Setting Account Status

The account status menu in the Password dialog box indicates the current state of the account (see Figure 4-8). Selecting an option changes the state of the account. The options are:

- **Closed** – denies the user access to the account. Use this until the account is fully specified. After a specified number of failed login attempts, an account will be closed automatically until this field is reset to Open.

Note – The security administrator can specify the number of failed logins by using `adminvi` (at ADMIN_LOW) to edit the `/etc/default/passwd` file and changing the *MAXBADLOGINS* variable, which is set to 3 by default.

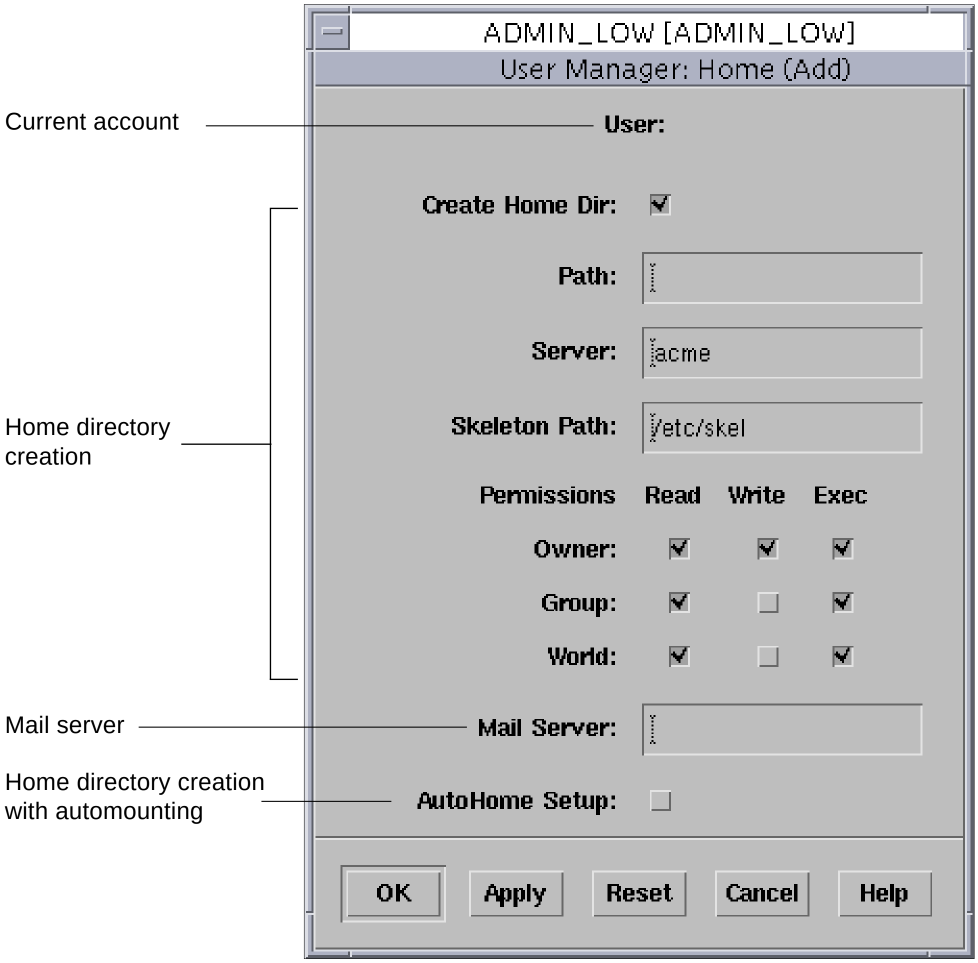
- **Open** – permits access to the account. Use this when all account information has been specified or when you need to restore access to a locked account.
- **Always Open** – permits permanent access to the account when the proper password is supplied. Use this for role accounts and any other accounts where inadvertent closing of the account would deny necessary services to users.

Updating the NIS+ Credential Table

Clicking the toggle button next to the Cred. Table Setup field adds the NIS+ principal's public and private keys to the `cred` table. This toggle should be set. See "Where Credential-Related Information Is Stored" in Chapter 5, "Administering NIS+ Credentials," of the *NIS+ and FNS Administration Guide - Solaris 2.5*.

Specifying Home Directory Information

Clicking the Home button in the Edit Navigation dialog box causes the Home Directory dialog box to be displayed (see Figure 4-10). The Home Directory dialog box lets you create the user's home directory using the User Manager.



Current account

Home directory creation

Mail server

Home directory creation with automounting

ADMIN_LOW [ADMIN_LOW]

User Manager: Home (Add)

User:

Create Home Dir: ☑

Path:

Server: acme

Skeleton Path: /etc/skel

Permissions	Read	Write	Exec
Owner:	☑	☑	☑
Group:	☑	□	☑
World:	☑	□	☑

Mail Server:

AutoHome Setup: □

Figure 4-10 User Manager: Home Directory Dialog Box

Clicking the Create Home Dir toggle indicates that the user's home directory is to be created as specified by the Path field, using the specified server and templates in the specified skeleton directory. The home directory permission toggle buttons let you specify the read, write, and execute permissions for owner, group, and world in the home directory.

Note – The server for the user's home directory must be configured prior to creation of the user account.

The Mail Server field lets you specify the user's mail server.

Clicking the AutoHome Setup toggle sets up the home directory for automounting.

Specifying Labels for Users

Clicking the Labels button in the Edit Navigation dialog box causes the Labels dialog box to be displayed (see Figure 4-11). The Labels dialog box lets you specify the user's account SL range and lets you specify how and if labels are to be displayed in the user's sessions.

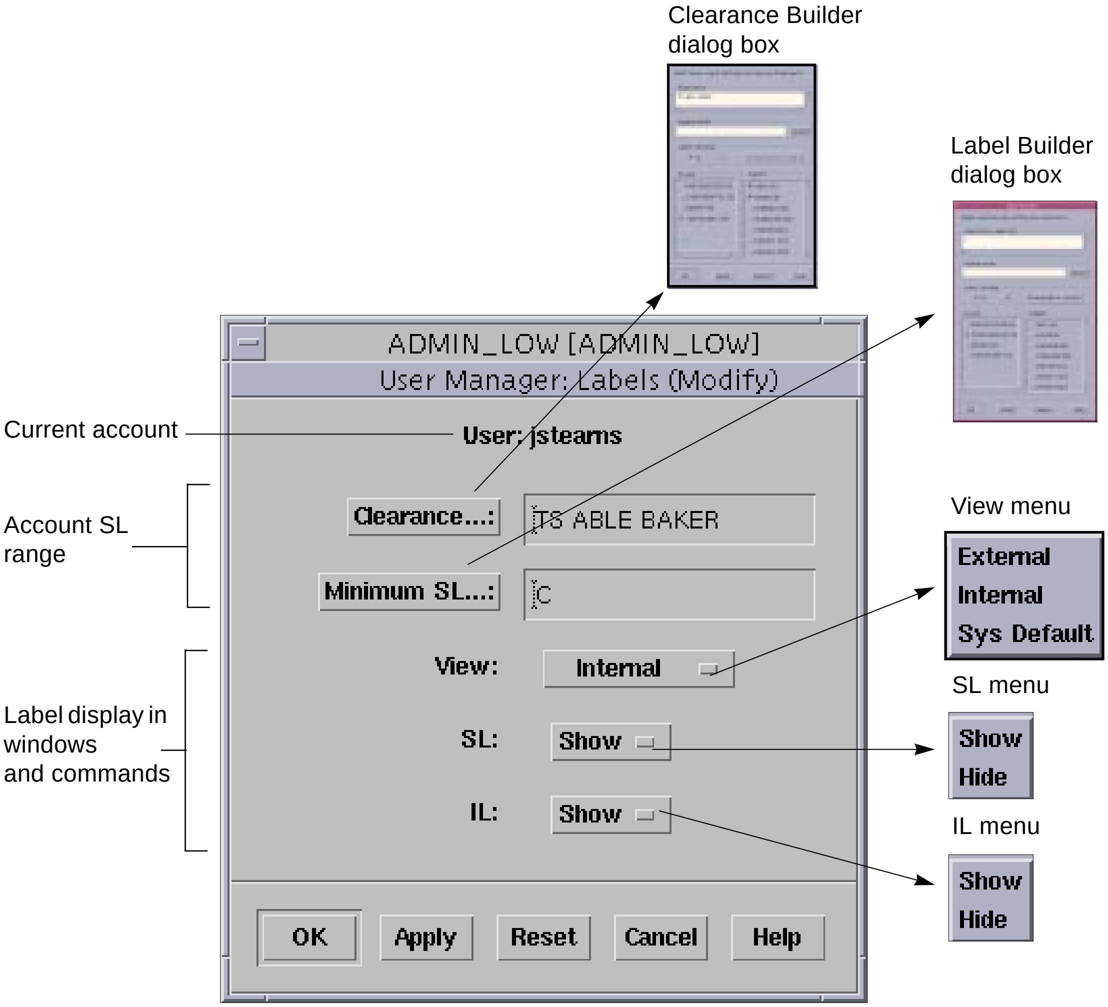


Figure 4-11 User Manager: Labels Dialog Box

Setting the User's Account SL Range

The *account SL range* is the range of sensitivity labels in which the user can operate. The top of the range is defined by the user's clearance. The bottom is defined by the user's minimum sensitivity label.

Clicking the Clearance button displays the Clearance Builder dialog box so that you can enter the user's clearance. The Clearance Builder dialog box lets you select the classification and compartment components that make up the user's clearance. When you close the Clearance Builder dialog box, the clearance you have selected appears in the Clearance field in the Labels dialog box.

Clicking the Minimum button displays the Label Builder dialog box, so that you can enter the user's minimum sensitivity label. The *minimum label* is the minimum sensitivity label in the user's account SL range. It is the default sensitivity label when the user begins a Trusted Solaris session and occupies a workspace. In similar fashion to setting the clearance, you use the Label Builder dialog box to select the classification and compartment components defining the minimum sensitivity label. When you close the Label Builder dialog box, the minimum sensitivity label appears in the Minimum field in the Labels dialog box.

Note – Both the Clearance Builder and Label Builder dialog boxes limit your choices to values within the user's account range. For more information, see "Account Label Range" on page 10 in Chapter 1, "Introduction to Administration." If some classifications are grayed out, there may be restrictions on the compartments for those particular classifications; try deselecting any selected compartments.

Displaying Labels

The lower part of the Labels dialog box lets you specify the display of sensitivity and information labels in the user session. When labels are displayed, they appear in the trusted path indicator at the bottom of the screen, at the top of window frames, and in the title bar on window icons. Sensitivity labels appear inside square brackets ([]) so that they can be distinguished from information labels. Users allowed to work at multiple levels need to have SLs displayed. Users in single-label sessions may not require SLs to be displayed.

The View menu provides these options:

- **External** – translates the ADMIN_HIGH and ADMIN_LOW sensitivity labels into label names described in the `label_encodings` file. For example, your policy may be to display the label PUBLIC instead of ADMIN_LOW.
- **Internal** – displays the ADMIN_HIGH and ADMIN_LOW sensitivity labels.
- **Sys default** – uses the system default regarding the display of the ADMIN_HIGH and ADMIN_LOW sensitivity labels, as defined in the `label_encodings` file.

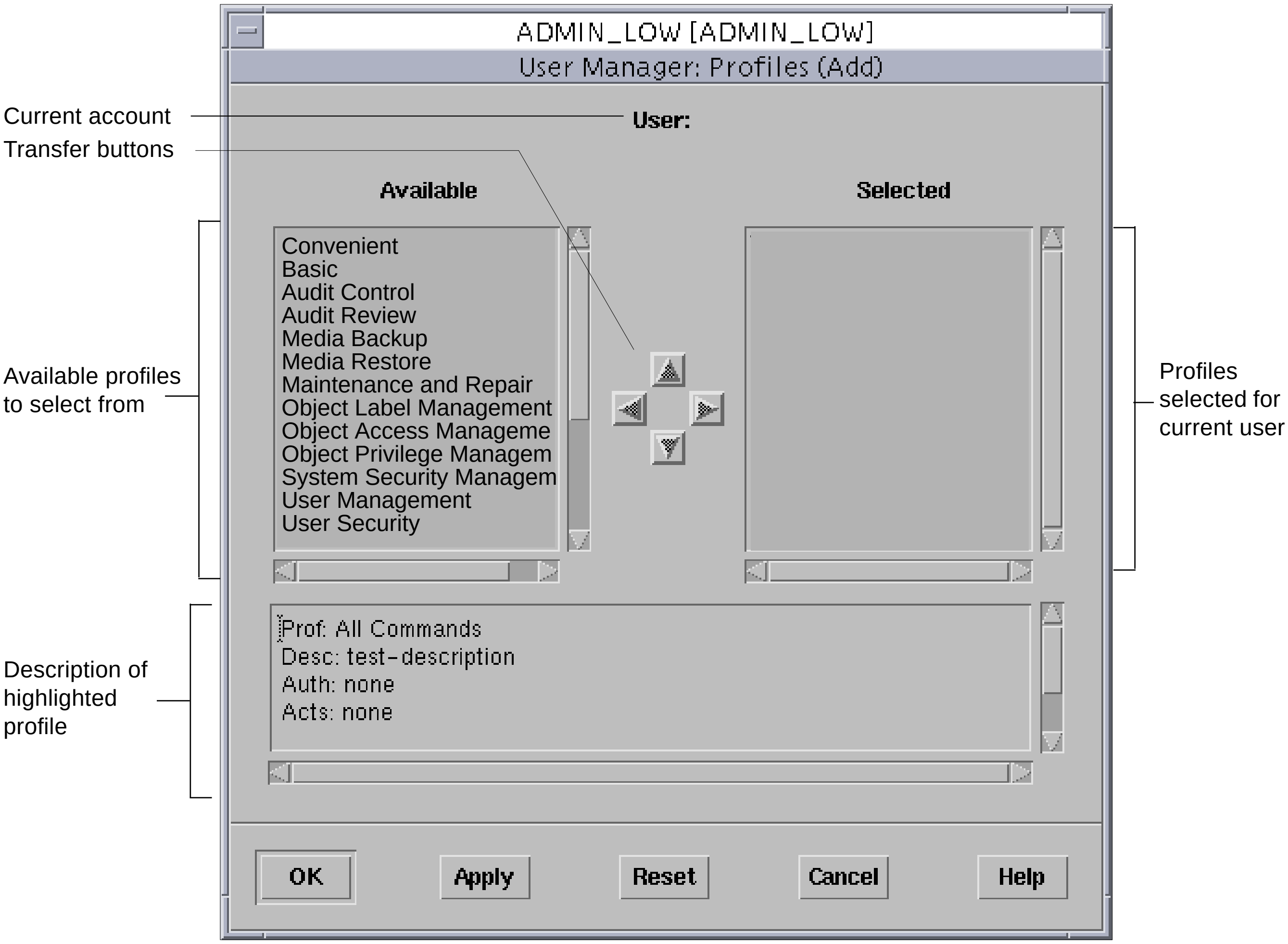
Note – The display options in the View are only operational if sensitivity labels or information labels are being displayed. See below.

The SL menu lets you specify whether sensitivity labels are displayed or hidden. The IL menu lets you specify whether information labels are displayed or hidden.

Specifying Execution Profiles for Users

Clicking the Profiles button in the Edit Navigation dialog box causes the Profiles dialog box to be displayed (see Figure 4-12). The Profiles dialog box lets you assign execution profiles to users and roles. An *execution profile* is a grouping of tools made up of CDE actions, commands, and authorizations (see “Using the Profile Manager” on page 120). A user cannot use a command in a profile shell unless that command is included in one of the profiles assigned to that user.

Execution profiles containing applications that are related to security are only assigned to roles not to users directly. Execution profiles containing applications that are not relevant to security can be assigned directly to users.



Current account

Transfer buttons

Available profiles to select from

Profiles selected for current user

Description of highlighted profile

Convenient
Basic
Audit Control
Audit Review
Media Backup
Media Restore
Maintenance and Repair
Object Label Management
Object Access Manageme
Object Privilege Managem
System Security Managem
User Management
User Security

Prof: All Commands
Desc: test-description
Auth: none
Acts: none

Figure 4-12 User Manager: Profiles Dialog Box

The list at the left of the dialog box displays the available execution profiles that have not been assigned to the user. Trusted Solaris provides a number of predefined execution profiles (see “Profiles Available in Trusted Solaris” on page 21) and also lets you create your own profiles (see “Using the Profile Manager” on page 120). The list at the right of the dialog box contains the execution profiles that have been selected for this user. If you click an execution profile, it becomes selected and its description (if there is one) is displayed in the description area at the bottom of the dialog box. Each description provides the following information:

- Prof – the name of the execution profile
- Desc – a short description of the purpose of the execution profile
- Auth – any authorizations included in the execution profile
- Acts – any actions included in the execution profile
- Cmds – any commands included in the execution profile

The left- and right-pointing transfer buttons let you move profiles between lists.

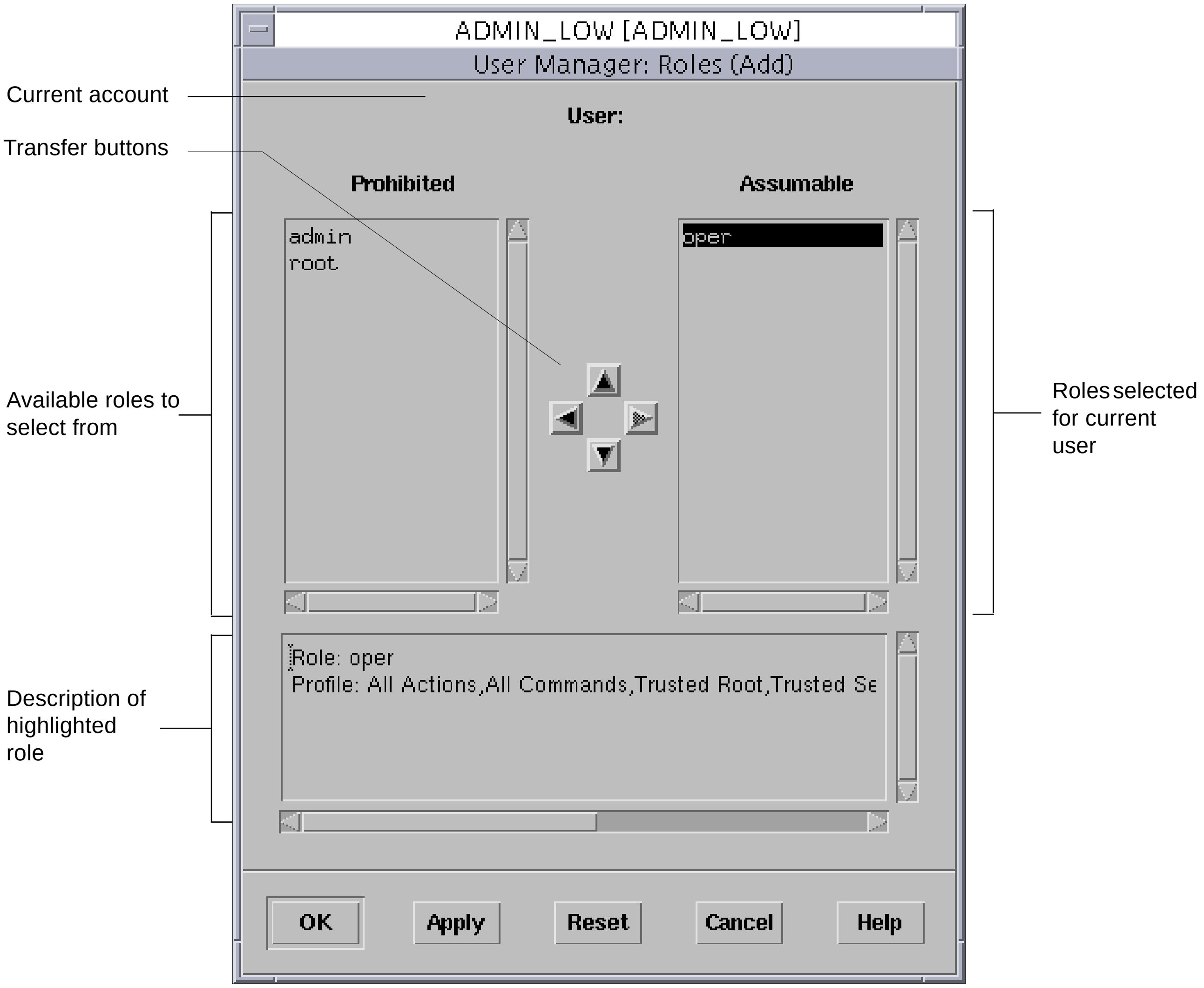
The up and down transfer buttons let you move the currently highlighted profile up or down in the selected list. The order of profiles is important because Trusted Solaris uses the order of this list when searching for commands in profiles, much the same way as the PATH variable works. Thus, if the user runs a command that appears in more than one profile, it will run as defined in the first occurrence in the profile list; be careful when working with such cases of duplicate commands.

Note – You can also use duplicate commands to your advantage. If you want to change a command's privileges, create a profile with the new privilege assignments for the command and insert that profile above the profile in which the command normally appears.

Specifying Roles for Users

Clicking the Roles button in the Edit Navigation dialog box causes the Roles dialog box to be displayed (see Figure 4-13). Note that you can only assign roles to users; you cannot assign roles to other roles. The Roles dialog box operates in similar fashion to the Profiles dialog box in terms of assigning items to a user. The list at the left of the dialog box displays the available roles that have not been assigned to the user. You assign a role by moving it to the list on the right. If you click a role, it becomes selected and its description (if there is one) is displayed in the description area at the bottom of the dialog box. The description identifies the role and any execution profiles assigned to it.

For more information on roles, see "Understanding Roles" on page 25.



ADMIN_LOW [ADMIN_LOW]

User Manager: Roles (Add)

User:

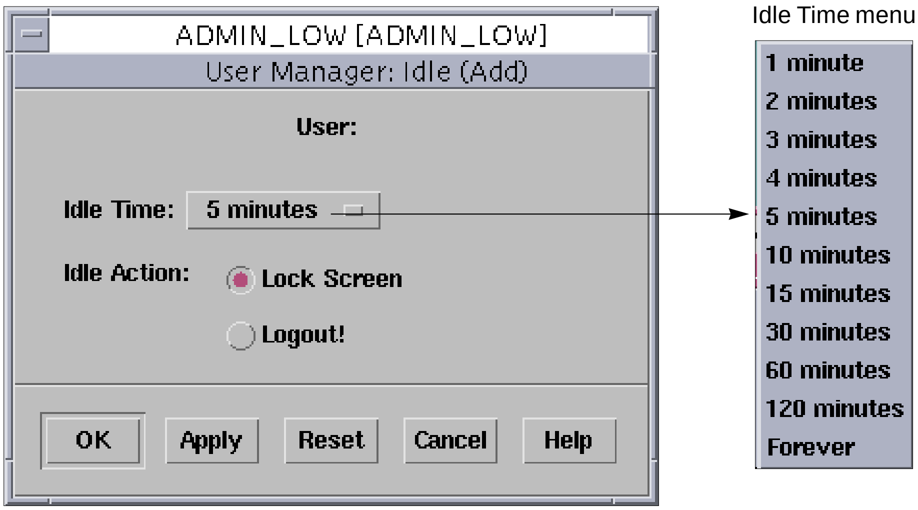
Role: oper
Profile: All Actions,All Commands,Trusted Root,Trusted Se

Current account
Transfer buttons
Prohibited
Assumable
admin
root
oper
Available roles to select from
Roles selected for current user
Description of highlighted role
OK
Apply
Reset
Cancel
Help

Figure 4-13 User Manager: Roles Dialog Box

Specifying User Idle Limits and Actions

Clicking the Idle button in the Edit Navigation dialog box causes the Idle dialog box to be displayed (see Figure 4-14). The Idle dialog box lets you specify what happens if the user performs no operation at the workstation for a set period.



ADMIN_LOW [ADMIN_LOW]

User Manager: Idle (Add)

User:

Idle Time: 5 minutes

Idle Action: ● Lock Screen ○ Logout!

Idle Time menu

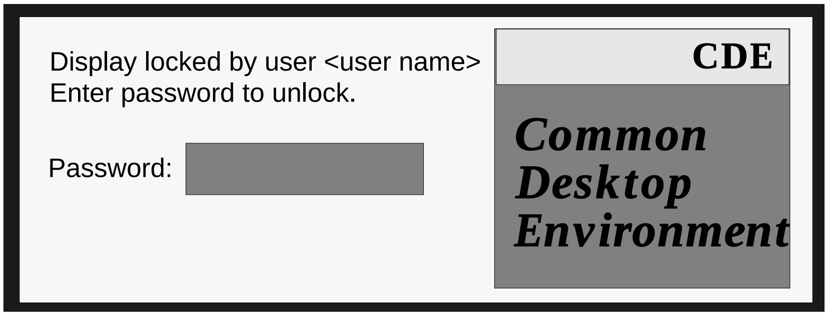
Figure 4-14 User Manager: Idle Dialog Box with Idle Time Menu

The Idle Time menu in the Idle dialog box lets you specify that either a lock screen or a logout action will be taken if the user performs no operation after 1, 2, 3, 4, 5, 10, 15, 30, 60, or 120 minutes of idleness. If you choose Forever, you are effectively disabling this feature and the session will stay up indefinitely.

The Idle Action field provides two options:

- Lock screen – locks the screen after the specified period of idleness has passed. The user must then supply a password to regain access to the session. Moving the mouse or pressing a key causes the dialog box shown in Figure 4-15 to display so that the user can enter the password.
- Logout – logs the user out of the system entirely when the specified period of idleness has passed. The user must log in again to regain access.

Caution – When you force a logout, processes running in the user's session are killed and may terminate abnormally.



Display locked by user <user name>
Enter password to unlock.

Password:

Figure 4-15 Lock-out Password Dialog Box

Deleting Users and Groups

Use the Edit menu in the main User Manager window to delete users from the system. You select the user's name from the list and then click the Delete selection from the menu. Before you delete a user from the system, you must ensure that the user's home directory and any objects owned by that user are also deleted. As an alternative to deleting objects owned by the user, you can change the ownership of these objects to a different user on the system. You must also ensure that all batch jobs associated with the deleted user are deleted as well and that there are no objects or processes that belonged to the deleted user remaining on the system. For auditing purposes, you should make sure that the user name and UID are never reused.

Administering Trusted Networking 5

This chapter describes networking in the Trusted Solaris environment. The Trusted Solaris 2.5 networking subsystem is an extended version of the Solaris 2.5.1 TCP/IP network. The extensions enable communication between workstations on the network in a trusted fashion. The networking subsystem helps ensure that the system's security policy (e.g., MAC, information label floating) is preserved across distributed applications. The amount of administration and protection required for your network depends on whether it is homogeneous or heterogeneous.

Note – In the default configuration, the security administrator role is responsible for network security.

Overview of Trusted Solaris Networking

This section covers the following networking topics:

- Homogeneous networks
- Heterogeneous networks
- Host types
- Network configuration databases
- Related subsystems
- How data is transmitted

Homogeneous Networks

A homogeneous network configuration is the easiest to administer and protect. In a *homogeneous network configuration*, all workstations run the Trusted Solaris 2.5 operating environment and use the same NIS+ master server with the same set of security attributes (sensitivity labels, information labels, etc.). A typical homogeneous network, served by a NIS+ master, is shown in Figure 5-1. The hosts in a homogeneous network are said to be in the same *security domain*.

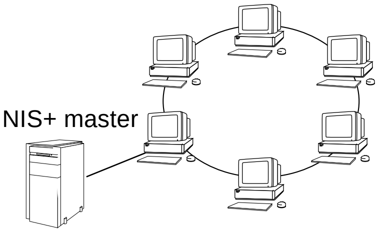


Figure 5-1 Homogeneous Network

Workstations are connected to networks by a physical connector called a *network interface*. Each network interface has an accreditation range, consisting of a maximum sensitivity label setting the upper boundary and a minimum sensitivity label for the lower boundary. The accreditation range controls the sensitivity of the information that can be transmitted or received through the interface.

Heterogeneous Networks

Trusted Solaris networks can also accommodate hosts running different network protocols. A heterogeneous configuration requires more protection than a homogeneous arrangement; you need to specify how data from hosts with different protocols will be treated with regard to security policy. Figure 5-2 shows a typical heterogeneous network and some different protocols with which a Trusted Solaris network can communicate.

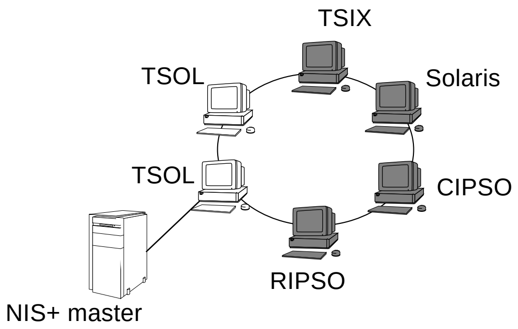


Figure 5-2 Heterogeneous Network

Host Types

To understand how Trusted Solaris workstations accept data from other Trusted Solaris workstations and hosts using other data protocols, it is useful to compare the standard Solaris data packet format (see Figure 5-3(a)) with the Trusted Solaris format (see Figure 5-3(b)).

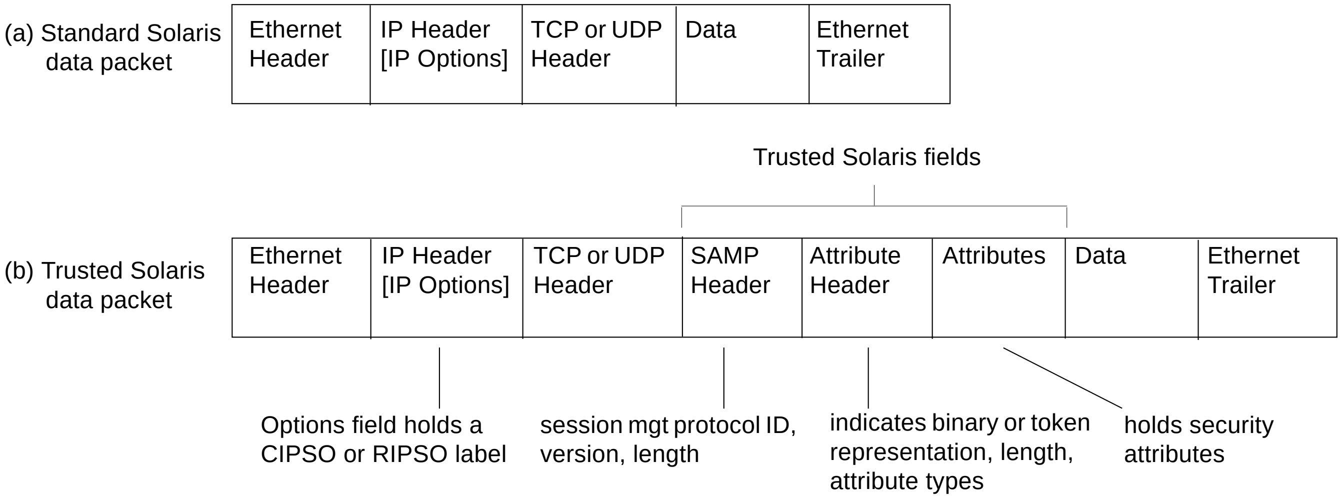

(a) Standard Solaris data packet
Ethernet Header
IP Header [IP Options]
TCP or UDP Header
Data
Ethernet Trailer
Trusted Solaris fields
(b) Trusted Solaris data packet
Ethernet Header
IP Header [IP Options]
TCP or UDP Header
SAMP Header
Attribute Header
Attributes
Data
Ethernet Trailer
Options field holds a CIPSO or RIPSO label
session mgt protocol ID, version, length
indicates binary or token representation, length, attribute types
holds security attributes

Figure 5-3 Comparison of Data Packet Formats

In the standard format, there are three headers, a data area, and a trailer; these are also in the Trusted Solaris format. The differences from the standard format are that the Trusted Solaris format

- Uses the IP Options field to hold a RIPSO (Revised Internet Protocol Security Option) or a CIPSO (Common Internet Protocol Security Option) label. There are two types of CIPSO options supported; tag type 1 for CIPSO hosts and tag type 3 for MSIX hosts.
- Includes a SAMP (Security Attribute Modulation Protocol) header identifying the session management protocol and version.
- Includes an attribute header indicating whether the attribute types are sent in binary or token form. Trusted Solaris uses binary representation only, but can accept data from protocols that use tokens.
- Includes security attributes.

Trusted Solaris classifies host types according to the networking protocols so that it can transmit data correctly. Trusted Solaris classifies host types as follows:

- *sun_tsol* – refers to workstations running Trusted Solaris 2.5. It uses binary representation for security attributes in the protocol. Trusted Solaris hosts can receive or pass on data with RIPSO or CIPSO IP options.
- *unlabeled* – refers to hosts that do not recognize security attributes.
- *tsix* – refers to hosts supporting the TSIX (RE) 1.1 (Trusted Systems Information eXchange for Restricted Environments standard). It uses the same format as Trusted Solaris hosts (see Figure 5-3) except that it uses tokens (arbitrary 32-bit numbers) rather than binary data to represent security attributes. The tokens use the security attribute token mapping protocol (SATMP).
- *msix* – refers to hosts supporting the MSIX 1.0 standard, which is used in Trusted Solaris 1.x networks.
- *cipso* – refers to hosts conforming to CIPSO. The only security attribute supported under CIPSO is the CIPSO DOI (domain of interpretation).
- *ripso* – refers to hosts conforming to RIPSO, as described in the IETF RFC 1108. Trusted Solaris 2.5 supports an administratively-set fixed RIPSO label to be applied to network packets sent to the particular host. Although this functionality does not fully meet the RFC specifications, it supplies sufficient functionality where RIPSO labels are needed.

Note – The tsix, msix, cipso, and ripso host types lie in the category of hosts running other trusted operating environments. The unlabeled host types is for those hosts that use the standard networking protocol and do not use security attributes.

When you configure the network configuration databases for your site, you specify all hosts with which workstations on your network can communicate. You set up templates with default security attribute values, categorized by the above host types. This is explained in the following section.

Network Configuration Databases

To accomplish external communication, you set up databases containing host, network interface, and default security attribute information. There are three network configuration databases for this:

- `tnrhdb`
- `tnrhtp`
- `tnidb`

These databases are loaded into the kernel and are used in accreditation checks as data is transmitted from one host to another. These databases are maintained using the Database Manager. Trusted Solaris uses NIS+ for central management of the `tnrhdb` and `tnrhtp` databases; the `tnidb` database is maintained separately on each host. Only the security administrator or possibly root can administer the network databases. To access the Database Manager, click the CDE Application Manager icon in the front panel. The Database Manager icon is accessed from the Solstice_Apps folder icon in the Application Manager. Clicking the Database Manager icon displays the Database Manager: Load dialog box with a scroll list of databases to select from.

The tnrhdb Database

The `tnrhdb(4TSOL)` database holds the IP addresses of all hosts permitted to communicate with workstations in the network and the templates (from `tnrhtp`) assigned to them. The database also can hold default values as part of a fallback mechanism (see Figure 5-4); substituting 0 in the rightmost byte(s) of the IP address serves as a wildcard for unlisted hosts with IP addresses that match the non-zero portion of the default. Note that the fallback mechanism does not apply to subnet masks.

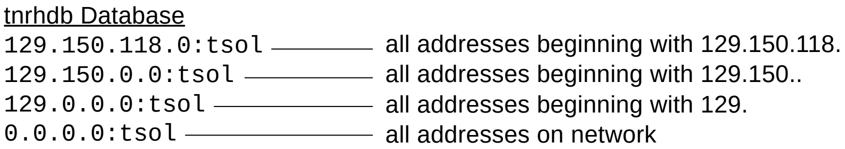
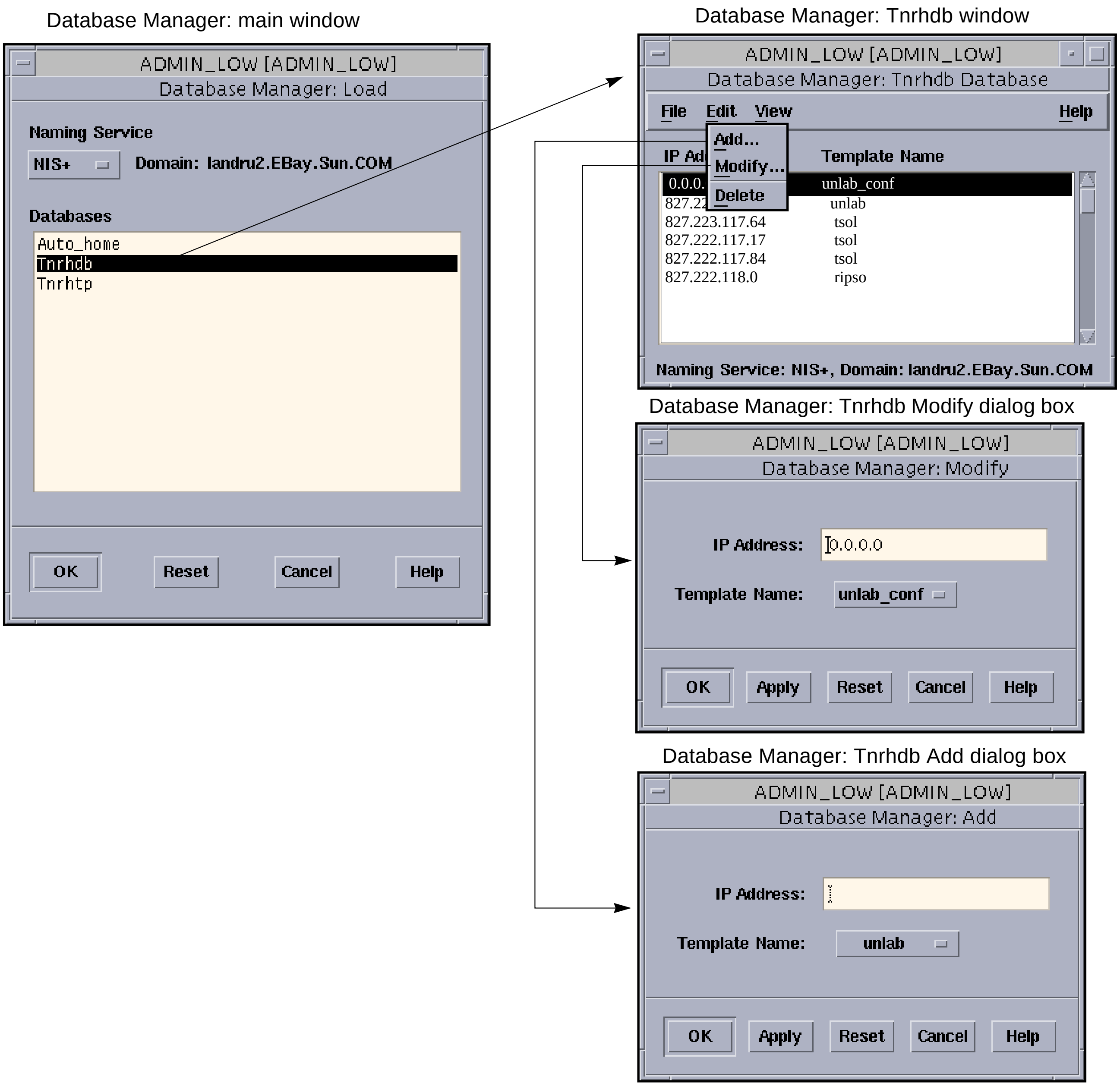

tnrhdb Database
129.150.118.0:tsol
all addresses beginning with 129.150.118.
129.150.0.0:tsol
all addresses beginning with 129.150..
129.0.0.0:tsol
all addresses beginning with 129.
0.0.0.0:tsol
all addresses on network

Figure 5-4 IP Address Fallback Mechanism

When you select the tnrhdb database from the Database Manager and load the database, the contents of the tnrhdb database are displayed in the main Database Manager window, showing IP addresses representing remote hosts and the templates applied to communications with that particular host. To edit the tnrhdb database, choose Add or select an IP address and choose Modify from the Edit menu; the appropriate dialog box displays. Figure 5-5 shows the Database Manager: Load window, the main window with tnrhdb database and the two dialog boxes available from the Edit menu.



Database Manager: main window

Database Manager: Tnrhdb window

Database Manager: Tnrhdb Modify dialog box

Database Manager: Tnrhdb Add dialog box

Naming Service: NIS+

Domain: landru2.EBay.Sun.COM

Databases

- Auto_home
- Tnrhdb
- Tnrhtp

IP Ad	Template Name
0.0.0.	unlab_conf
827.22	unlab
827.223.117.64	tsol
827.222.117.17	tsol
827.222.117.84	tsol
827.222.118.0	ripso

Naming Service: NIS+, Domain: landru2.EBay.Sun.COM

IP Address: 0.0.0.0

Template Name: unlab_conf

IP Address:

Template Name: unlab

Figure 5-5 Database Manager Windows for `tnrhdb`

The tnrhtp Database

The `tnrhtp(4TSOL)` database holds templates containing security attribute values to be assigned to source hosts. In a homogeneous network, only one template is needed; in a heterogeneous network, you need a separate template for each type of host. These attributes serve as defaults for missing attributes from incoming data. They also provide destination information for outgoing data and are use in accreditation checks for incoming packets. The relevant security attributes depend on the host type specified for the template. The security attributes that can be stored in `tnrhtp` are:

- sensitivity label
- clearance
- information label
- UID
- GID
- allowed privileges
- forced privileges
- minimum SL and maximum SL – defining the accreditation range
- IP label – identifies type of IP label: RIPSO, CIPSO, or none
- RIPSO label
- RIPSO error – protection authority flags used when ICMP error messages contain a RIPSO label
- CIPSO DOI – identifies the host's Domain of Interpretation (DOI) for CIPSO labeled packets
- audit UID
- audit mask
- audit terminal ID
- audit session ID

If the ip_label field in a template is set to *cipso*, or if the remote host type is *cipso*, then tag type 1 is used. Tag type 3 is used when the remote host type is MSIX. However, each type of attribute is only appropriate for certain host types. Table 5-1 shows which security attributes are permitted with which host types.

Table 5-1 Security Attributes by Host Type

Host Type	Security Attributes
unlabeled	sensitivity label, information label, clearance, UID, GID, forced privileges, audit UID, audit mask, audit terminal ID, audit session ID, (minimum SL and maximum SL for gateway hosts)
sun_tsol	allowed privileges, minimum SL and maximum SL, IP label, RIPSO label, RIPSO error, CIPSO DOI, audit UID, audit mask, audit terminal ID, audit session ID
ripso	sensitivity label, information label, clearance, UID, GID, forced privileges, RIPSO label, RIPSO error, audit UID, audit mask, audit terminal ID, audit session ID, (minimum SL and maximum SL for gateway hosts)
cipso	clearance, information label, UID, GID, forced privileges, minimum SL and maximum SL, CIPSO DOI, audit UID, audit mask, audit terminal ID, audit session ID
tsix	sensitivity label, information label, clearance, UID, GID, allowed privileges, forced privileges, minimum SL and maximum SL, IP label, RIPSO label, RIPSO error, CIPSO DOI, audit UID, audit mask, audit terminal ID, audit session ID
msix	sensitivity label, information label, clearance, UID, GID, minimum SL and maximum SL, audit UID, audit mask, audit terminal ID, audit session ID

When you select the `tnrhtp` database from the Database Manager and load the database, the contents of the `tnrhtp` database are displayed in the main Database Manager window, showing each remote host template name and the defaults associated with it. To edit the `tnrhtp` database, choose Add or select a template and choose Modify from the Edit menu. The dialog box shown in Figure 5-6 displays.

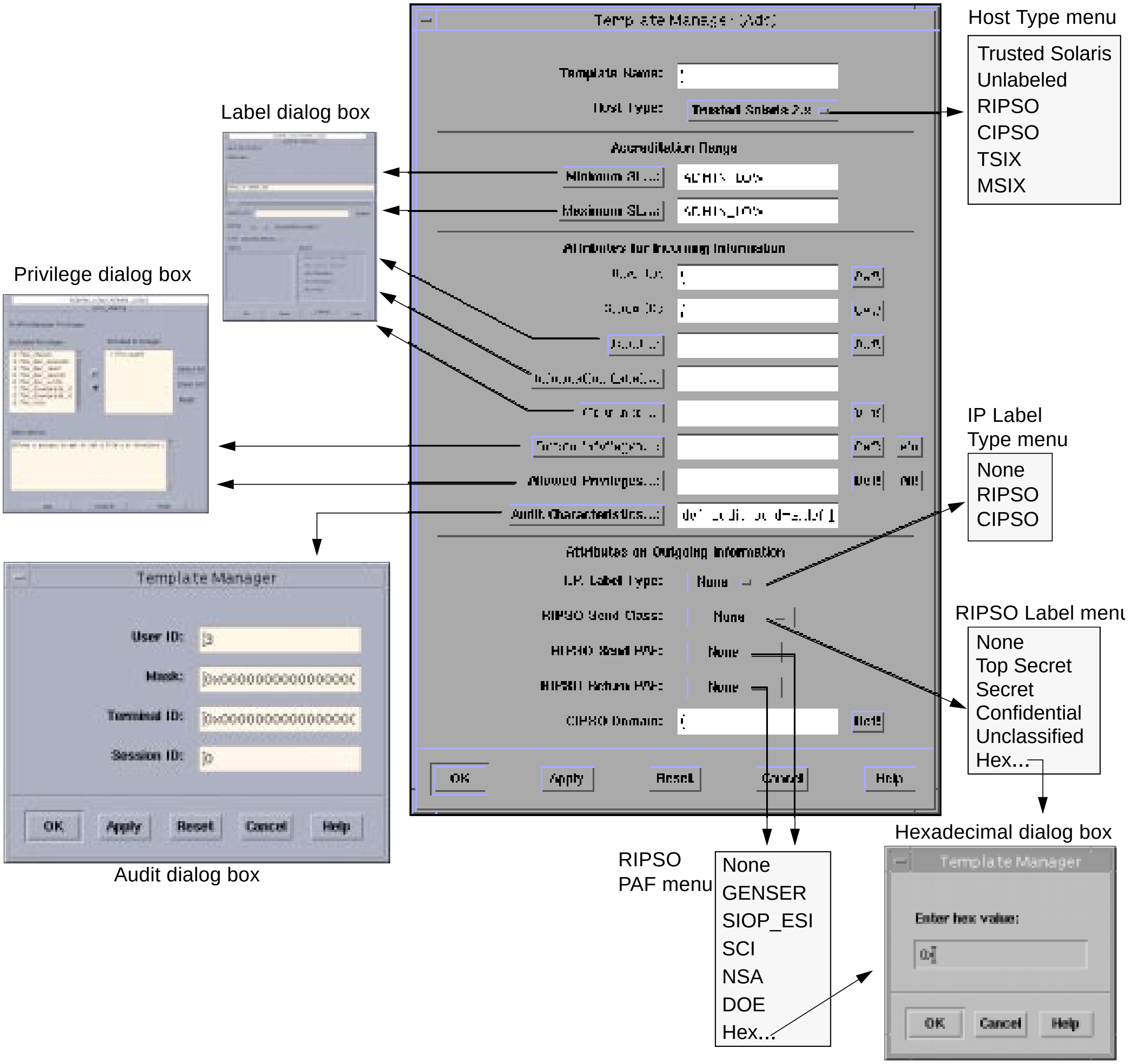


Figure 5-6 Database Manager Dialog for Adding Remote Host Templates (`tnrhtp`)

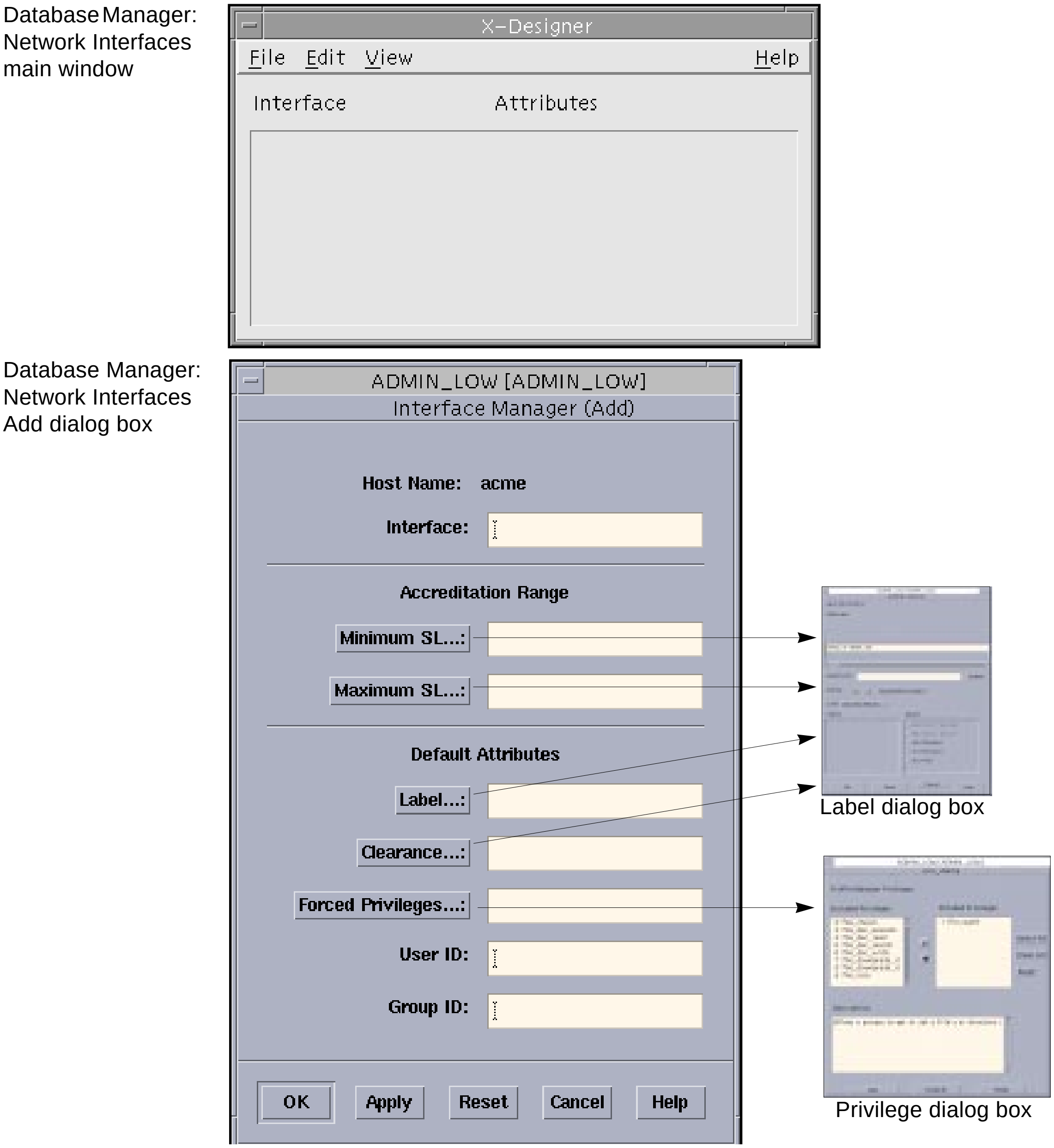
The dialog box is divided into four parts:

- **Template name and host type** – lets you identify the template. The host type menu lets you select the host type for the template.
- **Accreditation range** – the Minimum SL and Maximum SL fields let you establish the accreditation range for the template. These fields, like other fields containing sensitivity labels, information labels, or clearances, provide buttons that display label builder dialog boxes.
- **Attributes for incoming information** – lets you set values for the user ID, sensitivity label, information label, clearance, and forced and allowed privileges that can be applied to incoming information. The Forced Privileges and Allowed Privileges buttons cause a privilege selection dialog box to be displayed.
- **Attributes on outgoing information** – let you set values for the IP label type, RIPSO Send Class, RIPSO Send PAF, RIPSO Return PAF, and CIPSO domain that can be applied to outgoing information. The IP Label Type field has an option menu that lets you select none, RIPSO, or CIPSO. If this field is set to CIPSO (or if the host type is CIPSO), then CIPSO tag type 1 is used in the IP Options field in the data packet; if the host type is MSIX, CIPSO tag type 2 is used. The RIPSO Send Class field option menu lets you select the classification portion of the RIPSO label to be sent: none, Top Secret, Secret, Confidential, Unclassified, or Hex, which displays a dialog box in which you can enter a hexadecimal value directly. The RIPSO Send PAF field lets you enter the protection authority flag portion of the RIPSO label to be sent: none, GENSER, SIOP_ESI, SCI, NSA, DOE, or Hex. The RIPSO Return PAF field let you select error flags from an option menu with the choices: none, GENSER, SIOP_ESI, SCI, NSA, DOE, and Hex.

The tnidb Database

The `tnidb(4TSOL)` database is local to each host; it contains the host's network interfaces with their accreditation ranges and default values for sensitivity labels, clearances, effective UIDs/GIDs, and forced privileges. Note that the default values in `tnrhtp` override the values in `tnidb`.

When you select the `tnidb` database from the Database Manager and load the database, the contents of the `tnidb` database are displayed in the main Database Manager window, showing network interfaces, accreditation range for the interface, and the default security attributes associated with them. To edit the tnidb database, choose Add or select a network interface and choose Modify from the Edit menu and the appropriate dialog box displays. Figure 5-7 show the main window with the Add dialog box for the tnidb database. The Minimum SL and Maximum SL buttons define the accreditation range; when clicked, they display label builder dialog boxes. The Sensitivity Label and Clearance buttons also display label builder dialog boxes. The Forced Privileges button displays a privilege selection dialog box. The User ID and Group ID fields let you specify default IDs for the network interface.



Host Name: acme

Figure 5-7 Database Manager Main Window and Add Dialog Box for Adding Network Interface Data (`tnidb`)

Related Subsystems

The trusted NFS feature of Trusted Solaris 2.5 permits mounting between Trusted Solaris hosts and the other host types. Transmitted data is protected by MAC and DAC. Any missing security attributes are supplied by the `tnrhtp` and `tnidb` databases. See “Modified Solaris Network Commands” on page 106.

Routing in Trusted Solaris

In the Trusted Solaris operating environment, routes between hosts on different networks must maintain security at each step in the transmission.

Loading Routing Information at Boot Time

When a Trusted Solaris host boots, it loads routing information so it can transmit data. If the file `/etc/tsolgateways` (which is maintained manually by the administrator) exists, then the gateways in the file serve as the host’s defaults. If `/etc/tsolgateways` does not exist, then the host uses the default routes from the file `/etc/defaultrouter,` which is also maintained manually by the administrator. If either file exists, then the host is said to use *static routing.*

If neither the `/etc/tsolgateways` nor the `/etc/defaultrouter` file exists, then the host uses *dynamic routing* and must start a special daemon, either `in.rdisc(1MTSOL)`(the network router discovery daemon) if it is available, or `in.routed(1MTSOL)`(the network routing daemon) if `in.rdisc` is not available. If the host also serves as a gateway (that is, a host that connects to two or more networks), then both `in.rdisc` and `in.routed` are started.

At boot time, the `tnrhdb` and `tnrhtp` files (which reside in the `/etc/security/tsol/boot` directory) are loaded into the kernel to enable hosts to communicate with the NIS+ master; these default values are replaced when the trusted network daemon (`tnd(1MTSOL)` starts up. By default, `/etc/security/tsol/boot/tnrhdb` contains the entry `0.0.0.0:tsol`, indicating that the network is a Trusted Solaris network.

Routing Tables in the Trusted Solaris Environment

In the Trusted Solaris environment, the main objective for routing is to find the shortest secure route between two hosts. Trusted Solaris routing tables are based on extended metrics (called emetrics). An *emetric* is a combination of a routing metric and Security Routing Information (SRI), for measuring security. The SRI can incorporate these security attributes:

- Minimum SL
- Maximum SL
- DOI
- RIPSO label
- RIPSO error
- CIPSO only
- RIPSO only
- MSIX only

This information is propagated by the routing daemon `in.routed` using the Trusted Solaris-extended Routing Information Protocol if dynamic routing is used, or if static routing is used, by manual entry using the `route` command or through the `/etc/tsolgateways` or `/etc/defaultrouter` files. The emetric for a particular route is used for accreditation checks when this route is being considered.

Not every route in the routing table must have an emetric. If a route does not have an emetric, the remote host template of its first hop gateway is used for the accreditation check instead.

Accreditation Checking

To determine the suitability of a route regarding security, Trusted Solaris runs a series of tests called *accreditation checks* on the source host, destination host, and the route's emetrics. If the emetric for a particular route is missing, the security attributes for the first-hop gateway in the route are checked. A host's security attributes are derived from information in the `tnrhdb`, `tnrhtp`, and `tnidb` files. The tests check, for example, that a data packet's sensitivity label is within the range of each host in the route. As another example, transmitted data between a Trusted Solaris host and a TSIX host that does not use any IP options is disallowed if the route uses any CIPSO, RIPSO, or MSIX hosts as gateways.

Source Accreditation Checks

The accreditation checks conducted on the source host are:

- The sensitivity label of the data being sent must be within the destination host's accreditation range.
- The sensitivity label of the data must be within the accreditation range of the emetric for the route or if the emetric is not available, first-hop gateway's security attributes.
- The sensitivity label of the data must be within the accreditation range of the source host's network interface.
- If an outgoing packet has a CIPSO label, then its DOI must match the DOI of the destination and the route's emetric (or first-hop gateway).
- In similar fashion, an outgoing packet's RIPSO label must match the RIPSO label of the destination and the route's emetric (or first-hop gateway). Alternatively, the RIPSO error can match the destination's RIPSO error, the route's emetric, or the first-hop gateway's RIPSO error.
- If the destination is an MSIX machine, then the route's emetric or the first-hop gateway must also be MSIX (or Trusted Solaris) machines.

Gateway Accreditation Checks

The accreditation checks conducted on a Trusted Solaris gateway host are:

If the next hop is an unlabeled host, then the default label of the packet must match the default label of the destination host.

If the packet has the CIPSO option, the following conditions for forwarding must be true:

- The route's emetric (or next-hop gateway) must be able to accept data in the CIPSO protocol.
- The route's emetric (or next-hop gateway) must be in the data packet's DOI.
- The DOI (from the `tnrhtp` database) for the outgoing interface must be the same as the data packet's DOI.

If the packet has the RIPSO option, the following conditions for forwarding must be true:

- The route's emetric (or next-hop gateway) must be able to accept data in the RIPSO protocol.
- The route's emetric (or next-hop gateway) must have the same RIPSO label (or RIPSO error) as the data packet's RIPSO label (or RIPSO error).

Destination Accreditation Checks

When a Trusted Solaris machine receives data, the trusted network software checks for the following:

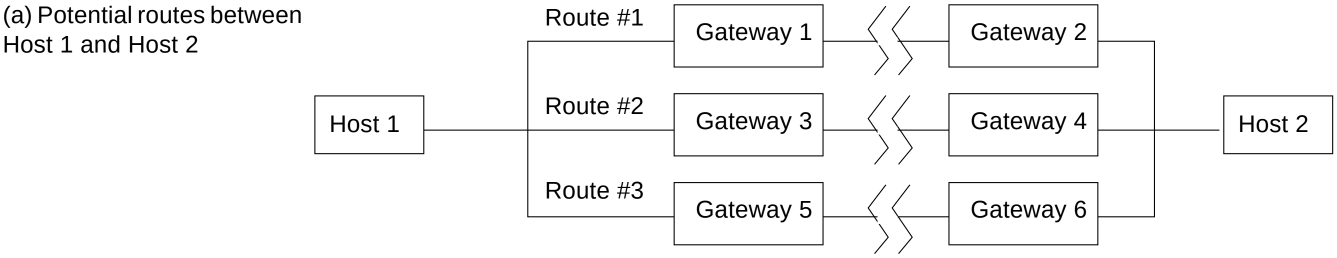
- The sensitivity label of the data is within the accreditation range of both the source machine and the network interface receiving the data.
- If a packet has a CIPSO label, then the DOI in the packet must be the same as the DOI in the remote host template for the destination.
- If a packet has a RIPSO label (or RIPSO error), then the RIPSO label (or RIPSO error) in the packet must be the same as the RIPSO label (or RIPSO error) in the remote host template for the destination.

After the data has passed the accreditation checks above, the system checks that all necessary security attributes are present. If there are missing attributes, the software looks up the source host (by its IP address or a target expression) in the `tnrhdb` database to get the name of the network security template assigned to the host. The software then retrieves the template's set of security attributes from the `tnrhtp` database. If there are still security attributes missing, the software looks up the network interface in the `tnidb` database and retrieves default security attributes. In terms of priority, the default attributes from `tnrhtp` override the attributes from `tnidb`.

Routing Example

An example of routing in the Trusted Solaris environment is shown in Figure 5-8; Figure 5-8 (a) shows the routing diagram and Figure 5-8 (b) shows the routing table. There are three potential routes between Host 1 and Host 2:

- Route #1 is the shortest with a Routing Information Protocol (RIP) metric of 3. Datagrams using route #1 are restricted to a sensitivity label range of CONFIDENTIAL (C) to SECRET (S).
- Route #2 has a larger sensitivity label range of ADMIN_LOW to ADMIN_HIGH. Datagrams using route #2 must use have an IP Option set to CIPSO.
- Route #3 has the longest distance of the three routes with an RIP of 6. Its Security Routing Information is unknown, so any security attributes must be derived from the template in `tnrhtp` for Gateway #5.



(a) Potential routes between Host 1 and Host 2

(b) Associated routing table

			Security Routing Information (SRI)							
Route	first-hop Gateway	RIP Metric	Min SL	Max SL	DOI	RIPSO Label	RIPSO Error	CIPSO Only	RIPSO Only	MSIX Only
1	Gateway 1	3	C	S						
2	Gateway 3	4	ADMIN_LOW	ADMIN_HIGH				Y		
3	Gateway 5	6								

Figure 5-8 Typical Trusted Solaris Routes and Routing Table

Using Routing Commands

To display the contents of the routing table, use the command `netstat` with the `-R` option. To make a manual change to the routing table, use the `route` command with the `add` or `delete` option. For example,

```
% route add net 129.150.115.0 129.150.118.39 -m
metric=2,min_sl=c,max_sl=ts,ripso_label=”top_secret
sci”,ripso_error=”genser;sci”
add net 129.150.115.0: gateway 129.150.118.39
```

adds to the routing table a loop with the hosts at 129.150.115.0 and 129.150.118.39 with a distance metric of 2, an SL range from C to TS, a RIPSO label = top_secret sci, and a RIPSO error = genser;sci. To see the results of the added loop, type:

```
% netstat -Rn
...
129.150.115.0        129.150.118.39        UG       0      0
        metric=2,min_sl=C,max_sl=TS,ripso_label=0x3d 0x20000000
(top_secret sci)
,ripso_error=0xa0000000 (genser;sci)
...
```

The new route is shown above. The other routes are replaced by ellipses (...). A second example of adding a route with two new emetrics and viewing the new routing table follows:

```
% route add net 129.150.114.0 129.150.118.39 -m
metric=3,min_sl=admin_low,max_sl=s,doi=3 -m
metric=4,min_sl=c,max_sl=admin_high,doi=4,ripso_label=”top_secret
sci”,ripso_error=”genser;sci”
add net 129.150.114.0: gateway 129.150.118.39
% netstat -Rn
...
129.150.115.0        129.150.118.39        UG       0      0
        metric=2,min_sl=C,max_sl=TS,ripso_label=0x3d 0x20000000
(top_secret sci)
,ripso_error=0xa0000000 (genser;sci)
129.150.114.0        129.150.118.39        UG       0      0
        metric=4,min_sl=C,max_sl=ADMIN_HIGH,doi=4,ripso_label=0x3d
0x20000000 (t
op_secret sci),ripso_error=0xa0000000 (genser;sci)
        metric=3,min_sl=ADMIN_LOW,max_sl=S,doi=3
...
```

Routing through Non-Trusted Solaris Gateways Clusters

It is possible to route secure data through clusters containing non-Trusted Solaris gateways. This procedure is called *tunneling*. For our purposes, a *cluster* is a contiguous set of either Trusted Solaris hosts and gateways only or non-Trusted Solaris hosts and gateways only. An edge gateway is a gateway (Trusted Solaris or non-Trusted Solaris) that connects a cluster to a cluster of the opposite type.

Figure 5-9 shows an example of tunneling. The shaded rectangles represent non-Trusted Solaris gateways. The loops with thick lines indicate clusters. Cluster #1 is a non-Trusted Solaris cluster; cluster #2 is a Trusted Solaris cluster.

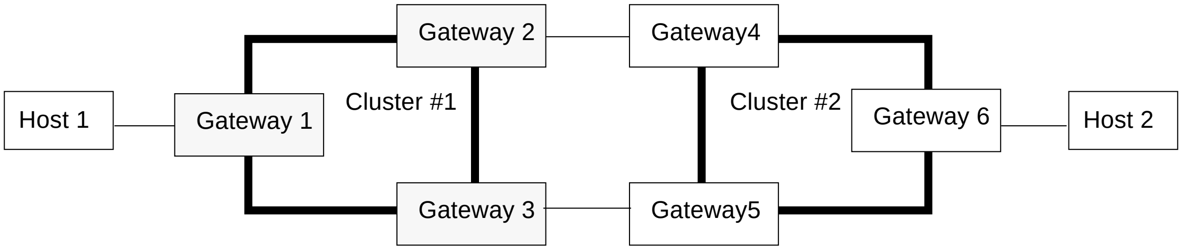


Figure 5-9 Tunneling Example

To transmit data from host #1 to host #2 requires a route through cluster #1, a non-Trusted Solaris cluster, and cluster #2, a Trusted Solaris cluster. This is permitted under these two conditions only:

- All the gateways in the non-Trusted Solaris cluster (in the example, gateways #1, #2, and #3) must have the same security attributes. At start-up, each gateway must have a local file called `/etc/security/tsol/tunnel` containing the addresses of target hosts with which it can connect.
- If there is more than one possible route and the routes enter the non-Trusted Solaris cluster through the same edge gateway and can exit from the cluster through different edge gateways, then the emetric for these routes must be equal. For example, assume that gateway #4 has an SL range of CONFIDENTIAL to SECRET and gateway #5 has a broader range of ADMIN_LOW to ADMIN_HIGH. Because gateway #1 is a non-Trusted Solaris host, it uses a standard routing table without security attributes and would be unable to distinguish between the route through gateway #4 and the route through gateway #5.

Modified Solaris Network Commands

The network commands in this section come from the base version of Solaris and have been modified to operate in the Trusted Solaris environment:

- arp
- ifconfig
- netstat
- route
- snoop
- spray
- ndd
- rdate

arp

The `arp(1MTSOL)` command lets you display and modify the Internet-to-Ethernet translation tables used by the address resolution protocol. The Trusted Solaris version of the `arp` command needs to inherit the sys_net_config privilege to run with options `-d`, `-s` and `-f`. The `-a` option must be run at ADMIN_HIGH with the effective UID 0; this restriction can be overridden by the file_mac_read and file_dac_read privileges.

ifconfig

The `ifconfig(1MTSOL)` command lets you configure network parameters and assign addresses to network interfaces. The Trusted Solaris version of the `ifconfig` command requires the sys_net_config privilege. The `ether`, `auto-revarp`, and `plumb` options need to open ADMIN_HIGH network devices that are readable by root only. These options can be invoked at ADMIN_HIGH with an effective user ID of 0; alternatively, the file_dac_read and file_mac_read privileges let you override the restrictions to these options.

ndd

The `-set` option for the `ndd(1M)` command must inherit the sys_net_config privilege to set driver parameters.

netstat

The `netstat(1MTSOL)` command displays the contents of network-related data structures (including sockets, routing tables, and other structures) in various formats. When communicating with a host on a different net, use **`netstat -rn`** to make sure that the gateway(s) are configured. The Trusted Solaris version of the `netstat` command requires a sensitivity label of ADMIN_HIGH to access kernel and network configuration information. This restriction can be overridden by the file_mac_read privilege.

The -R option lets you get the security as well as metric information for each route in the dynamic routing table. It additionally requires the net_rawaccess privilege. See “Using Routing Commands” on page 103 for examples.

rdate

The `rdate(1MTSOL)` command requires the sys_config privilege to run properly.

route

The `route(1MTSOL)` command lets you manipulate the network routing tables, including the addition and deletion of emetrics (security information). The Trusted Solaris version of the `route` command needs to inherit the sys_net_config privilege to run properly. There are three additional options in the Trusted Solaris environment:

- `-m` – specifies extended metric information on the command line
- `-e` – specifies a file containing extended metric information
- `-t` – specifies a file containing routes to be added (with simple or extended metrics)

To open the IP device for adding or deleting a route, this program must inherit the sys_net_config privilege and run at a sensitivity label of ADMIN_HIGH, and effective user ID of 0 or in the sys group. The file_mac_read privilege can override the ADMIN_HIGH MAC policy. The file_dac_read privilege can override the UID 0 or sys group DAC requirement. See “Routing in Trusted Solaris” on page 99.

snoop

The snoop(1MTSOL) command captures packets from the network and displays their contents. When opening network devices, the Trusted Solaris version of the snoop command need to run with a sensitivity label of ADMIN_HIGH and an effective UID of 0. These two requirements are not necessary if the process has the file_mac_read and file_dac_read privileges. In addition, snoop needs to inherit the sys_net_config privilege. The -i option opens a file rather a network device so that its requirements are not the same.

The snoop command can display the packet's SAMP security attributes and IP options.

spray

The spray(1MTSOL) command sends a one-way stream of packets to a specified host using RPC and reports how many were received along with the transfer rate. If the host is a broadcast address, this program needs to inherit the net_broadcast privilege to run properly.

Trusted Solaris Network Commands

The network commands in this section are only in Trusted Solaris:

- tnchkdb
- tnctl
- tnd
- tninfo
- tokmapd
- tokmapctl

The tnd and tokmapd commands launch the trusted network daemon and token mapping daemons respectively. Token mapping is used when your network is communicating with TSIX host types. The tnctl command loads networking information into the kernel caches; the tninfo command lets you check this information. The tnchkdb examines the network configuration databases for problems. The tokmapctl command lets you troubleshoot problems with TSIX token mapping.

tnchkdb

The tnchkdb(1MTSOL) command checks for errors in the format of the tnrhdb, tnrhtp, and tnidb databases. It should be run every time the database is modified or created.

tnctl

The tnctl(1MTSOL) command lets you configure Trusted Solaris network daemon control parameters for debugging, updating a kernel interface cache, updating a kernel remote host cache, and updating a kernel template cache.

The tnctl command must be started from the trusted path menu and needs to inherit the sys_net_config privilege for updating kernel caches.

tnd

The tnd(1MTSOL) (trusted network daemon) command initializes the kernel with trusted network databases and also reloads the databases on demand. The trusted network daemon is started at the beginning of the boot process. It loads the tnrhdb, tnrhtp, and tnidb databases into the kernel.

The tnd command must be started from the trusted path and inherit the privileges net_privaddr, net_mac_read, and sys_net_config to run. It should be started from an rc script and run at the ADMIN_LOW sensitivity label.

The -d option lets you turn on debugging for tnd and write debugging information to a log file. The file /var/tsol/tndlog is the default log file for debugging the network. It contains one record for each debugging message containing the debug message and time.

By default, tndlog does not exist unless debugging is enabled. Besides the -d option of tnd, the tndlog file can be created using tnctl.

tninfo

The tninfo(1MTSOL) command lets you print out host information (-h), template information (-t), and kernel level network information and statistics (-k). Use tninfo to check that the information that the kernel is caching is correct. This command is intended to be run at ADMIN_HIGH and effective user ID 0. These restrictions can be overridden by the file_mac_read, sys_trans_label, and file_dac_read privileges. The tninfo executable should be maintained with a sensitivity label of ADMIN_LOW with permission bits 555, owner, root, and group sys.

```
# tninfo
==================
kernel statistics
==================
fails host accreditation: 1496
fails interface accreditation: 0
number of seccom structures allocated: 29020
deallocated but memory not yet reclaimed: 28885
memory reclaimed: 28885
```

tokmapd

The tokmapd(1MTSOL) (token mapping daemon) command implements the SATMP token mapping protocol to support the labeling of information transferred over the trusted network. The information is labeled using tokens that represent attribute values. tokmapd is responsible for mapping tokens to attribute values and vice versa. tokmapd accepts token mapping requests from the kernel and from token mapping servers on other hosts. The tokmapd command also provides a number of options for debugging.

The tokmapd command must be started from the trusted path. It must inherit the net_privaddr, proc_setclr and proc_setsl privileges and should be run at sensitivity label ADMIN_HIGH.

tokmapctl

The tokmapctl(1MTSOL) command provides an interface to send control and configuration requests to a tokmapd process. It must be started from the trusted path and must inherit the net_privaddr and net_mac_read privileges. The tokmapctl command should be run at sensitivity label ADMIN_HIGH.

Troubleshooting Networks

The Trusted Solaris tools and commands described in this section can help you debug networking problems. For information on the commands, refer to the appropriate man pages. Refer also to Part 3, “Managing Hosts and Networks,” in the *Trusted Solaris Administrator’s Procedures* manual. In addition, standard network debugging commands such as `snoop(1MTSOL)`, `ipcs(1TSOL)`, and `netstat(1MTSOL)` are also available in the Trusted Solaris environment.

- **To get security information for the source, destination, and gateway hosts in the transmission,** use `tninfo(1MTSOL)`. You can check whether the information that the kernel is caching is correct. This command is intended to be run at ADMIN_HIGH and effective user ID 0. These restrictions can be overridden by the file_mac_read, sys_trans_label, and file_dac_read privileges. The tninfo executable should be maintained with a sensitivity label of ADMIN_LOW with permission bits 555, owner, root, and group sys. Use `tninfo` as follows:
  - **`tninfo -h [<`*`hostname`*`>]`** displays the IP Address, port, and template for all hosts or the given host.
  - **`tninfo -t <`*`templatename`*`>`** displays the following information for all templates or the given template: host type, minimum sensitivity label (in label and hex format), maximum sensitivity label (in label and hex format), allowed privileges, and IP label type (RIPSO, CIPSO, or none).
  - **`tninfo -k`** displays kernel statistics: number of host accreditation check failures, number of network accreditation check failures, and memory allocation statistics.
- **To change or check network security information,** use the Database Manager to access the `tnrhtp`, `tnrhdb`, and `tnidb` files. If you are not using the NIS+ tables for networking, these changes will take place immediately after you save the file(s). If you are using NIS+ tables, then the changes will take place when the network daemon next polls the databases or when the system is rebooted. If you wish the change to take place sooner, you can shorten the polling interval using `tnd(1MTSOL)` with the -p option on the host that needs the updated information, but do not forget to reset the polling period after the change happens.
- **To collect debugging information from the network daemon,** use `tnd(1MTSOL)` with the `-d` option when you start up the network. Debugging data is written by default to the file `/var/tsol/tndlog`. Search the log file for failures and other symptoms of problems.

- **To collect debugging information from the network daemon if the network is already running,** use `tnctl(1MTSOL)` with the `-d` option. Debugging data is written by default to the file `/var/tsol/tndlog`. Search the log file for failures and other symptoms of problems.
- **To check CIPSO transmissions,** use `tninfo -h` and `-t` to make sure that the DOI is the same for the source, destination, and any gateways and that all other security attributes are in order.
- **To check RIPSO transmissions,** use `tninfo -h` and `-t` to make sure that the RIPSO label is the same for the source, destination, and any gateways and that all other security attributes are in order.
- **To check TSIX transmissions,** use `tokmapd` with the `-d` option (or `tokmapctl -d`) to create a log and choose an appropriate debugging level. Debugging data is written by default to the file `/var/tsol/tokmapdlog`. Use `snoop(1MTSOL)` to check that both source and destination can transmit tokens.
- **To check MSIX transmissions,** make sure that `/etc/group` has a special group named “wheel”. Use `tokmapd` with the `-d` option (or `tokmapctl -d`) to create a log and choose an appropriate debugging level. Debugging data is written by default to the file `/var/tsol/tokmapdlog`. Use `snoop(1MTSOL)` to check that both source and destination can transmit tokens.

Administering Auditing 6

This chapter introduces you to auditing in the Trusted Solaris environment. *Auditing* is the process of capturing user activity and other events on the system, storing this information in a set of files called an *audit trail*, and producing system activity reports to fulfill site security policy. Should a breach of security occur, the audit records may enable you to determine how the breach occurred and which user or users were involved. For a more complete description of the auditing process, refer to the *Trusted Solaris Audit Administration* guide.

Planning and Setting Up Auditing

Before you set up auditing for your site, you need to

- Decide which classes of events to audit, including any new classes or events you wish to add to your site.
- Plan where to store the auditing information.
- Define the audit configuration files.

Audit Classes

You need to decide which events you want to audit. You can capture user actions or non-attributable events (that is, events such as interrupts which cannot be attributed to specific users). For the user actions, you can separate successful and failed transactions. Auditing events are organized into classes in Trusted Solaris. The auditing classes for files fall into these general areas:

- Open for reading
- Open for writing
- Attribute changes
- Creations
- Deletions

You can also create your own classes and events as needed and can rearrange the mapping of classes to events. Other classes keep track of such items as process operations, network events, window operations, IPC operations, administrative actions, logins, logouts, application-defined events, ioctl system calls, program executions, Xserver operations, and miscellaneous events. Because auditing information can take up so much room, you need to decide carefully which events are to be audited and only select the classes that contain those events necessary for your site security policy.

Public Objects

A good way to reduce the amount of auditing information collected is to specify certain files and directories as *public objects*. A public object typically contains read-only information, is not modifiable by normal users, and has no implications on security, eliminating the need to track who accesses the object. The system clock is a good example of a public object. When you set the public object flag designating a public object, any other auditing flags specifying the object are ignored.

Audit Information Storage

The large amount of disk space needed for auditing requires that you plan carefully where the information is going to be collected.

If your site uses individual non-networked workstations, it is recommended that each workstation have a dedicated disk for audit records. The dedicated disk should have at least two partitions:

- a primary storage area
- a partition for holding overflow records

For a network of workstations, you should dedicate at least one separate server for collecting audit information and a second server for administering and analyzing the audit data.

In any case, you should set MAC and DAC protections on the audit files and directories to preserve their integrity and prevent snooping.

Audit Configuration Files

The specifications for auditing at a site are stored in these configuration files, which reside in the `/etc/security` subdirectory:

- `audit_control(4TSOL)`– stores audit control information used by the audit daemon, including the preferred order of directories where audit information is stored (the audit daemon uses a directory until the minimum free space warning limit is reached, at which point it stores audit records in the next directory in the list), minimum free space warning limit, system-wide audit flags indicating classes to be audited, and special audit flags for events that cannot be attributed to specific users. The audit flags set in this file are applied to all users. Any exceptions to these flags are set on a per-user basis and specified in the `audit_user` file.
- `audit_user(4TSOL)` – stores auditing criteria for users who are exceptions to the auditing specifications in `audit_control`. This information includes user name, events that are always to be audited, and events that are never to be audited.
- `audit_class(4TSOL)` – stores audit class definitions, including the class mask (that is, the filter that determines which classes are to be tracked), class name, and description.

- audit_event(4TSOL) – stores audit event information, including event number, event name, description, and audit flags identifying the audit class.

If you are setting up auditing for a network, there must be identical versions of the audit_user, audit_class, and audit_event files on each workstation.

Auditing Tools

This section describes the main utility programs and scripts for administering auditing. In summary, auditing is enabled during system installation. You can enable or disable auditing by editing the /etc/init.d/audit scrip and the /etc/system filet. The auditd command starts the audit daemon (if auditing has been enabled). The audit command can halt the daemon, which stops the recording but not the collection of audit records; the audit command provides other options as well for controlling the daemon. The audit_startup script lets you configure auditing parameters during system startup. The audit_warn script lets you specify warnings to send out and other actions to take when there are auditing problems. The praudit command lets you view audit records, auditreduce merges audit trails for convenience in selecting records, and auditstat displays auditing statistics.

audit

The audit(1MTSOL) command is an interface to control the current audit daemon. The audit daemon (auditd) controls the generation and location of audit trail files, using information from the audit_control file. The audit command lets you

- Reset the first directory in the list of audit storage directories in the audit_control file.
- Open a new audit file in the audit directory specified in the audit_control file, as last read by the audit daemon.
- Signal the audit daemon to close the audit trail and halt the recording but not the collection of audit records.

auditconfig

The auditconfig(1MTSOL) command provides a command line interface to get and set kernel audit parameters, including setting various aspects of auditing policy.

audit_startup

The audit_startup(1MTSOL) script initializes the audit subsystem before the audit daemon is started. This script currently consists of a series of auditconfig commands to set the system default policy and download the initial event-to-class mapping. The security administrator can access audit_startup by opening the system_admin folder in the Application Manager. You can configure it as necessary for your site.

audit_warn

The audit_warn(1MTSOL) script processes warning and error messages from the audit daemon. When a problem is encountered, the audit daemon calls audit_warn with the appropriate arguments. The option argument specifies the error type. You can specify a list of mail recipients to be notified when an audit_warn situation arises by defining a mail alias called audit_warn in aliases(4).

praudit

The praudit(1MTSOL) command prints the contents of an audit trail file in readable form.

auditreduce

The auditreduce(1MTSOL) command lets you select or merge records from audit trail files from one or more machines. The merge function merges audit records from one or more input audit trail files into a single output file. The select function lets you select audit records on the basis of criteria relating to the record's content. The merge and select functions can be combined in a script with the praudit command to produce customized reports for your site.

auditstat

The auditstat(1MTSOL) command displays kernel audit statistics, such as the number of audit records processed and how much memory is being used by the kernel audit module.

Other Trusted Solaris Utilities 7

This chapter presents overviews of the Profile Manager, File Manager, and Device Allocation Manager as well as other various utility programs for administering Trusted Solaris.

For a complete listing of commands available in the Trusted Solaris environment, see the man pages: `Intro(1TSOL)`, `Intro(1MTSOL)`, `Intro(2TSOL)`, `Intro(3TSOL)`, `Intro(4TSOL)`, `Intro(5TSOL)`, `Intro(7TSOL)`, `Intro(9TSOL)`, and `Intro(9FTSOL)`.

7

Using the Profile Manager

The Profile Manager is the main tool for working with profiles. The default execution profiles provided with Trusted Solaris are meant to cover most of an organization’s needs for normal users and Trusted Solaris administrative roles. The Profile Manager is provided for situations where you need to change an application’s privileges, add a new application that uses privileges for a limited set of users, or modify or create a profile. The Profile Manager lets you maintain values in the tsolprof database as shown in the following figure.

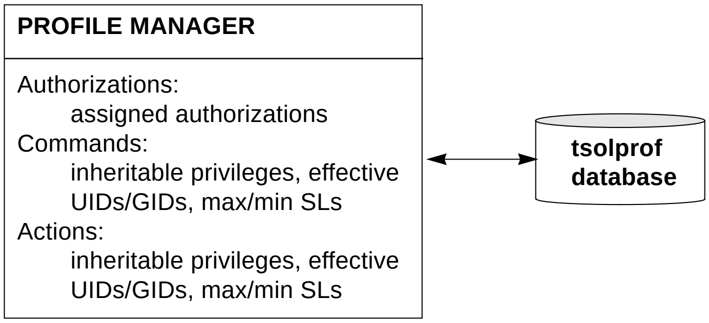
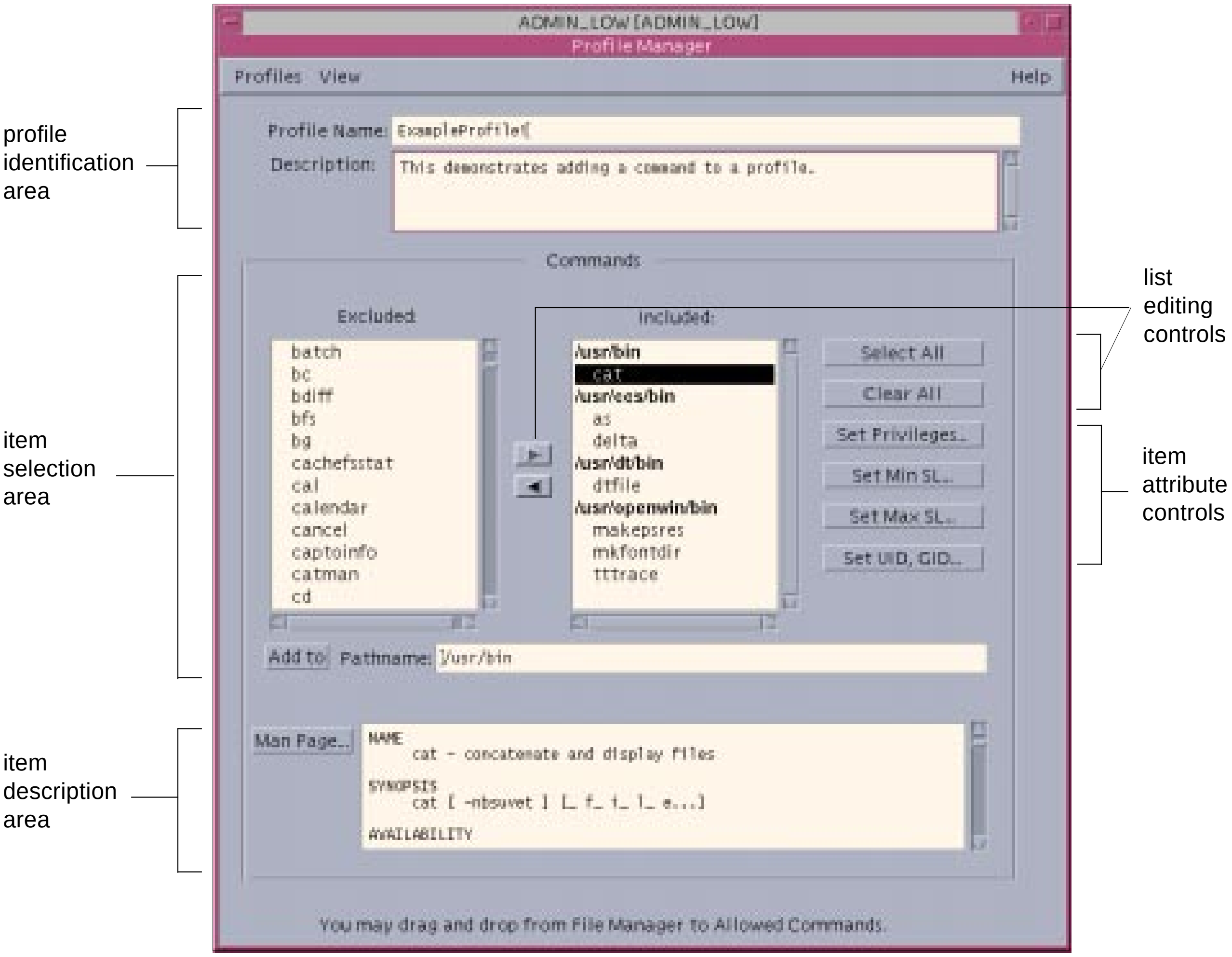

PROFILE MANAGER
Authorizations:
assigned authorizations
Commands:
inheritable privileges, effective UIDs/GIDs, max/min SLs
Actions:
inheritable privileges, effective UIDs/GIDs, max/min SLs
tsolprof database

Figure 7-1 How Execution Profiles are Maintained

You access the Profile Manager from the Application Manager. The main Profile Manager window has three view modes (with different graphical interface configurations) depending on the information you are entering: *action view mode*, *command view mode*, and *authorization view mode*. You select these modes through the View menu. Figure 7-2 shows the Profile Manager in command mode.



Profile Name: ExampleProfile

Description: This demonstrates adding a command to a profile.

Excluded:

```
batch
bc
bdiff
bfs
bg
cachefsstat
cal
calendar
cancel
captoinfo
catman
cd
```

Included:

```
/usr/bin
  cat
/usr/ccs/bin
  as
  delta
/usr/dt/bin
  dtfile
/usr/openwin/bin
  makepsres
  mkfontdir
  tttrace
```

Pathname: /usr/bin

```
NAME
     cat - concatenate and display files

SYNOPSIS
     cat [ -nbsuvet ] [_ f_ i_ l_ e...]

AVAILABILITY
```

You may drag and drop from File Manager to Allowed Commands.

Figure 7-2 Profile Manager

The Profile Manager has these major features which appear in some or all of its viewing modes:

- Profiles menu – lets you create, open, and save profiles.
- View menu – lets you change view modes.
- profile identification area – identifies and describes the profile.

- item selection lists – let you specify included and excluded items.
- item description area – describes the selected item. Authorizations and actions have descriptions. Commands display man pages on request.
- list editing controls – let you move items between lists. The arrows move one item at a time; the buttons move a whole list. Double-clicking an item is a shortcut for moving individual items. The Select All button moves all profiles into the included list. The Clear All button removes all profiles from the included list.

Note – You can also drag and drop commands from the File Manager and actions from the Application Manager onto the included list.

- item attribute controls – let you specify the label range and effective UID and GID for the selected item.

Figure 7-3 summarizes how the Profile Manager is used to build an execution profile. Here is how to interpret the figure:

- When in authorization view mode, the Profile Manager lets you add or edit authorizations in the profile.
- When in action view mode, the Profile Manager lets you add or edit actions in the profile. If you need to assign privileges, a maximum or minimum sensitivity label, or an effective UID or GID, you click the appropriate button to display a dialog box for making the assignment.
- When in command view mode, the Profile Manager lets you add or edit commands in the profile. In similar fashion to action view mode, you can assign privileges, sensitivity labels, and effective UIDs/GIDs.

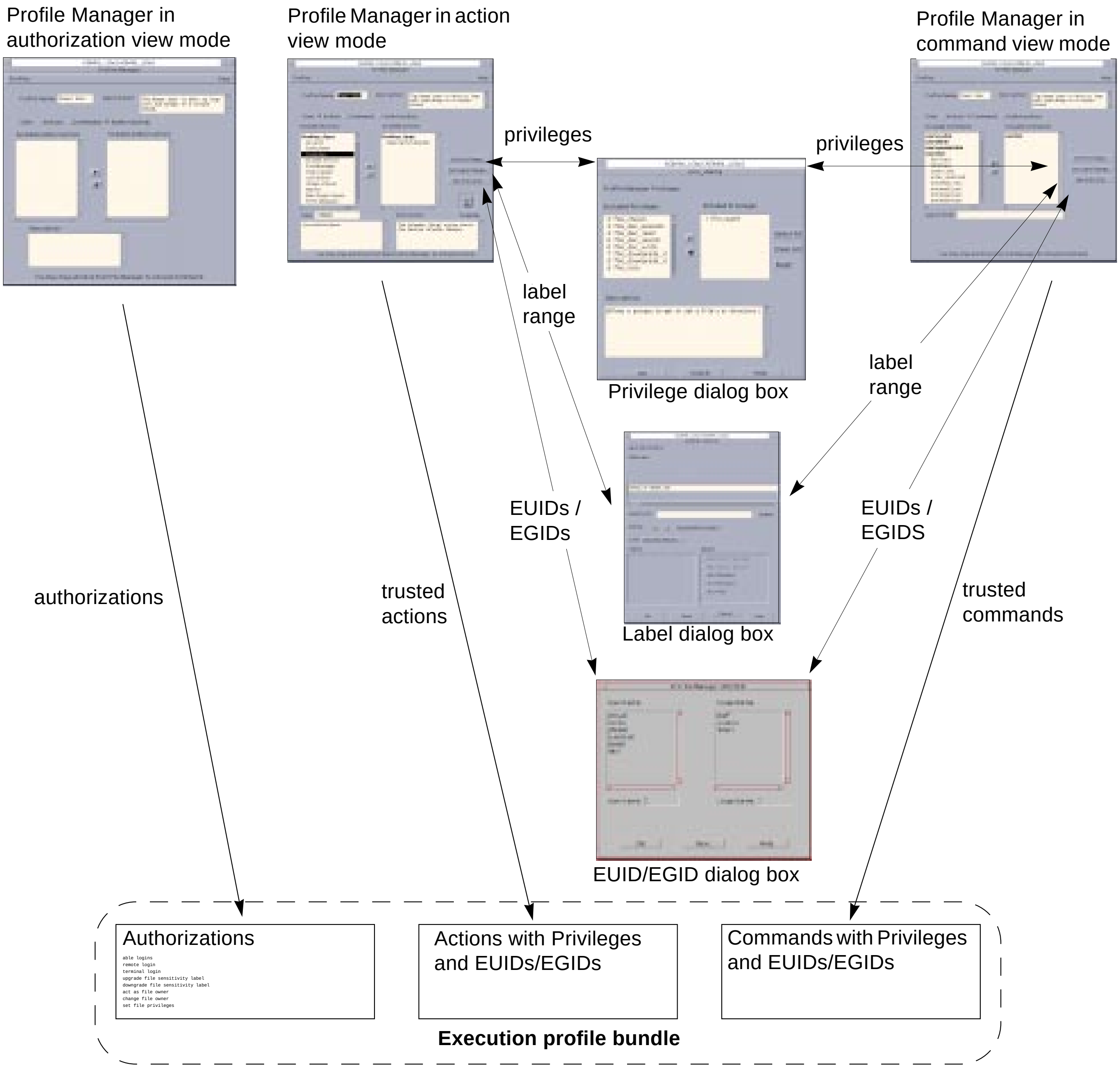


Figure 7-3 How Profiles are Built from the Profile Manager

≡ *7*

Overview of Trusted NFS Mounting

Mounting filesystems in the Trusted Solaris environment is similar to mounting in the regular Solaris system. You can enter the standard mounting information in the `vfstab` file on the client and the sharing information in the `dfstab` file on the server or you can set up mounting dynamically using the `mount(1MTSOL)` command.

The major differences for setting up mounts in the Trusted Solaris environment are:

- The `vfstab(4)` file is supplemented by a special file called `vfstab_adjunct`, whose purpose is to hold security attributes to be applied to the file system.
- The server needs to have a template in its `tnrhdb` file that it can apply to the client. If you are setting up a mount between two Trusted Solaris hosts (sun_tsol), use the host template for Trusted Solaris hosts. If you are setting up a mount between a Trusted Solaris host and an unlabeled host, all data is transmitted by default at the single sensitivity label specified for the unlabeled host in the `tnrhdb` file; however, you can specify different security attributes at mount time using the `vfstab_adjunct` file or the `mount` command with the `-S` option. Mounts are only supported between Trusted Solaris, TSIX, and unlabeled hosts.
- The physical connection between the server and the client must be capable of passing the accreditation checks discussed in "Routing in Trusted Solaris" on page 99.
- The `mount(1M)` command requires that UID is 0. Thus you can only run `mount` from a role or user account with an execution profile that includes `mount`, specifies an effective UID of 0, and runs at ADMIN_LOW. The `mount` command may need these privileges: sys_mount, file_dac_read, file_dac_write, file_dac_search, file_mac_read, file_mac_write, file_mac_search, net_privaddr, proc_setsl, proc_setil (if information labels configured), proc_setclr and sys_trans_label. See `priv_desc(4TSOL)`for more information on these privileges. See "Mounting File Systems in Trusted Solaris" on page 125 for descriptions of commands related to mounting. See also Chapter 11, "Managing Files and File Systems," in *Trusted Solaris Administrator's Procedures.*

Specifying Security Attributes for Mounting

The `vfstab_adjunct` file and `mount` command with `-S` option let you specify the security attributes when you are mounting mounts; these attributes can supply attributes where none exist.

The available security attributes are:

- Access ACL – the access control list to be applied by default to directories and files in the mounted filesystem
- Default ACL - the ACL to be applied to new directories and files created in the mounted filesystem
- UID – the owner of the mounted filesystem
- GID – the group to which the owner of the mounted filesystem belongs
- information label – the information label of all files in the mounted filesystem
- sensitivity label – the sensitivity label of the mounted filesystem
- forced privileges – the set of forced privileges to be applied to executable files in the mounted filesystem
- allowed privileges – the set of allowed privileges to be applied to executable files in the mounted filesystem
- label range – the range of sensitivity labels that can be applied to directories and files in the mounted filesystem
- audit preselection attributes

In any mounts involving a Trusted Solaris host and an unlabeled system, the sensitivity label in the unlabeled host's template is applied unless overridden by the `vfstab_adjunct` file or the `mount` command.

Note – A Trusted Solaris filesystem with security attributes set by either the `setfsattr(1MTSOL)` command or `newsecfs(1MTSOL)` command cannot be overridden at mount time.

Using the File Manager to Change Privileges and Labels

In Trusted Solaris, while most users can set permissions with the File Manager, only administrators and authorized users can change privileges and labels. This section covers the File Manager features for setting privileges and label security attributes on file systems. For a description of the File Manager, refer to Chapter 5, “Managing Files and Directories,” in the *Trusted Solaris User’s Guide*.

In the Trusted Solaris environment, the properties of a file can be maintained from either the File Manager or from command line commands as shown in the following figure.

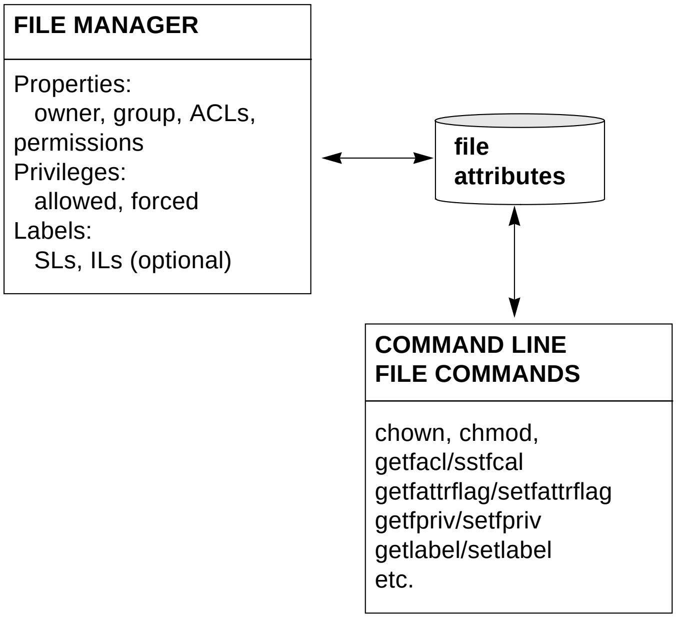


Figure 7-4 How File Attributes are Maintained

The File Manager’s pop-up menu (see Figure 7-5) provides these items, which are not available in base Solaris:

- Change Privileges
- Change Labels

For information on changing a file's security attributes from the command line, see "Changing a File's Security Attributes from the Command Line" on page 131.

For information on changing a file system's security attributes, see "File System Utilities" on page 134.

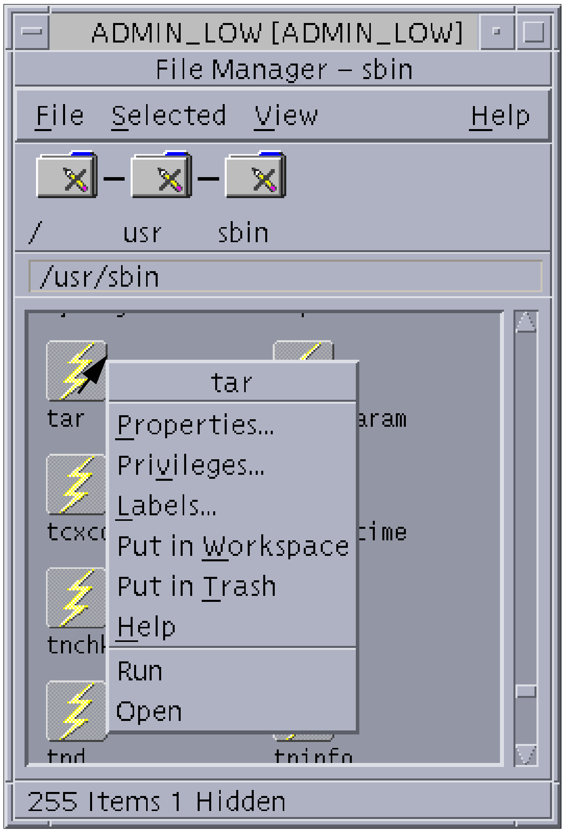
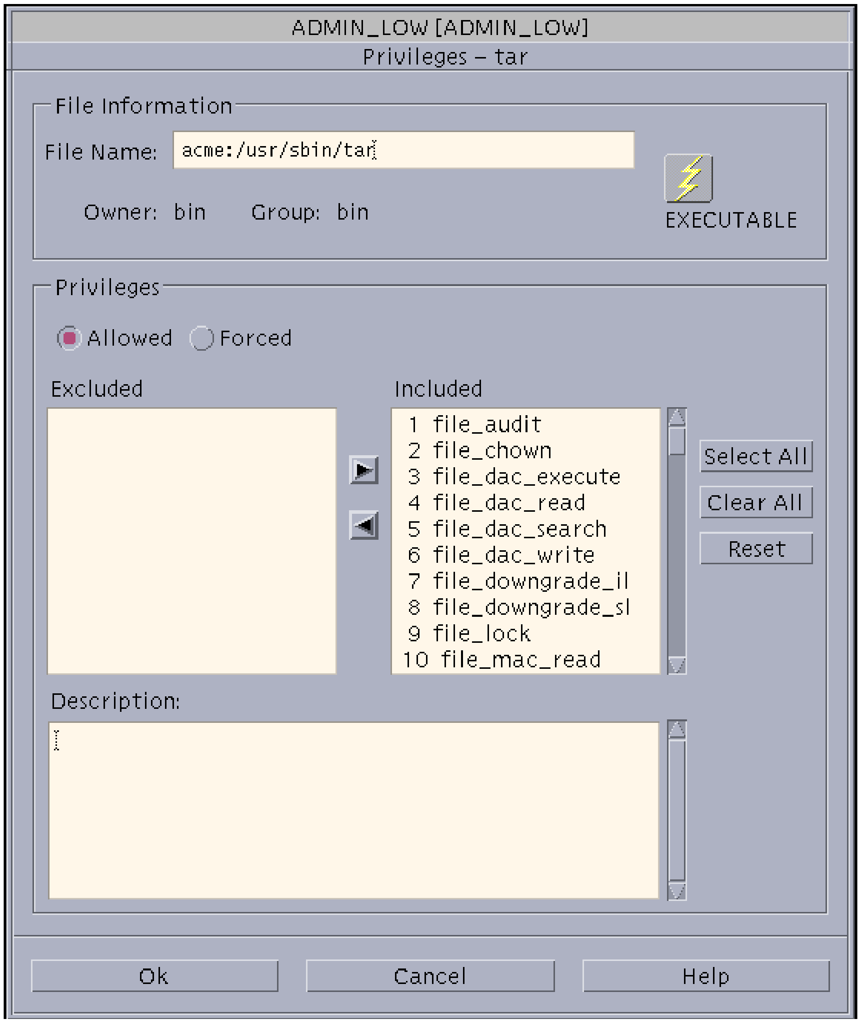


Figure 7-5 File Manager Popup Menu

Changing a File’s Privileges

The Change Privileges option in the File Manager popup menu displays the File Manager Privileges dialog box (see Figure 7-6), which lets you assign allowed and forced privileges to the file selected in the File Manager. See “Allowed and Forced Privilege Assignment” on page 30 for more information on using the File Manager to assign privileges to files.



ADMIN_LOW [ADMIN_LOW]

Privileges – tar

File Information

File Name: acme:/usr/sbin/tar

Owner: bin Group: bin

EXECUTABLE

Privileges

● Allowed ○ Forced

Excluded

Included

1 file_audit
2 file_chown
3 file_dac_execute
4 file_dac_read
5 file_dac_search
6 file_dac_write
7 file_downgrade_il
8 file_downgrade_sl
9 file_lock
10 file_mac_read

Select All

Clear All

Reset

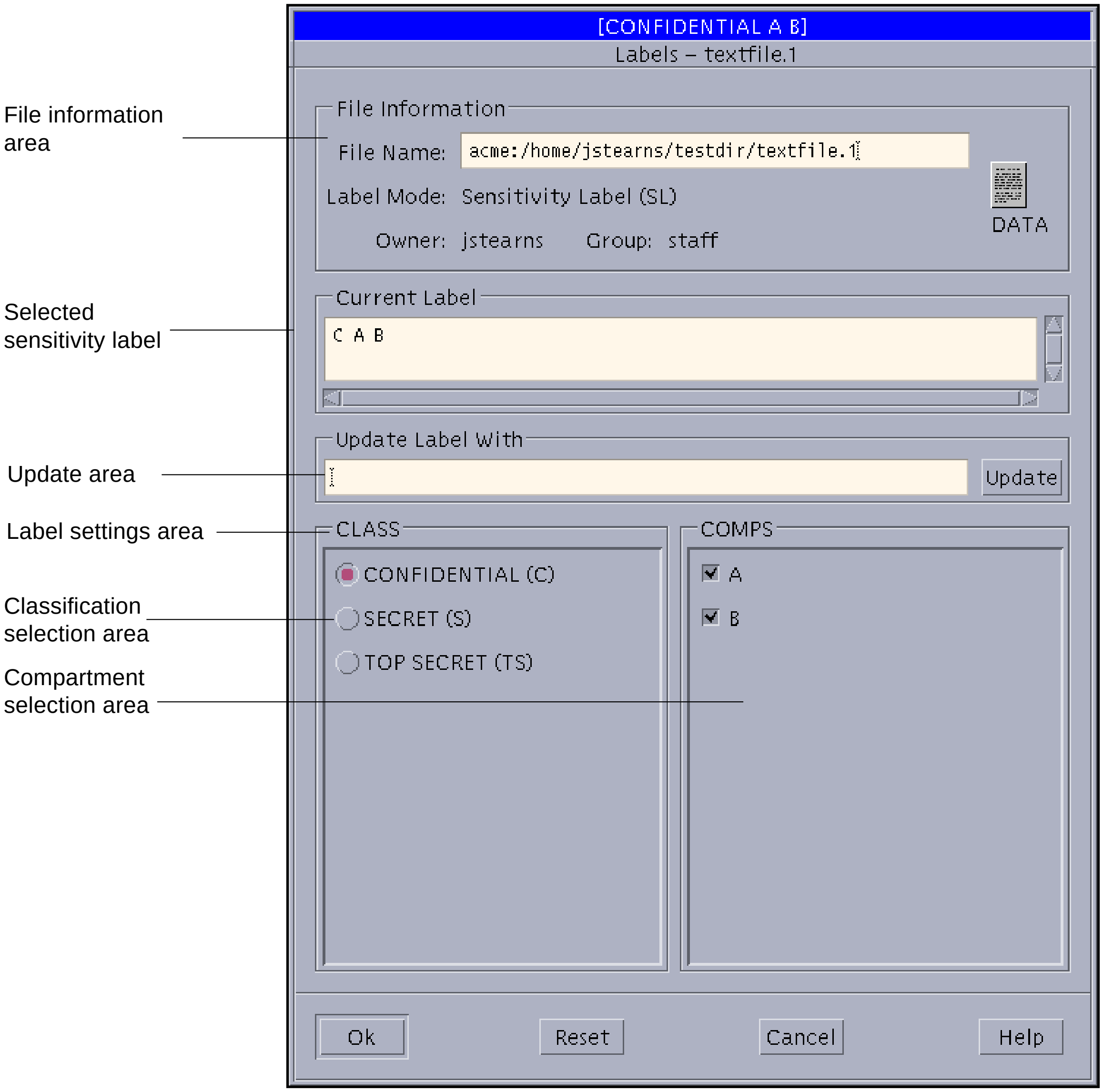
Description:

Ok Cancel Help

Figure 7-6 File Manager Privileges Dialog Box

Changing a File's Labels

The Change Labels option in the File Manager popup menu displays the File Manager Labeler dialog box (see Figure 7-7). It operates in similar fashion to other label dialog boxes in the Trusted Solaris environment. It lets you set the sensitivity label and information label for the file selected in the File Manager.


File information area
Selected sensitivity label
Update area
Label settings area
Classification selection area
Compartment selection area

[CONFIDENTIAL A B]

Labels – textfile.1

File Information

File Name: acme:/home/jstearns/testdir/textfile.1

Label Mode: Sensitivity Label (SL)

Owner: jstearns Group: staff

DATA

Current Label

C A B

Update Label With

Update

CLASS

● CONFIDENTIAL (C)

○ SECRET (S)

○ TOP SECRET (TS)

COMPS

☑ A

☑ B

Ok Reset Cancel Help

Figure 7-7 File Manager: Change Labels Dialog Box

Changing a File’s Security Attributes from the Command Line

This section covers these commands for getting and setting file security attributes:

- `getfattrflag` and `setfattrflag`
- `getfpriv` and `setfpriv`
- `getlabel` and `setlabel`
- `testfpriv`

getfattrflag and setfattrflag

The `getfattrflag(1TSOL)` and `setfattrflag(1TSOL)` commands get and set the security attributes flags for the specified filename. A file’s attribute flag information is only readable to the user if the user has discretionary read, write or execute permission to all directories listed in the path name leading to the file. Mandatory read access to the file is required.

The `setfattrflag(1TSOL)` command can set a directory to multilevel and can make a directory or filename a public object. If you are not the owner of the directory or filename, you need the FILE_OWNER privilege to change its public object flag.

The `getfattrflag(1TSOL)` command indicates whether the pathname is a multilevel directory, a public object, and if it is a directory containing files whose sensitivity labels have been upgraded.

This example shows a file called `myFile` that is private at first and then converted to public using the `setfattrflag(1TSOL)` command.

```
% getfattrflag myFile
myFile: not a public object

% setfattrflag -p 1 myFile

% getfattrflag myFile
myFile: is a public object
```

getfpriv and setfpriv

The getfpriv(1TSOL) and setfpriv(1TSOL) commands get and set the privileges (both forced and allowed) on a file. This example gets the privileges currently on a file called myFile and sets the file_mac_read privilege for that file.

```
% getfpriv myFile
myFile FORCED: none ALLOWED: all

% setfpriv -s -f file_mac_read myFile

% getfpriv myFile
myFile FORCED: file_mac_read ALLOWED: all
```

getlabel and setlabel

The getlabel(1TSOL) and setlabel(1TSOL) commands get and set the sensitivity labels and information labels for a file.

This example gets the initial sensitivity label and information label for a file called myFile. It then sets the information label to CONFIDENTIAL (using the -i option and the short form of the CONFIDENTIAL label. It displays the resulting label and then sets the sensitivity label (using the -s option). Finally, the example sets the combined information and sensitivity labels (called the *CMW label*) by enclosing it in quotation marks and displays the results.

```
% getlabel myFile
myFile: ADMIN_LOW [C]

% setlabel -i C myFile

% getlabel myFile
myFile: CONFIDENTIAL [C]

% setlabel -s SECRET myFile

% getlabel myFile
myFile: CONFIDENTIAL [S]

% setlabel “UNCLASSIFIED [UNCLASSIFIED]” myFile

% getlabel myFile
myFile: UNCLASSIFIED [U]
```

testfpriv

The `testfpriv(1TSOL)` command lets you check or test the privilege sets associated with a file. Basically, you specify some privileges (indicating forced or allowed) and a file, and the command indicates whether the those privileges are included in the file’s set of privileges. You need the file_mac_read privilege to use this command.

7

File System Utilities

This section describes the differences between working with file systems in base Solaris and in Trusted Solaris.

File System Security Attributes

In Trusted Solaris, there is a variety of security attributes associated with file systems. In addition to access control lists (ACLs) and file permissions, which are present in base Solaris, Trusted Solaris provides these attributes:

- attribute flags – these flags describe various characteristics of the file system, such as if the directory is a multilevel directory (FAF_MLD), whether the filesystem is public and therefore not requiring auditing (FAF_PUBLIC), and whether the directory's sensitivity label is dominated by file objects it contains (FAF_UPG_SL)
- information label (IL) – the directory's information label
- sensitivity label (SL) – the directory's sensitivity label
- sensitivity level range – the upper and lower bounds of the directory's sensitivity labels (applies only to multilevel directories)
- multilabel directory (MLD) prefix – the annotation that indicates a directory is multilevel
- allowed privilege set – the set of allowed privileges assigned to the directory
- forced privilege set – the set of forced privileges assigned to the directory

File System Attribute Commands

The commands for administering the file system attributes are:

- `getfsattr`
- `setfsattr`
- `newsecfs`

getfsattr

The `getfsattr(1MTSOL)` command displays the security attributes for the specified file system.

setfsattr

The `setfsattr(1MTSOL)` command sets security attributes on an existing or newly created file system. The file system must be unmounted before using `setfsattr`. When using `setfsattr` with a file system, the file system must be in `/etc/vfstab`.

newsecfs

The `newsecfs(1MTSOL)` command works similarly to `setfsattr`. It sets security attributes on new file systems.

Mounting File Systems in Trusted Solaris

Mounting file systems in Trusted Solaris is performed slightly differently from mounting in base Solaris. The commands related to mounting file systems are:

- `mount`
- `mountd`
- `mount_ufs`
- `mount_hsfs`
- `mount_tmpfs`
- `mount_nfs`
- `share_nfs`
- `share`
- `unshare`
- `nfsstat`
- `nfsd`

For a general description of mounting in the Trusted Solaris environment, see “Modified Solaris Network Commands” on page 106.

mount

The Trusted Solaris version of the mount(1MTSOL) command requires the sys_mount privilege. Both mandatory and discretionary read access (or overriding privileges) are required to the mount point and the device being mounted. Depending on the configuration of the vfstab_adjunct file, the process may need some combination of the proc_setsl, proc_setil, and proc_setclr privileges. The mount command supports mounts to multilabel directories (MLDs). It has a special option, -S which lets you specify security attributes to be associated with the filesystem mount (this option requires that you have sufficient clearance for the sensitivity label specified).

mountd

The Trusted Solaris version of the mountd(1MTSOL) command requires the sys_nfs privilege and supports mounts to multilabel directories (MLDs).

mount_ufs

The Trusted Solaris version of the mount_ufs(1MTSOL) command does not support quotas. It provides two options specified with -o:

- nodev – disallows opens on device special files.
- nopriv – ignores forced privileges on executables.

The mount_ufs command requires the sys_mount privilege. Both mandatory and discretionary read access (or overriding privileges) are required to the mount point and the device being mounted. Depending on the configuration of the vfstab_adjunct file, the process may need some combination of the proc_setsl, proc_setil, and proc_setclr privileges.

mount_hsfs

The Trusted Solaris version of the mount_hsfs(1M) command requires the sys_mount privilege. Both mandatory and discretionary read access (or overriding privileges) are required to the mount point and the device being mounted. Depending on the configuration of the vfstab_adjunct file, the process may need some combination of the proc_setsl, proc_setil, and proc_setclr privileges.

mount_tmpfs

The Trusted Solaris version of the `mount_tmpfs(1MTSOL)` command requires the sys_mount privilege. Both mandatory and discretionary read access (or overriding privileges) are required to the mount point and the device being mounted. Depending on the configuration of the `vfstab_adjunct` file, the process may need some combination of the proc_setsl, proc_setil, and proc_setclr privileges.

mount_nfs

The Trusted Solaris version of the `mount_nfs(1MTSOL)` command provides these options with `-S`:

- `dev|nodev` – Access to character and block devices is allowed or disallowed. The default is `dev`.
- `priv|nopriv` – Forced privileges on executables are allowed or disallowed. The default is `priv`.

The options `quota|noquota` have been removed.

Running `mount_nfs` requires the following:

- sys_mount privilege
- proc_upgrade_sl privilege
- effective UID 0
- process CMW label of ADMIN_LOW[ADMIN_LOW]

The `mount_nfs` command requires the sys_mount and the net_privaddr privileges. Both mandatory and discretionary read access (or overriding privileges) are required to the mount point and the device being mounted. Depending on the configuration of the `vfstab_adjunct` file, the process may need some combination of the proc_setsl, proc_setil, and proc_setclr privileges.

share_nfs

The Trusted Solaris version of the `share_nfs(1MTSOL)` command provides these options with `-S`:

- dev|nodev – access to character and block devices is allowed or disallowed. The default is dev.

- priv | nopriv – Forced privileges on execution are allowed or disallowed. The default is priv.

Running `share_nfs` requires the following:

- sys_nfs privilege
- effective uid 0
- process CMW label of ADMIN_LOW[ADMIN_LOW]

share and unshare

The `share(1M)` command makes a resource of a specified file system type available for mounting. The `unshare(1M)` command makes a resource unavailable for mounting. The Trusted Solaris version of both commands require the sys_nfs privilege.

nfsstat

The `nfsstat(1MTSOL)` command lets you display statistics concerning the NFS and RPC (remote procedure call) interfaces to the kernel. The Trusted Solaris version of the `nfsstat` command requires that you have the net_config privilege when using the `-z` option, which reinitializes the statistics.

nfsd

The `nfsd(1MTSOL)` (MFS daemon) command handles client file system requests. The Trusted Solaris version of the `nfsd` command requires the sys_nfs and net_mac_read privileges to run.

Process Commands

This section describes the following commands for working with processes:

- ipcrm
- ipcs
- pattr
- pclear
- plabel
- ppriv
- pprivtest
- runpd

ipcrm

The ipcrm(1TSOL) command lets you remove a message queue, semaphore set, or shared memory ID.

ipcs

The ipcs(1TSOL) command prints information about active interprocess communication facilities. Without options, information is printed in short format for message queues, shared memory, and semaphores that are currently active in the system.

```
% ipcs
IPC status from <running system> as of Thu Dec 26 12:55:26 1996
Message Queue facility not in system.
Shared Memory:
Semaphores:
s      0 0x000187cf --ra-ra-ra-     root     root
s      1 0x000187ce --ra-ra-ra-     root     root
```

pattr

The pattr(1TSOL) command lets you display the viewable Process Attribute Flags of the current process or a process specified by pid. Those flags that cannot be viewed normally can be viewed with privilege. The Process Attribute Flags are a collection of security flags including:

- Trusted Path Flag
- Privilege Debugging Flag
- NET_TCB
- Flag
- Label View Flags (External View or Internal View)
- Label Translation Flags

```
% pattr
Trusted Path (1 bit):         Enabled/Disabled
Privilege Debugging (1 bit):  Enabled/Disabled
Label Translation (15 bits):  Specific flag (Enabled/Disabled)
Label View (2 bits):          Internal/External
NET_TCB (1 bit):         Enabled/Disabled
```

pclear

The `pclear(1TSOL)` command lets you display the clearance at which the selected process is running.

```
# pclear -p 10546
10546:  ADMIN_HIGH
```

plabel

The `plabel(1TSOL)` command gets the CMW label (that is, combined sensitivity label and information label) for the process.

```
# plabel -p 10546
10546:  ADMIN_LOW [ADMIN_LOW]
```

ppriv

The `ppriv(1TSOL)` command gets the effective privileges of a process.

```
# ppriv -p 10546
10546: file_chown, file_net_search, net_broadcast, net_mac_read,
net_reply_equal, sys_net_config, sys_trans_label
```

pprivtest

The `pprivtest(1TSOL)` command tests if the specified privileges are currently in effect.

runpd

The `runpd(1MTSOL)` command helps you debug problems with privileges. It lets you display the privileges required for a running process. The command must be invoked from the trusted path. `runpd` turns on the priv_debug process attribute and executes the program specified by command. Privilege checking logs are generated for the command process, which inherits the priv_debug process attribute from runpd. (The priv_debug process attribute can be turned on only by a trusted path program such as `runpd`.)

The exit code returned by `runpd` is the exit code returned by command. The `runpd` command displays a list of any privileges the command was lacking.

- `-p` – Execute the command with the trusted path process attribute.

To enable privilege debugging with `runpd` on the system, the *tsol:tsol_privs_debug kerne*l variable in `/etc/system` must be set to 1, and the entry for `kern.debug` and `local0.debug` must be uncommented in the `/etc/syslog.conf` file.

Note – The `runpd` command is uncommitted, which means that it may change between minor releases of Trusted Solaris.

Label Utilities

The complete set of clearances, sensitivity labels, and information labels available to users and roles in Trusted Solaris is defined in the `label_encodings` file (see “Understanding Labels” on page 4). When used internally, labels are stored in a hexadecimal format; unless otherwise specified, they appear to users in ASCII format.

Trusted Solaris provides three commands for administering labels:

- `chk_encodings`
- `atohexlabel`
- `hextoalabel`

chk_encodings

The `chk_encodings(1MTSOL)` command checks the syntax and optionally the semantics of the specified `label_encodings` file. Any errors are written to the standard output file.

atohexlabel

The `atohexlabel(1MTSOL)` command converts an ASCII coded label (sensitivity label, information label, or clearance) into its standard formatted hexadecimal equivalent and writes the result to the standard output file. If no ASCII coded label is specified, one is read from standard input.

hextoalabel

The `hextoalabel(1MTSOL)` command converts a hexadecimal label (sensitivity label, information label, or clearance) into its standard formatted ASCII coded equivalent and writes the result to the standard output file. If no hexadecimal label is specified, one is read from standard input.

Devices and Drivers

Devices need to be secured because they provide a means for importing and exporting data from a machine. In the Trusted Solaris environment, devices are controlled through authorizations assigned in execution profiles and through mandatory access control. Tape drives, floppy disk drives, and microphones are examples of allocatable devices.

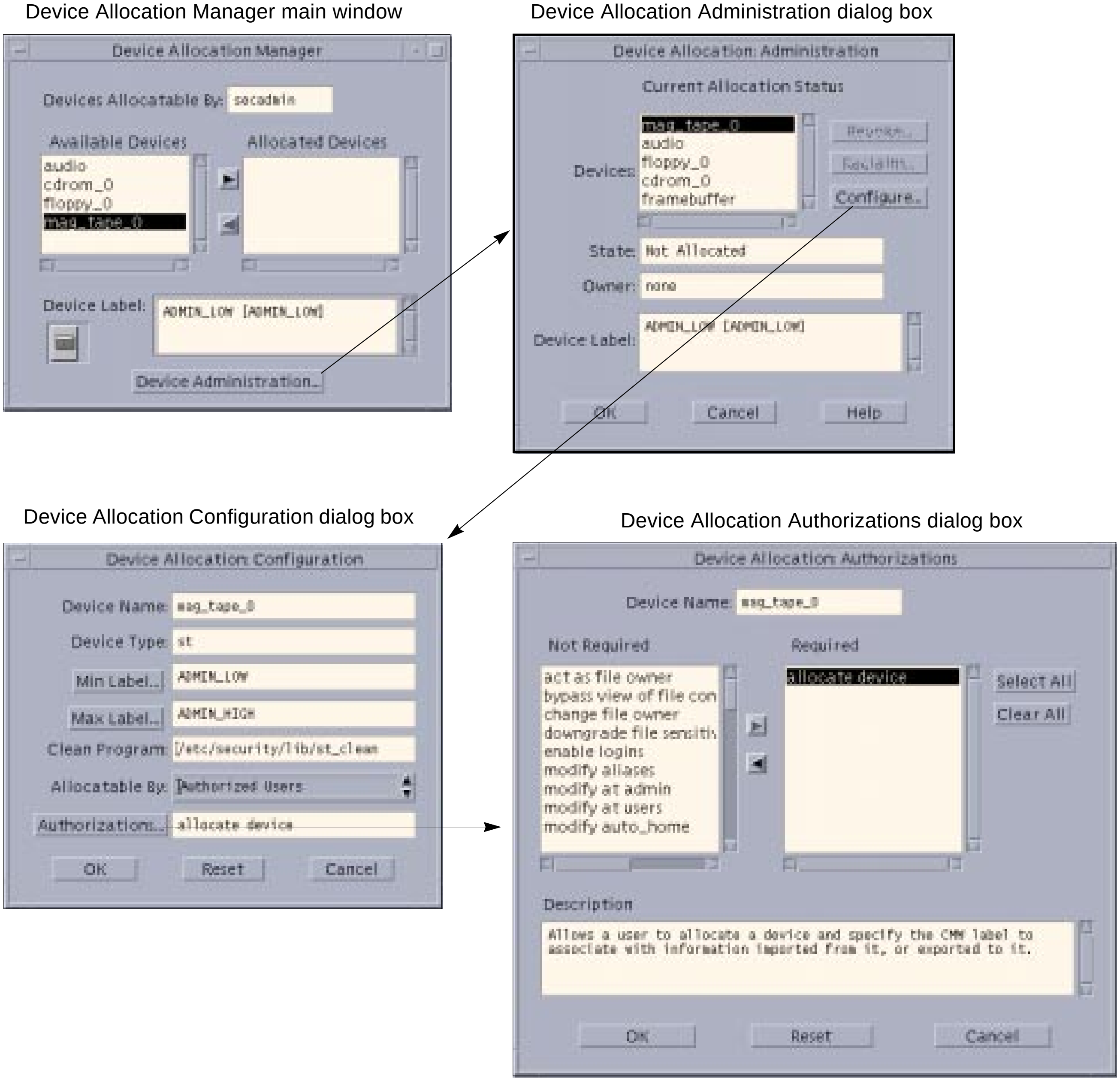
Device allocation is provided by the Solaris SunSHIELD Basic Security Module (BSM); refer to Chapter 4, “Device Allocation,” in the *SunSHIELD Basic Security Module Guide*. Label ranges are unique to Trusted Solaris.

Device allocation provides a way to control the import and export of data. In the Trusted Solaris environment, the administrator decides which devices, if any, can be used to import and export data and includes the devices in two files: `/etc/security/device_maps` and `/etc/security/device_allocate`.

Users allocate devices through the Device Allocation Manager. The Device Allocation Manager mounts the device, runs a clean script to prepare the device (see “Device Clean Scripts” on page 147), and performs the allocation. When finished, the user deallocates the device through the Device Allocation Manager, which runs another clean script and unmounts and deallocates the device.

Administering Devices through the Device Allocation Manager

The Device Allocation Manager is accessed from the Trusted Desktop subpanel above the Style Manager in the Front Panel. The Device Allocation Manager is available to users with the allocate device authorization for allocation and deallocation only. Normal users cannot see if a device is currently allocated to another user and cannot perform maintenance through the Device Administration button in the Device Allocation Manager, which is available to authorized users and administrators only. The Device Allocation Manager administration tools are summarized in Figure 7-8.



Device Allocation Manager main window

Device Allocation Manager

Devices Allocatable By: secadmin

Available Devices: audio, cdrom_0, floppy_0, mag_tape_0

Allocated Devices:

Device Label: ADMIN_LOW [ADMIN_LOW]

Device Allocation Administration dialog box

Device Allocation: Administration

Current Allocation Status

Devices: mag_tape_0, audio, floppy_0, cdrom_0, framebuffer

State: Not Allocated

Owner: none

Device Label: ADMIN_LOW [ADMIN_LOW]

Device Allocation Configuration dialog box

Device Allocation: Configuration

Device Name: mag_tape_0

Device Type: st

Min Label...: ADMIN_LOW

Max Label...: ADMIN_HIGH

Clean Program: /etc/security/lib/st_clean

Allocatable By: Authorized Users

Authorizations...: allocate device

Device Allocation Authorizations dialog box

Device Allocation: Authorizations

Device Name: mag_tape_0

Not Required: act as file owner, bypass view of file con, change file owner, downgrade file sensitiv, enable logins, modify aliases, modify at admin, modify at users, modify auto_home

Required: allocate device

Description

Allows a user to allocate a device and specify the CMW label to associate with information imported from it, or exported to it.

Figure 7-8 Device Allocation Administration Dialog Boxes

Device Administration Dialog Box

Clicking the Device Administration button in the Device Allocation Manager main window causes the Device Administration dialog box to be displayed. The Device Administration dialog box lets you select a device; its state is then displayed. Clicking the Reclaim button lets you make available a device that is currently in an error state. Clicking the Revoke button moves the the selected device from a busy (allocated) state to an available (deallocated) state. The revoke or reclaim device authorization is required to use these buttons.

Device Allocation Configuration Dialog Box

To use the Device Allocation Configuration dialog box requires the configure device attributes authorization. Clicking the Configuration button in the Device Allocation Maintenance dialog box causes the Device Allocation Configuration dialog box to be displayed, in which you can set the minimum and maximum sensitivity labels in the device's label range, designate a new clean program, and specify which users are permitted to use the device.

Device Allocation Authorizations Dialog Box

If you click the Authorizations button in the Device Allocation Configuration dialog box, the Device Allocation Authorizations dialog box is displayed. It lets you specify the authorizations required for using the device.

Device Allocation Security Policy

It is possible to change security policy regarding the allocation of devices. This done by editing the `device_policy(4TSOL)` file. See Chapter 15, "Managing Devices," in *Trusted Solaris Administrator's Procedures*.

Allocation Commands

If you do not have access to the Device Allocation Manager, you can use the commands described in this section to administer allocatable devices. Note that these commands are not intended for users.

allocate

The `allocate(1MTSOL)` command manages the ownership of devices through its allocation mechanism. It ensures that each device is used by only one qualified user at a time.

deallocate

The `deallocate(1MTSOL)` command deallocates a device allocated to the evoking user. The device can be a device defined in `device_deallocate(4TSOL)` or one of the device special files associated with the device. It resets the ownership and the permission on all device special files associated with device, disabling the user's access to that device. This option can be used by the super user to remove access to the device by another user.

When deallocation or forced deallocation is performed, the appropriate device cleaning program is executed, based on the contents of `device_allocate(4TSOL)`. These cleaning programs are normally stored in /etc/security/lib.

list_devices

The `list_devices(1MTSOL)` command lists the allocatable devices in the system according to specified qualifications.

The device and all device special files associated with the device are listed. The device argument is optional and if it is not present, all relevant devices are listed.

dminfo

The `dminfo(1MTSOL)` command displays information about device entries in the device maps file.

add_drv

The `add_drv(1MTSOL)` command is used to inform the system about newly installed devices. Using `add_drv` requires the sys_devices privilege.

rem_drv

The rem_drv(1MTSOL) command is used to inform the system about removed devices. Using rem_drv requires the sys_devices privilege.

Device Clean Scripts

Device clean scripts are special scripts that address two security concerns:

- **Object reuse** – the requirement that a device is clean of previous data before being allocated or reallocated
- **Media labeling** – the requirement that removable information storage media have a physical label indicating its sensitivity label and information label. While the ultimate responsibility for putting the labels on the removable media rests with the user, the device clean scripts can prompt the user to do so.

The name of a device clean script for a specific device is stored with that device's entry in the device_allocate (4TSOL) file. The operations of each device clean program is specific to each device.

The following is a list of tasks that a device clean program may perform:

- **Eject media** – Devices that store information on removable media must be forced to eject that media upon deallocation or reallocation of the device, to prevent passing information to the next user of the device who may be at a different sensitivity label.
- **Reset device state** – Devices that keep state information can potentially be used as a covert channel by the users. Thus driver status information must be reset to default values during deallocation of the device.
- **Remind user about media labeling** – It is a requirement that removable information storage media be labeled with appropriate external media labels. The device user's sensitivity label and information label are passed to the device clean program when it is invoked (See device_clean(1MTSOL) man page for interface detail).

Not all allocatable devices require a device clean program. Devices that do not keep states and do not use removable media do not need a device clean program.

Device clean programs for tape, floppy disk, CD-ROM, and audio devices are provided by Trusted Solaris. The configurable nature of the user device allocation mechanism lets an administrator install new devices and configure device clean programs accordingly.

Allocation Databases

The files for configuring device allocation are:

- `device_allocate`
- `device_deallocate`
- `device_maps`

device_allocate

The `device_allocate(4TSOL)` file contains authorization and mandatory access control information about each allocatable physical device. Each entry contains:

- device name
- device type
- device minimum label
- device maximum label
- device authorization list
- device clean program (a script for enforcing the object reuse policy)
- comment

device_deallocate

The `device_deallocate(4TSOL)` file specifies device deallocation options for allocated devices that have not been deallocated by the user in the events of system boot, user logout, and timeout-forced logout at which point the device deallocation mechanism needs to know whether to force-deallocate the device, to leave it as is, or to prompt the user for a decision.

Each device’s deallocation options are represented by an entry containing:

- device name
- system boot option (for treating allocated devices at boot time)
- user logout (for treating allocated devices when users log out)

- forced logout (for treating allocated devices when users are forced to log out)

device_maps

The `device_maps(4TSOL)` file maps physical device names to device special files. Each device is represented by an entry containing:

- device name
- device type
- device special file list (listing device special files associated with the physical devices)

Device Label Ranges

Each allocatable device has a sensitivity label range. The user's process sensitivity label is used for data imported or exported while the device is allocated to the user.

Tape drives, diskette and CD-ROM drives, and printers are examples of devices that have label ranges.

Device Driver Security

The `ndd(1MTSOL)` command for managing selected configuration parameters in certain kernel drivers needs to inherit the SYS_NET_CONFIG privilege to set driver parameters.

With the `kstat(3K)`command, users can access kernel driver statistics, such as the number of interrupts received by a driver or the number of NFS operations performed. This type of information opens a potential covert channel, because it is noisy, difficult to modulate, and dependent on the rate at which recorded operations are performed. If this is an unacceptable situation, then the sensitivity label for `/dev/kstat` should be changed from ADMIN_LOW to ADMIN_HIGH at installation and those programs that read or write to `/dev/kstat` need to run at ADMIN_HIGH or with the file_mac_read / file_mac_write privilege.

To change the /dev/kstat sensitivity label, edit the /etc/security/tsol/minor_perm.adjunct file and uncomment the line that sets kstat to ADMIN_HIGH. The line begins like this: “#kstat:kstat 0x7777777 ...”.

The commands that access /dev/kstat and need to run at ADMIN_HIGH or with privilege are: netstat, in.rwhod, cachefslog, cachefsstat, nfsstat, fuser, iostat, mpstat, prtdiag, psrinfo, rpc.rstat, sad, sendmail, vmstat, w, and lux.

Miscellaneous Utilities

adminvi

The adminvi(1MTSOL) command is a modified version of vi that provides a restricted text editing environment. It provides all the capabilities of vi except that it does not allow the user to execute shell commands or to write any files other than the files specified on the command line.

rdate

The rdate(1MTSOL) command for setting the system date from a remote host needs to inherit the sys_config privilege to run properly.

sendmail

The Trusted Solaris version of the sendmail(1MTSOL) command for sending messages has been modified to accommodate security considerations.

The Trusted Solaris version adds these privacy options:

- tsoladminlowupgrade – upgrades mail to user minimum label.
- tsoladminlowaccept – delivers mail at ADMIN_LOW.
- tsoladminlowreturn – returns ADMIN_LOW mail to sender.
- tsolotherlowupgrade – upgrades mail to user minimum label.
- tsolotherlowaccept – delivers mail below user minimum label.
- tsolotherlowreturn – returns mail below user minimum label to sender (default).

The tsol* options set the desired action when a message is received at a sensitivity label of ADMIN_LOW or at some other sensitivity label below the recipient's minimum sensitivity label. In each case, there are three options that can be specified:

- upgrade – deliver the message at the recipient's minimum sensitivity label.
- accept – deliver the message at the message's sensitivity label.
- return – return the message to the sender.

Options -ba, -bd, -bi, -bs, -bt, -bv, -M, and -q require that you invoke sendmail from the trusted path and that certain privileges be inherited. The -d and -X options are ignored if sendmail is not invoked from the trusted path. The -bp option will only list queued messages that are dominated by the process. The -p processing option in the configuration file specifies actions to take for mail received at a sensitivity label that is below the recipient's minimum label. The modified options are:

- -ba – goes into ARPANET mode. All input lines must end with a RETURN-LINEFEED, and all messages will be generated with a RETURN-LINEFEED at the end. Also, the From: and Sender: fields are examined for the name of the sender. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the same privileges as for the -bd option.
- -bd – runs as a daemon, waiting for incoming SMTP connections. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the NET_MAC_READ, NET_PRIVADDR, PROC_NOFLOAT and PROC_SETIL privileges.
- -bi – initializes the aliases(4) database. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the same privileges as for the -bd option.
- -bp – prints a summary of the mail queue. Only messages with sensitvity labels dominated by the process are displayed.
- -bs – uses the SMTP protocol as described in RFC 821. This flag implies all the operations of the -ba flag that are compatible with SMTP. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the same privileges as for the -bd option.

- `-bt` – runs in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the same privileges as for the `-bd` option.
- `-bv` – verifies names only - do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the same privileges as for the `-bd` option.
- `-d` *X* – sets debugging value to *X*. This option is not available unless it is invoked from an administrative role.
- `-f` *name* – Sets the name of the "from" person, that is, the sender of the mail. Can only be used by trusted users.
- `-M` *id* – attempts to deliver the queued message with message -id *id*. This option is supported for backward compatibility and the -qI option is preferred. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the same privileges as for the -q option.
- `-q` [*time*] – processes saved messages in the queue at given intervals. If time is omitted, process the queue once. time is given as a tagged number, with s being seconds, m being minutes, h being hours, d being days, and w being weeks. For example, -q1h30m or -q90m would both set the timeout to one hour thirty minutes. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the file_mac_read, file_mac_search, proc_nofloat, and proc_setil privileges.
- `-q` *Xstring* – runs the queue once, limiting the jobs to those matching Xstring. The key letter *X* can be:
  - `I` – to limit based on queue identifier (see -M option).
  - `R` – to limit based on recipient (see - R option).
  - `S` – to limit based on sender.

  A particular queued job is accepted if one of the corresponding addresses contains the indicated string. To use this option, sendmail must be invoked from the trusted path at sensitivity label ADMIN_LOW. It must inherit the file_mac_read, file_mac_search, proc_nofloat, and proc_setil privileges.
- `-R` *string* – Go through the queue of pending mail and attempt to deliver any message with a recipient containing the specified string. This is useful for clearing out mail directed to a machine which has been down for

awhile. This option is supported for backward compatibility and the `-q`*R* option is preferred. This option is available only if `sendmail` is invoked from the trusted path at the ADMIN_LOW sensitivity label. It must inherit the file_mac_read, file_mac_search, proc_nofloat, and proc_setil privileges.

- `-X` *logfile* – Log all traffic in and out of sendmail in the indicated logfile for debugging mailer problems. This produces a lot of data very quickly and should be used sparingly. This option is ignored if not invoked from the trusted path.

sysh

The system shell `sysh(1MTSOL)` is a modified version of the Bourne shell, `sh(1)`. It is used to control the use of privileges in commands run from the rc scripts. `sysh` allows any command to be executed, but consults profiles for the privileges, UID, GID and sensitivity label with which the command is to be run.

The system shell can only be run from a process with the Trusted Path attribute.

Refer to the `sh(1)` man page for a complete usage description. From the `sysh` shell, you can run the `setprof` and `clist` commands, as follows:

- `setprof` *profilename* – `sysh` switches to the specified profile to determine security attributes and privileges to use for executing subsequent commands. This is useful in cases where the same command needs to be run with different privileges at different times. The default profile is the boot profile. It is used when `sysh` starts up, and is the default profile to switch to if `setprof` is called with no arguments.
- `clist` [`-h`] [`-p`] [`-n`] [`-i`] [`-l`] [`-u`] – Displays a list of the commands that are permitted for the user.
  - `-h` – includes a hexadecimal list of the privileges assigned to each command in the command list.
  - `-p` – includes an ASCII list of the privileges assigned to each command in the command list.
  - `-n` – includes a list of the privileges assigned to each command in the command list. The privileges are displayed in decimal number format, separated by commas.
  - `-i` – includes the UID and GID assigned to each command in the command list.

- -l – includes the Sensitivity Label assigned to each command in the command list.
- -u – lists only those commands where the profile assigned privileges that sysh does not have.

sysh normally has all privileges forced so it can run commands with privileges. If for some reason, sysh finds that a command needs privileges that are not permitted, a warning message is printed and the command is run with no privileges.

Note – This interface is uncommitted, which means it may change between minor releases of Trusted Solaris.

tar

In the Trusted Solaris environment, tar(1TSOL)provides a function modifier T for creating, processing, and extracting a tarfile containing the extended security attributes, and MLD and SLD information. When an MLD is encountered in creating or updating a tarfile, the MLD is traversed according to the tar process's sensitivity label and privileges.

In addition, tar provides another function modifier for processing and extracting a tarfile created on a Trusted Solaris 1.x system. The function modifier d can be used only with the function letters t and x.

MAC restrictions apply when tar is used. Appropriate privileges may be required to override access checks that are enforced for the create, update and extract operations.

For creating or updating a tarfile, one or more of the following privileges may be required: file_mac_read, file_mac_write, file_mac_search, file_dac_read, file_dac_write, file_dac_search, or sys_trans_label.

The extended security attributes that require privileges to restore, are restored when the appropriate privileges are present. Hence, to successfully extract files from a tarfile and restore the extended security attributes, one or more of the following privileges may be required: file_mac_read, file_mac_write, file_dac_read, file_dac_write, file_setdac, file_setid, file_chown, file_owner, file_downgrade_sl, file_downgrade_il, file_upgrade_sl, file_upgrade_il, file_setpriv, file_audit, sys_devices, or sys_trans_label.

Index