



Solaris Trusted Extensions Reference Manual



Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819-3219-11
June 2007

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, ToolTalk, Sun Ray, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

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.

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux États-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certains composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux États-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux États-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux États-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des États-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

Contents

Preface	7
Introduction	9
Intro(3TSOL)	10
User Commands	13
dtappsession(1)	14
getlabel(1)	16
getzonepath(1)	17
plabel(1)	18
setlabel(1)	19
System Administration Commands	21
add_allocatable(1M)	22
atohexlabel(1M)	24
chk_encodings(1M)	25
hextoalabel(1M)	26
remove_allocatable(1M)	27
smtnrhdb(1M)	28
smtnrhtp(1M)	32
smtnzonedcfg(1M)	36
tnchkdb(1M)	40
tnctl(1M)	42
tnd(1M)	44
tninfo(1M)	46
updatehome(1M)	48

System Calls	51
fgetlabel(2)	
getlabel(2)	52
Trusted Extensions Library	55
bcleartoh(3TSOL)	
bcleartoh_r(3TSOL)	
bcleartos(3TSOL)	
blcompare(3TSOL)	56
bldominates(3TSOL)	
blequal(3TSOL)	
blinrange(3TSOL)	
blmaximum(3TSOL)	
blminimum(3TSOL)	
blminmax(3TSOL)	57
blstrictdom(3TSOL)	
bltcolor(3TSOL)	58
bltcolor_r(3TSOL)	
bltos(3TSOL)	60
bsltoh(3TSOL)	
bsltoh_r(3TSOL)	
bsltos(3TSOL)	
btohex(3TSOL)	63
getdevicerange(3TSOL)	65
getpathbylabel(3TSOL)	66
getplabel(3TSOL)	68
getuserrange(3TSOL)	69
getzoneidbylabel(3TSOL)	
getzonelabelbyid(3TSOL)	70
getzonelabelbyname(3TSOL)	
getzonerootbyid(3TSOL)	71
getzonerootbylabel(3TSOL)	
getzonerootbyname(3TSOL)	
h_alloc(3TSOL)	
hextob(3TSOL)	72

h_free(3TSOL)	
htobclear(3TSOL)	
htobsl(3TSOL)	
labelbuilder(3TSOL)	73
labelclipping(3TSOL)	78
label_to_str(3TSOL)	80
m_label(3TSOL)	82
m_label_alloc(3TSOL)	
m_label_dup(3TSOL)	
m_label_free(3TSOL)	
sbcleartos(3TSOL)	
sbltos(3TSOL)	83
sbsltos(3TSOL)	
setflabel(3TSOL)	85
stobclear(3TSOL)	
stobl(3TSOL)	87
stobsl(3TSOL)	
str_to_label(3TSOL)	90
tsol_getrhtype(3TSOL)	92
tsol_lbuild_create(3TSOL)	
tsol_lbuild_destroy(3TSOL)	
tsol_lbuild_get(3TSOL)	
tsol_lbuild_set(3TSOL)	
Xbcleartos(3TSOL)	
Xbsltos(3TSOL)	
X Library Extensions	93
XTSOLgetClientAttributes(3XTSOL)	94
XTSOLgetPropAttributes(3XTSOL)	95
XTSOLgetPropLabel(3XTSOL)	96
XTSOLgetPropUID(3XTSOL)	97
XTSOLgetResAttributes(3XTSOL)	98
XTSOLgetResLabel(3XTSOL)	99
XTSOLgetResUID(3XTSOL)	100
XTSOLgetSSHeight(3XTSOL)	101

XTSOLgetWorkstationOwner(3XTSOL)	102
XTSOLIsWindowTrusted(3XTSOL)	103
XTSOLMakeTPWindow(3XTSOL)	104
XTSOLsetPolyInstInfo(3XTSOL)	105
XTSOLsetPropLabel(3XTSOL)	106
XTSOLsetPropUID(3XTSOL)	107
XTSOLsetResLabel(3XTSOL)	108
XTSOLsetResUID(3XTSOL)	109
XTSOLsetSessionHI(3XTSOL)	110
XTSOLsetSessionLO(3XTSOL)	111
XTSOLsetSSHheight(3XTSOL)	112
XTSOLsetWorkstationOwner(3XTSOL)	113
File Formats	115
label_encodings(4)	116
sel_config(4)	121
tnrhdb(4)	123
tnrhtp(4)	126
tnzonecfg(4)	129
TrustedExtensionsPolicy(4)	132
Standards, Environments, and Macros	135
labels(5)	136
pam_tsol_account(5)	138
Index	141

Preface

Both novice users and those familiar with the Solaris Operating System can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview of Solaris Trusted Extensions Man Pages

The following describes each Trusted Extensions man page section:

- Section 1 describes commands that are unique to Solaris Trusted Extensions software. This section includes a man page, `dtappsession`, which extends CDE functionality.
- Section 1M describes Trusted Extensions commands that are used chiefly for system maintenance and administration.
- Section 2 describes the Trusted Extensions system calls.
- Section 3TSOL describes functions that are exclusive to Trusted Extensions software.
- Section 3XTSOL describes functions that extend X Windows software for Solaris Trusted Extensions. These functions are exclusive to Trusted Extensions software.
- Section 4 outlines the formats of Trusted Extensions files. Where applicable, the C structure declarations for the file formats are given.
- Section 5 contains a PAM module and a labels description.

Solaris Trusted Extensions man pages follow the generic format for Solaris OS man pages. For more information and details about each section, see `man(1)` and the Introductory man pages to each section.

REFERENCE

Introduction

Name Intro – introduction to Trusted Extensions interfaces

Description This page introduces all Trusted Extensions man pages, not just those man pages that have the suffix 3TSOL. Trusted Extensions man pages include commands that are available to users and system administrators, the files that are unique to Trusted Extensions, and the interfaces that are available to developers. Trusted Extensions man pages follow the format for Solaris OS man pages.

Interfaces that are exclusive to Trusted Extensions software are in the printed copy of this reference manual. Trusted Extensions modifications to existing Solaris interfaces are not in the printed copy of this reference manual.

Where Trusted Extensions extends Solaris interfaces, the descriptions are on the appropriate Solaris man page. For example, the audit classes that are exclusive to Trusted Extensions are described on the `audit_class(4)` man page. For more information and details about man page sections, see `man(1)` and the introductory man page for each section in the Solaris OS man pages.

Overview of Solaris
Trusted Extensions
Interfaces

The following describes each Trusted Extensions man page section:

- Section 1 describes commands that are unique to Trusted Extensions software. This section includes a man page, [dtappsession\(1\)](#), which extends CDE functionality.
- Section 1M describes Trusted Extensions commands that are used chiefly for system maintenance and administration.
- Section 2 describes the Trusted Extensions system calls.
- Section 3TSOL describes functions that are exclusive to Trusted Extensions software.
- Section 3XTSOL describes functions that extend X Windows software for Trusted Extensions. These functions are exclusive to Trusted Extensions software.
- Section 4 outlines the formats of Trusted Extensions files. Where applicable, the C structure declarations for the file formats are given.
- Section 5 contains a PAM module and a labels description.

Trusted Extensions
Library Interfaces and
Headers

Trusted Extensions adds three libraries:

(3TSOL) These functions constitute the Trusted Extensions library, `libtsol`, and various functions in other libraries that are used only by Trusted Extensions software.

`libtsol.so` is implemented as a shared object but is not automatically linked by the C compilation system. To link with the `libtsol` library, specify `-ltsol` on the `cc` command line.

Functions from a private library, `libtsnet`, are included in this section. To link with the `libtsnet` library, specify `-ltsnet` on the `cc` command line. These libraries are described in the Solaris man pages `libtsol(3LIB)` and `libtsnet(3LIB)`.

(3XTSOL) These functions constitute the Trusted Extensions to the X windows library `libXtsol`. `libXtsol.so` is implemented as a shared object but is not automatically linked by the C compilation system. To link with the `libXtsol` library, specify `-lX11` and then `-lXtsol` on the `cc` command line (`cc -lX11 -lXtsol`).

See Also `Intro(1)`, `man(1)`

Solaris Trusted Extensions Developer's Guide

Solaris Trusted Extensions Administrator's Procedures

REFERENCE

User Commands

Name dtappsession – start a new Application Manager session

Synopsis /usr/dt/bin/dtappsession [*hostname*]

Description dtappsession is a specialized version of the Xsession shell script. It is an alternative to using the CDE remote login that allows you to access a remote host without logging out of your current CDE session. dtappsession starts a new instance of the CDE Application Manager in its own ToolTalk™ session. It can be used to remotely display the Application Manager back to your local display after logging in to a remote host with the rlogin(1) command.

A new, independent instance of ttsession(1) starts a simple session management window. This window displays the title

remote_hostname: Remote Administration

where *remote_hostname* is the system that is being accessed. The window also displays an Exit button. Clicking Exit terminates the ToolTalk session and all windows that are part of the session.

The Application Manager that is displayed can be used to start remote CDE actions to run in this session. Exiting the Application Manager does not terminate the session, and it is not recommended. Clicking Exit is the recommended way to end the session. To avoid confusing the remote CDE applications with local ones, it is recommended that a new CDE workspace be created for clients in the remote session.

The *hostname* is not needed when the DISPLAY environment variable is set to the local hostname on the remote host.

On a system that is configured with Trusted Extensions, dtappsession can be used for remote administration by administrative roles that have the ability to log in to the remote host.

dtappsession does not require any privilege, and it does not need to run on a system that is configured with Trusted Extensions. When installed in /usr/dt/bin on a Solaris system, along with the startApp.ds file, dtappsession can be used to administer the remote Solaris system from a local system that is configured with Trusted Extensions. However, in this case, the CDE workspace that is used for remote display must be a normal workspace, rather than a role workspace.

Examples EXAMPLE 1 Remote Login and dtappsession

After creating a new CDE workspace, type the following in a terminal window:

```
# rlogin remote_hostname
password: /*type the remote password*/

# dtappsession local_hostname /* on the remote host */
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdttsu

Files `/usr/dt/bin/startApp.ds` dt Korn shell script for session manager window

Bugs X11/CDE applications that do not register with the ToolTalk session manager will not exit automatically when the session is terminated. Such applications must be explicitly terminated.

See Also `dtfile(1)`, `rlogin(1)`, `ttsession(1)`, `attributes(5)`

Chapter 8, “Remote Administration in Trusted Extensions (Tasks),” in *Solaris Trusted Extensions Administrator’s Procedures*

Name getlabel – display the label of files

Synopsis /usr/bin/getlabel [-sS] *filename...*

Description getlabel displays the label that is associated with each *filename*. When options are not specified, the output format of the label is displayed in default format.

Options -s Display the label that is associated with *filename* in short form.
-S Display the label that is associated with *filename* in long form.

Return Values getlabel exits with one of the following values:

- 0 Successful completion.
- 1 Unsuccessful completion due to usage error.
- 2 Unable to translate label.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability (Command Line)	Stable
Stability (Output)	Not an interface

See Also [setlabel\(1\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#)

Name getzonepath – display root path of the zone corresponding to the specified label

Synopsis /usr/bin/getzonepath {sensitivity-label}

Description getzonepath displays the root pathname of the running labeled zone that corresponds to the specified sensitivity label. The returned pathname is relative to the caller's root pathname, and has the specified sensitivity label.

If the caller is in the global zone, the returned pathname is not traversable unless the caller's processes have the `file_dac_search` privilege.

If the caller is in a labeled zone, the caller's label must dominate the specified label. Access to files under the returned pathname is restricted to read-only operations.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability	Stable

Diagnostics getzonepath exits with one of the following values:

- 0 Success
- 1 Usage error
- 2 Failure; error message is the system error number from `getzonerootbylabel(3TSOL)`

See Also `getzonerootbylabel(3TSOL)`, `attributes(5)`

“Acquiring a Sensitivity Label” in *Solaris Trusted Extensions Developer's Guide*

Name plabel – get the label of a process

Synopsis /usr/bin/plabel [-sS] [*pid*...]

Description plabel, a proc tools command, gets the label of a process. If the *pid* is not specified, the label displayed is that of the plabel command. When options are not specified, the output format of the label is displayed in default format.

Options -s Display the label that is associated with *pid* in short form.

-S Display the label that is associated with *pid* in long form.

Return Values plabel exits with one of the following values:

- 0 Successful completion.
- 1 Unsuccessful completion because of a usage error.
- 2 Inability to translate label.
- 3 Inability to allocate memory.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability	Stable
Stability (Output)	Not an interface

See Also proc(1), [getplabel\(3TSOL\)](#), attributes(5)

Name setlabel – move files to zone with corresponding sensitivity label

Synopsis /usr/bin/setlabel *newlabel filename...*

Description setlabel moves files into the zone whose label corresponds to *newlabel*. The old file pathname is adjusted so that it is relative to the root pathname of the new zone. If the old pathname for a file's parent directory does not exist as a directory in the new zone, the file is not moved. Once moved, the file might no longer be accessible in the current zone.

Unless *newlabel* and *filename* have been specified, no labels are set.

Labels are defined by the security administrator at your site. The system always displays labels in uppercase. Users can enter labels in any combination of uppercase and lowercase. Incremental changes to labels are supported.

Refer to [setlabel\(3TSOL\)](#) for a complete description of the conditions that are required to satisfy this command, and the privileges that are needed to execute this command.

Return Values setlabel exits with one of the following values:

- 0 Successful completion.
- 1 Usage error.
- 2 Error in getting, setting or translating the label.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability	Stable

Usage On the command line, enclose the label in double quotes unless the label is only one word. Without quotes, a second word or letter separated by a space is interpreted as a second argument.

```
% setlabel SECRET somefile
% setlabel "TOP SECRET" somefile
```

Use any combination of upper and lowercase letters. You can separate items in a label with blanks, tabs, commas or slashes (/). Do not use any other punctuation.

```
% setlabel "ts a b" somefile
% setlabel "ts,a,b" somefile
% setlabel "ts/a b" somefile
% setlabel " TOP SECRET A B " somefile
```

Examples EXAMPLE 1 To Set a Label

To set *somefile*'s label to SECRET A:

```
example% setlabel "Secret a" somefile
```

EXAMPLE 2 To Turn On a Compartment

Plus and minus signs can be used to modify an existing label. A plus sign turns on the specified compartment for *somefile*'s label.

```
example% setlabel +b somefile
```

EXAMPLE 3 To Turn Off a Compartment

A minus sign turns off the compartments that are associated with a classification. To turn off compartment A in *somefile*'s label:

```
example% setlabel -A somefile
```

If an incremental change is being made to an existing label and the first character of the label is a hyphen (-), a preceding double-hyphen (--) is required.

To turn off compartment -A in *somefile*'s label:

```
example% setlabel -- -A somefile
```

Notes This implementation of setting a label is meaningful for the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. For more information, see [label_encodings\(4\)](#).

See Also [setflabel\(3TSOL\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#)

REFERENCE

System Administration Commands

Name add_allocatable – add entries to allocation databases

Synopsis /usr/sbin/add_allocatable [-f] [-s] [-d] -n *name* -t *type* -l *device-list*
[-a *authorization*] [-c *clean*] [-o *key=value*]

Description add_allocatable creates new entries for user allocatable devices that are to be managed by the device allocation mechanism. add_allocatable can also be used to update existing entries of such devices.

add_allocatable can also create and update entries for non-allocatable devices, such as printers, whose label range is managed by the device allocation mechanism.

add_allocatable can be used in shell scripts, such as installation scripts for driver packages, to automate the administrative work of setting up a new device.

Use list_devices(1) to see the names and types of allocatable devices, their attributes, and device paths.

Options

-f	Force an update of an already-existing entry with the specified information. add_allocatable exits with an error if this option is not specified when an entry with the specified device name already exists.
-s	Turn on silent mode. add_allocatable does not print any error or warning messages.
-d	If this option is present, add_allocatable updates the system-supplied default attributes of the device type specified with -t.
-n <i>name</i>	Adds or updates an entry for device that is specified by <i>name</i> .
-t <i>type</i>	Adds or updates device entries that are of a type that are specified by <i>type</i> .
-l <i>device-list</i>	Adds or updates device paths to the device that is specified with -n. Multiple paths in <i>device-list</i> must be separated by white spaces and the list must be quoted.
-a <i>authorization</i>	Adds or updates the authorization that is associated with either the device that is specified with -n or with devices of the type that is specified with -t. When more than one authorization is specified, the list must be separated by commas and must be quoted. When the device is not allocatable, <i>authorization</i> is specified with an asterisk (*) and must be quoted. When the device is allocatable by any user, <i>authorization</i> is specified with the at sign (@) and must be quoted. Default authorization is '@'.
-c <i>clean</i>	Specifies the device_clean(5) program <i>clean</i> to be used with the device that is specified with -n or with devices of the type that is specified with -t. The default clean program is /bin/true.

- o key=value* Accepts a string of colon-separated *key=value* pairs for a device that is specified with *-n* or with devices of the type that is specified with *-t*. The following keys are currently interpreted by the system:
- `minlabel` The minimum label at which the device can be used.
 - `maxlabel` The maximum label at which the device can be used.
 - `class` Specifies a logical grouping of devices. For example, all Sun Ray™ devices of all device types is a logical grouping. The `class` keyword has no default value.
 - `xdpv` Specifies the display name of the X session. This keyword is used to identify devices that are associated with the X session. The `xdpv` keyword has no default value.

Errors When successful, `add_allocatable` returns an exit status of 0 (true). `add_allocatable` returns a nonzero exit status in the event of an error. The exit codes are as follows:

- 1 Invocation syntax error
- 2 Unknown system error
- 3 An entry already exists for the specified device. This error occurs only when the *-f* option is not specified.
- 4 Permission denied. User does not have DAC or MAC access record updates.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The invocation is Uncommitted. The options are Uncommitted. The output is Not-an-interface.

See Also `allocate(1)`, `deallocate(1)`, `list_devices(1)`, `remove_allocatable(1M)`, `attributes(5)`, `device_clean(5)`

Name atohexlabel – convert a human readable label to its internal text equivalent

Synopsis /usr/sbin/atohexlabel [*human-readable-sensitivity-label*]
/usr/sbin/atohexlabel -c [*human-readable-clearance*]

Interface Level This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Description atohexlabel converts a human readable label into an internal text representation that is safe for storing in a public object. If no option is supplied, the label is assumed to be a sensitivity label.

Internal conversions can later be parsed to their same value. This internal form is often hexadecimal. The converted label is written to the standard output file. If no human readable label is specified, the label is read from the standard input file. The expected use of this command is emergency repair of labels that are stored in internal databases.

Options -c Identifies the human readable label as a clearance.

Exit Status The following exit values are returned:

- 0 On success.
- 1 On failure, and writes diagnostics to the standard error file.

Files /etc/security/tsol/label_encodings
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	See NOTES below

See Also [hextoalabel\(1M\)](#), [label_to_str\(3TSOL\)](#), [str_to_label\(3TSOL\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#)

“How to Obtain the Hexadecimal Equivalent for a Label” in *Solaris Trusted Extensions Administrator’s Procedures*

Notes The stability of the command output is Stable for systems with the same label_encodings file. The stability of the command invocation is Stable for systems that implement the DIA MAC policy.

- Name** chk_encodings – check the label encodings file syntax
- Synopsis** /usr/sbin/chk_encodings [-a] [-c *maxclass*] [*pathname*]
- Interface Level** This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.
- Description** chk_encodings checks the syntax of the label-encodings file that is specified by *pathname*. With the -a option, chk_encodings also prints a semantic analysis of the label-encodings file that is specified by *pathname*. If *pathname* is not specified, chk_encodings checks and analyzes the /etc/security/tsol/label_encodings file.
- If label-encodings file analysis was requested, whatever analysis can be provided is written to the standard output file even if errors were found.
- Options**
- a Provide a semantic analysis of the label encodings file.
 - c *maxclass* Accept a maximum classification value of *maxclass* (default 255) in the label encodings file CLASSIFICATIONS section.
- Errors** When successful, chk_encodings returns an exit status of 0 (true) and writes to the standard output file a confirmation that no errors were found in *pathname*. Otherwise, chk_encodings returns an exit status of nonzero (false) and writes an error diagnostic to the standard output file.
- Attributes** See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	Mixed. See NOTES below

- Files** /etc/security/tsol/label_encodings
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

See Also [label_encodings\(4\)](#), [attributes\(5\)](#), [labels\(5\)](#)

“How to Analyze and Verify the label_encodings File” in *Solaris Trusted Extensions Label Administration*

- Notes** The stability of the syntactic checking is considered standard and controlled by DIA document DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993. The stability of the command output is undefined. The stability of the command invocation is stable for systems that implement the DIA MAC policy.

Name hextoalabel – convert an internal text label to its human readable equivalent

Synopsis /usr/sbin/hextoalabel [*internal-text-sensitivity-label*]
/usr/sbin/hextoalabel -c [*internal-text-clearance*]

Interface Level This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Description hextoalabel converts an internal text label into its human readable equivalent and writes the result to the standard output file. This internal form is often hexadecimal. If no option is supplied, the label is assumed to be a sensitivity label.

If no internal text label is specified, the label is read from the standard input file. The expected use of this command is emergency repair of labels that are stored in internal databases.

Options -c Identifies the internal text label as a clearance.

Exit Status The following exit values are returned:

- 0 On success.
- 1 On failure, and writes diagnostics to the standard error file.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	See NOTES below

Files /etc/security/tsol/label_encodings
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

See Also [atohexlabel\(1M\)](#), [label_to_str\(3TSOL\)](#), [str_to_label\(3TSOL\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#)

“How to Obtain a Readable Label From Its Hexadecimal Form” in *Solaris Trusted Extensions Administrator’s Procedures*

Notes The stability of the command output is Stable for systems with the same label_encodings file. The stability of the command invocation is Stable for systems that implement the DIA MAC policy.

Name remove_allocatable – remove entries from allocation databases

Synopsis /usr/sbin/remove_allocatable [-f] -n *name*
 /usr/sbin/remove_allocatable [-f] [-d] -t *dev-type*

Description remove_allocatable removes entries of user allocatable devices from the device allocation mechanism. remove_allocatable also removes entries of some non-allocatable devices, such as printers, whose label range is managed by the mechanism.

Options

- d Removes system-supplied default attributes of the device type that is specified with -t.
- f Force the removal of an entry. remove_allocatable exits with an error if this option is not specified when an entry with the specified device name no longer exists.
- n *name* Removes the entry for the device *name*.
- t *dev-type* Removes devices of type *dev-type*.

Errors When successful, remove_allocatable returns an exit status of 0 (true). remove_allocatable returns a nonzero exit status in the event of an error. The exit codes are as follows:

- 1 Invocation syntax error
- 2 Unknown system error
- 3 Device *name* or *dev-type* not found. This error occurs only when the -f option is not specified.
- 4 Permission denied. User does not have DAC or MAC access to database.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The invocation is Uncommitted. The options are Uncommitted. The output is Not-an-interface.

See Also allocate(1), deallocate(1), [add_allocatable\(1M\)](#), attributes(5), device_clean(5)

Name smtnrhdb – manage entries in the tnrhdb database

Synopsis /usr/sadm/bin/smtnrhdb *subcommand* [*auth_args*] -- *subcommand_args*

Description The smtnrhdb command adds, modifies, deletes, and lists entries in the tnrhdb database.

smtnrhdb *subcommands* are:

- add** Adds a new entry to the tnrhdb database. To add an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.
- delete** Deletes an entry from the tnrhdb database. To delete an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.
- list** Lists all entries in the tnrhdb database. To list an entry, the administrator must have the `solaris.network.host.read` and `solaris.network.security.read` authorizations.
- modify** Modifies an entry in the tnrhdb database. To modify an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

Options The smtnrhdb authentication arguments, *auth_args*, are derived from the smc arg set. These arguments are the same regardless of which subcommand you use. The smtnrhdb command requires the Solaris Management Console to be initialized for the command to succeed (see smc(1M)). After rebooting the Solaris Management Console server, the first smc connection can time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must be *preceded* by the -- option.

auth_args The valid *auth_args* are -D, -H, -l, -p, -r, and -u; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user might be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either -D or --domain.

-D | --domain *domain*

Specifies the default domain that you want to manage. The syntax of *domain=type:host_name/domain_name*, where *type* is `dns`, `ldap`, or `file`; *host_name* is the name of the server; and *domain_name* is the name of the domain you want to manage.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage,

meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

- H | --hostname *host_name:port* Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, 898. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port 898.
- l | --rolepassword *role_password* Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
- p | --password *password* Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
- r | --rolename *role_name* Specifies a role name for authentication. If you do not specify this option, no role is assumed.
- u | --username *user_name* Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
- This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the -- option.

subcommand_args Note: Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- h Displays the command's usage statement.
- H *hostname* Specifies the name of the host. For the list subcommand, the *hostname* argument is not specified. This is not required if the *ipaddress* subcommand argument is specified.
- i *ipaddress* Specifies the IP address of the host. This is not required if the *hostname* subcommand argument is specified.
- n *templatename* Specifies the name of the template.

`-p prefixlen` Specifies the prefix length (in bits) of a wildcard representation of the IP address. The prefix is the left-most portion of the IP address.

`-w ipaddress-wildcard` Specifies the IP address of the subnet using a wildcard.

- One of the following sets of arguments must be specified for subcommand `add`:

```
-H hostname -n templatename |
-i ipaddress -n templatename |
-w ipaddress-wildcard -n templatename [ -p prefixlen ] |
-h
```

- One of the following sets of arguments must be specified for subcommand `modify`:

```
-H hostname -n templatename |
-i ipaddress -n templatename |
-w ipaddress-wildcard -n templatename [ -p prefixlen ] |
-h
```

- One of the following sets of arguments must be specified for subcommand `delete`:

```
-H hostname |
-i ipaddress |
-w ipaddress-wildcard [ -p prefixlen ] |
-h
```

- The subcommand `list` takes the following argument:

```
-h
```

Examples **EXAMPLE 1** Specifying the Template Name for a Wildcard IP Address

The admin role specifies the template name, `cipso_lan`, for a series of hosts that use the IP address wildcard `192.168.113.0` on the local file system. Since no authorization arguments were specified, the administrator connects to port 898 of the local host on the local server with the file domain type, which are the defaults. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtnrhdb add -- -w 192.168.113.0 -n cipso_lan
```

EXAMPLE 2 Deleting an Entry in the `tnrhdb` Database

The admin role connects to port 898 (which happens to be the default) of the LDAP server and deletes a host entry from the database by specifying its IP address, `192.168.113.8`. Since the domain was not specified, the file domain type and local server are used by default. The administrator is prompted for the admin password.

```
/usr/sadm/bin/smtnrhdb delete \
-D ldap:/example.domain -i 192.168.113.8
```

Exit Status The following exit values are returned:

-
- 0 Successful completion.
 - 1 Invalid command syntax. A usage message displays.
 - 2 An error occurred while executing the command. An error message displays.

Files The following files are used by the smtnrhdb command:

/etc/security/tsol/tnrhdb Trusted network remote-host database. See [tnrhdb\(4\)](#).

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmgts

See Also [smc\(1M\)](#), [tnrhdb\(4\)](#), [attributes\(5\)](#)

Name smtnrhttp – manage entries in the trusted network template database

Synopsis /usr/sadm/bin/smtnrhttp *subcommand* [*auth_args*] -- [*subcommand_args*]

Description The smtnrhttp command adds, modifies, deletes, and lists entries in the tnrhttp database.

smtnrhttp *subcommands* are:

- add** Adds a new entry to the tnrhttp database. To add an entry, the administrator must have the `solaris.network.security.read` and `solaris.network.security.write` authorizations.
- modify** Modifies an entry in the tnrhttp database. To modify an entry, the administrator must have the `solaris.network.security.read` and `solaris.network.security.write` authorizations.
- delete** Deletes an entry from tnrhttp database. To delete an entry, the administrator must have the `solaris.network.security.read` and `solaris.network.security.write` authorizations.
- list** Lists entries in the tnrhttp database. To list an entry, the administrator must have the `solaris.network.security.read` authorizations.

Options The smtnrhttp authentication arguments, *auth_args*, are derived from the smc arg set and are the same regardless of which subcommand you use. The smtnrhttp command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first smc connection can time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must be *preceded* by the -- option.

auth_args The valid *auth_args* are -D, -H, -l, -p, -r, and -u; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user might be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either -D or --domain.

-D | --domain *domain*

Specifies the default domain that you want to manage. The syntax of *domain=type:/host_name/domain_name*, where *type* is `dns`, `ldap`, or `file`; *host_name* is the name of the server; and *domain_name* is the name of the domain you want to manage.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server.

	Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
-H --hostname <i>host_name:port</i>	Specifies the <i>host_name</i> and <i>port</i> to which you want to connect. If you do not specify a <i>port</i> , the system connects to the default port, 898. If you do not specify <i>host_name:port</i> , the Solaris Management Console connects to the local host on port 898.
-l --rolepassword <i>role_password</i>	Specifies the password for the <i>role_name</i> . If you specify a <i>role_name</i> but do not specify a <i>role_password</i> , the system prompts you to supply a <i>role_password</i> . Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
-p --password <i>password</i>	Specifies the password for the <i>user_name</i> . If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
-r --rolename <i>role_name</i>	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
-u --username <i>user_name</i>	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
--	This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the -- option.
<i>subcommand_args</i>	<i>Note:</i> Descriptions and other arg options that contain white spaces must be enclosed in double quotes.
-h	Displays the command's usage statement.
-n <i>templatename</i>	Specifies the name of the template.
-t <i>hosttype</i>	Specifies the hosttype of the new host. Valid values are unlabeled and cipso.
-x doi= <i>doi-value</i>	Specifies the DOI value.
-x max= <i>maximum-label</i>	Specifies the maximum label. Values can be a hex value or string (such as admin_high).

- x min=*minimum-label* Specifies the minimum label. Values can be a hex value or string (such as admin_low).
- x label=*default-label* Specifies the default label when the host type is unlabeled. This option does not apply if *hosttype* is CIPSO. Values can be a hex value or string (such as admin_low).
- x slset=*l1,l2,l3,l4* Specifies a set of sensitivity labels. You can specify up to four label values, separated by commas. Values can be a hex value or string (such as admin_low).

- One of the following sets of arguments must be specified for subcommand `add`:

```
-n template name (
  -t cipso [ -x doi=doi-value -x min=minimum-label -x max=maximum-label -x
    slset=l1,l2,l3,l4 ] |
  -t unlabeled [ -x doi=doi-value -x min=minimum-label -x max=maximum-label -x
    label=default-label -x slset=l1,l2,l3,l4 ] |
  -h
)
```

- One of the following sets of arguments must be specified for subcommand `modify`:

```
-n template name (
  -t cipso [ -x doi=doi-value -x min=minimum-label -x max=maximum-label -x
    slset=l1,l2,l3,l4 ] |
  -t unlabeled [ -x doi=doi-value -x min=minimum-label -x max=maximum-label -x
    label=default-label -x slset=l1,l2,l3,l4 ] |
  -h
)
```

Note: If the host type is changed, all options for the new host type must be specified.

- One of the following sets of arguments must be specified for subcommand `delete`:

```
-n templatename |
-h
```

- The following argument can be specified for subcommand `list`:

```
-n templatename |
-h
```

Examples EXAMPLE 1 Adding a New Entry to the Network Template Database

The admin role connects to port 898 of the LDAP server and creates the unlabeled_ntk entry in the tnrrhtp database. The new template is assigned a host type of unlabeled, a domain of

EXAMPLE 1 Adding a New Entry to the Network Template Database (Continued)

interpretation of 1, minimum label of public, maximum label of restricted, and a default label of needtoknow. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtnrhtp \
add -D ldap:directoryname -H servername:898 -- \
-n unlabeled_ntk -t unlabeled -x DOI=1 \
-x min=public -x max=restricted -x label="need to know"
```

Exit Status The following exit values are returned:

- 0 Successful completion.
- 1 Invalid command syntax. A usage message displays.
- 2 An error occurred while executing the command. An error message displays.

Files The following files are used by the smtnrhtp command:

/etc/security/tsol/tnrhtp Trusted network remote-host templates. See [tnrhtp\(4\)](#).

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmgts

See Also [smc\(1M\)](#), [tnrhtp\(4\)](#), [attributes\(5\)](#)

Name smtzonecfg – manage entries in the zone configuration database for Trusted Extensions networking

Synopsis /usr/sadm/bin/smtzonecfg *subcommand* [*auth_args*] -- [*subcommand_args*]

Description The smtzonecfg command adds, modifies, deletes, and lists entries in the tzonecfg database.

smtzonecfg *subcommands* are:

add Adds a new entry to the tzonecfg database. To add an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

modify Modifies an entry in the tzonecfg database. To modify an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

delete Deletes an entry from the tzonecfg database. To delete an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

list Lists entries in the tzonecfg database. To list an entry, the administrator must have the `solaris.network.host.read` and `solaris.network.security.read` authorizations.

Options The smtzonecfg authentication arguments, *auth_args*, are derived from the `smc arg` set and are the same regardless of which subcommand you use. The smtzonecfg command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first `smc` connection can time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must be *preceded* by the `--` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user can be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `--domain`.

`-D | --domain domain` Specifies the default domain that you want to manage. The syntax of `domain=type:/host_name/domain_name`, where *type* is `dns`, `ldap`, or `file`; *host_name* is the name of the server; and *domain_name* is the name of the domain you want to manage.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis. This option specifies the domain for all other tools.

- H | --hostname *host_name:port* Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, 898. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port 898.
- l | --rolepassword *role_password* Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
- p | --password *password* Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
- r | --rolename *role_name* Specifies a role name for authentication. If you do not specify this option, no role is assumed.
- u | --username *user_name* Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
- This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the -- option.

subcommand_args Note: Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- h Displays the command's usage statement.
- n *zonename* Specifies the zone name for the entry. This name is used when the zone is configured. *zonename* is case-sensitive. The specified zone name must be one of the configured zones on the system. The following command returns a list of configured zones:

```
/usr/sbin/zoneadm list -c
```

- l *label* Specifies the label for the zone. This field is used to label the zone when the zone is booted.
- x *polycymatch=0|1* Specifies the policy match level for non-transport traffic. Only values of 0 (match the label) or 1 (be within the label range of the zone) are accepted. See [tnzonecfg\(4\)](#) for more detail. This subcommand argument is optional. If not specified, it will have a default value of 0.
- x *mlpzone=""|port/protocol* Specifies the multilevel port configuration entry for zone-specific IP addresses. Multiple *port/protocol* combinations are separated by a semi-colon. The empty string can be specified to remove all existing MLP zone values. This subcommand argument is optional.
- x *mlpshared=""|port/protocol* Specifies the multilevel port configuration entry for shared IP addresses. Multiple *port/protocol* combinations are separated by a semi-colon. The empty string can be specified to remove all existing MLP shared values. This subcommand argument is optional.

- One of the following sets of arguments must be specified for subcommand `add`:

```
-n zonename -l label [-x polycymatch=policy-match-level \
-x mlpzone=port/protocol;... | -x mlpshared=port/protocol;... ]
```

```
-h
```

- One of the following sets of arguments must be specified for subcommand `modify`:

```
-n zonename [-l label] [-x polycymatch=policy-match-level \
-x mlpzone=port/protocol;... | -x mlpshared=port/protocol;... ]
```

```
-h
```

- One of the following arguments must be specified for subcommand `delete`:

```
-n zonename |
```

```
-h
```

- The following argument can be specified for subcommand `list`:

```
-n zonename |
```

```
-h
```

Examples EXAMPLE 1 Adding a New Entry to the Zone Configuration Database

The admin role creates a new zone entry, `public`, with a label of `public`, a policy match level of 1, and a shared MLP port and protocol of 666 and TCP. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtzonecfg add -- -n public -l public \  
-x policymatch=1 -x mlpshared=666/tcp
```

EXAMPLE 2 Modifying an Entry in the Zone Configuration Database

The admin role changes the `public` entry in the `tnzonecfg` database to `needtoknow`. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtzonecfg modify -- -n public -l needtoknow
```

EXAMPLE 3 Listing the Zone Configuration Database

The admin role lists the entries in the `tnzonecfg` database. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtzonecfg list --
```

Exit Status The following exit values are returned:

- 0 Successful completion.
- 1 Invalid command syntax. A usage message displays.
- 2 An error occurred while executing the command. An error message displays.

Files The following files are used by the `smtzonecfg` command:

`/etc/security/tsol/tnzonecfg` Trusted zone configuration database. See [tnzonecfg\(4\)](#).

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmgts

See Also [smc\(1M\)](#), [tnzonecfg\(4\)](#), [attributes\(5\)](#)

Name tnchkdb – check file syntax of trusted network databases

Synopsis /usr/sbin/tnchkdb [-h *path*] [-t *path*] [-z *path*]

Description tnchkdb checks the syntax of the [tnrhtp\(4\)](#), [tnrhdb\(4\)](#), and [tnzonecfg\(4\)](#) databases. By default, the *path* for each file is:

- /etc/security/tsol/tnrhtp
- /etc/security/tsol/tnrhdb
- /etc/security/tsol/tnzonecfg

You can specify an alternate path for any or all of the files by specifying that path on the command line by using the -h (tnrhdb), -t (tnrhtp) and -z (tnzonecfg) options. The options are useful when testing a set of modified files before installing the files as new system databases.

All three database files are checked for integrity. tnchkdb returns an exit status of 0 if all of the files are syntactically and, to the extent possible, semantically correct. If one or more files have errors, then an exit status of 1 is returned. If there are command line problems, such as an unreadable file, an exit status of 2 is returned. Errors are written to standard error.

To avoid cascading errors, when there are errors in tnrhtp, the template names in tnrhdb are not validated.

tnchkdb can be run at any label, but the standard /etc/security/tsol files are visible only in the global zone.

- Options**
- h [*path*] Check *path* for proper tnrhdb syntax. If *path* is not specified, then check /etc/security/tsol/tnrhdb.
 - t [*path*] Check *path* for proper tnrhtp syntax. If *path* is not specified, then check /etc/security/tsol/tnrhtp.
 - z [*path*] Check *path* for proper tnzonecfg syntax. If *path* is not specified, then check /etc/security/tsol/tnzonecfg.

Examples EXAMPLE 1 Sample Error Message

The tnchkdb command checks for CIPSO errors. In this example, the admin_low template has an incorrect value of ADMIN_HIGH for its default label.

```
# tnchkdb
checking /etc/security/tsol/tnrhtp ...
tnchkdb: def_label classification 7fff is invalid for cipso labels:
line 14 entry admin_low
tnchkdb: def_label compartments 241-256 must be zero for cipso labels:
line 14 entry admin_low
checking /etc/security/tsol/tnrhdb ...
checking /etc/security/tsol/tnzonecfg ...
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability (Command Line)	Evolving
Stability (Output)	Unstable

Files

<code>/etc/security/tsol/tnrhdb</code>	Trusted network remote-host database
<code>/etc/security/tsol/tnrhtp</code>	Trusted network remote-host templates
<code>/etc/security/tsol/tnzonecfg</code>	Trusted zone configuration database

See Also `tnd(1M)`, `tnctl(1M)`, `tnrhdb(4)`, `tnrhtp(4)`, `tnzonecfg(4)`, `attributes(5)`

“How to Check the Syntax of Trusted Network Databases” in *Solaris Trusted Extensions Administrator’s Procedures*

Notes It is possible to have inconsistent but valid configurations of `tnrhtp` and `tnrhdb` when LDAP is used to supply missing templates.

Name tnctl – configure Trusted Extensions network parameters

Synopsis /usr/sbin/tnctl [-dfv] [-h *host* [/i>prefix] [:*template*]] [-m *zone:mlp:shared-mlp*]
[-t *template* [:*key=val* [:*key=val*]]] [-HTz] *file*

Description tnctl provides an interface to manipulate trusted network parameters in the Solaris kernel.

As part of Solaris Trusted Extensions initialization, tnctl is run in the global zone by an smf(5) script during system boot. The tnctl command is not intended to be used during normal system administration. Instead, if a local trusted networking database file is modified without using the Solaris Management Console, the administrator first issues [tnchkdb\(1M\)](#) to check the syntax, and then refreshes the kernel copy with this command:

```
# svcadm restart svc:/network/tnctl
```

See WARNINGS about the risks of changing remote host and template information on a running system.

Options -d

Delete matching entries from the kernel. The default is to add new entries.

When deleting MLPs, the MLP range must match exactly. MLPs are specified in the form:

port[-*port*]/*protocol*

Where *port* can be a number in the range 1 to 65535, or any known service (see [services\(4\)](#)), and *protocol* can be a number in the range 1 to 255, or any known protocol (see [protocols\(4\)](#)).

-f

Flush all kernel entries before loading the entries that are specified on the command line. The flush does not take place unless at least one entry parsed successfully.

-v

Turn on verbose mode.

-h *host*[/*prefix*][:*template*]

Update the kernel remote-host cache on the specified *host* or, if a template name is given, change the kernel's cache to use the specified *template*. If *prefix* is not specified, then an implied prefix length is determined according to the rules used for interpreting the [tnrhdb\(4\)](#). If -d is specified, then a template name cannot be specified.

-m *zone:mlp:shared-mlp*

Modify the kernel's multilevel port (MLP) configuration cache for the specified *zone*. *zone* specifies the zone to be updated. *mlp* and *shared-mlp* specify the MLPs for the zone-specific and shared IP addresses. The *shared-mlp* field is effective in the global zone only.

- t *template*[*key=val*[:*key=val*]]
Update the kernel template cache for *template* or, if a list of *key=val* pairs is given, change the kernel's cache to use the specified entry. If -d is specified, then *key=val* pairs cannot be specified. See [tnrhtp\(4\)](#) for the format of the entries.
- T *file*
Load all template entries in *file* into the kernel cache.
- H *file*
Load all remote host entries in *file* into the kernel cache.
- z *file*
Load just the global zone's MLPs from *file* into the kernel cache. To reload MLPs for a non-global zone, reboot the zone:

zoneadm -z non-global zone reboot

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	Unstable

Files

/etc/security/tsol/tnrhdb	Trusted network remote-host database
/etc/security/tsol/tnrhtp	Trusted network remote-host templates
/etc/security/tsol/tzonecfg	Trusted zone configuration database
/etc/nsswitch.conf	Configuration file for the name service switch

See Also [svcs\(1\)](#), [svcadm\(1M\)](#), [tinfo\(1M\)](#), [tnd\(1M\)](#), [tnchkdb\(1M\)](#), [zoneadm\(1M\)](#), [nsswitch.conf\(4\)](#), [protocols\(4\)](#), [services\(4\)](#), [tnrhdb\(4\)](#), [tnrhtp\(4\)](#), [tzonecfg\(4\)](#), [attributes\(5\)](#), [smf\(5\)](#)

“How to Synchronize the Kernel Cache With Trusted Network Databases” in *Solaris Trusted Extensions Administrator's Procedures*

Notes The `tnctl` service is managed by the service management facility, [smf\(5\)](#), under the service identifier:

```
svc:/network/tnctl
```

The service's status can be queried by using [svcs\(1\)](#). Administrative actions on this service, such as refreshing the kernel cache, can be performed using [svcadm\(1M\)](#), as in:

```
svcadm refresh svc:/network/tnctl
```

Warnings Changing a template while the network is up can change the security view of an undetermined number of hosts.

Name tnd – trusted network daemon

Synopsis /usr/sbin/tnd [-p *poll-interval*]

Description The tnd (trusted network daemon) initializes the kernel with trusted network databases and also reloads the databases on demand from an LDAP server and local files. tnd follows the order specified in the `nsswitch.conf(4)` file when loading configuration databases. tnd is started at the beginning of the boot process.

tnd loads two databases into the kernel: the remote host database, [tnrhdb\(4\)](#) and the remote-host template database, [tnrhtp\(4\)](#). These databases and their effect on the trusted network are described in their respective man pages. When the associated LDAP database or local databases are changed, tnd also updates the local kernel cache at the predetermined interval.

If a local trusted networking database file is modified, the administrator should run [tnchkdb\(1M\)](#) to check the syntax, and should also run `svcadm refresh svc:/network/tnd` to initiate an immediate database scan by tnd.

tnd is intended to be started from an `smf(5)` script and to run in the global zone. The following signals cause specific `svcadm` actions:

SIGHUP	Causes <code>svcadm refresh svc:/network/tnd</code> to be run. Initiates a rescan of the local and LDAP <code>tnrhdb</code> and <code>tnrhtp</code> databases. tnd updates the kernel database with any changes found.
SIGTERM	Causes <code>svcadm disable svc:/network/tnd</code> to be run. Terminates the tnd daemon. No changes are made to the kernel database.

Options -p *poll-interval* Set poll interval to *poll-interval* seconds. The default *poll-interval* is 1800 seconds (30 minutes).

Examples EXAMPLE 1 Changing the Poll Interval

The following command changes the polling interval to one hour, and puts this interval in the SMF repository. At the next boot, the tnd poll interval will be one hour.

```
# svccfg -s network/tnd setprop tnd/poll_interval=3600
```

The following command changes the polling interval, but does not update the repository. At the next boot, the tnd poll interval remains the default, 30 minutes.

```
# tnd -p 3600
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level (Command)	Stable
Stability Level (Service)	Project Private

Files /etc/security/tsol/tnrhdb Trusted network remote-host database
 /etc/security/tsol/tnrhtp Trusted network remote-host templates
 /etc/security/tsol/tnzonecfg Trusted zone configuration database
 /etc/nsswitch.conf Configuration file for the name service switch

See Also [svcs\(1\)](#), [svcadm\(1M\)](#), [tninfo\(1M\)](#), [tnctl\(1M\)](#), [tnchkdb\(1M\)](#), [tnrhdb\(4\)](#), [tnrhtp\(4\)](#), [tnzonecfg\(4\)](#), [nsswitch.conf\(4\)](#), [attributes\(5\)](#), [smf\(5\)](#)

“How to Synchronize the Kernel Cache With Trusted Network Databases” in *Solaris Trusted Extensions Administrator's Procedures*

Notes The tnd service is managed by the service management facility, [smf\(5\)](#), under the service identifier:

```
svc:/network/tnd
```

The service's status can be queried by using [svcs\(1\)](#). Administrative actions on this service, such as requests to restart the daemon, can be performed using [svcadm\(1M\)](#), as in:

```
svcadm restart svc:/network/tnd
```

Name tninfo – print kernel-level network information and statistics

Synopsis /usr/sbin/tninfo [-h *hostname*] [-m *zone-name*] [-t *template*]

Description tninfo provides an interface to retrieve and display kernel-level network information and statistics.

Options

- h *hostname* Display the security structure for the specified host in the remote-host cache. The output should reflect what is specified in the tn rhdb database.
- m *zone-name* Display the MLP configuration associated with the specified zone. The output should reflect what is specified in the tnzonecfg database.
- t *template* Display the structure associated with the specified *template*. The output should reflect what is specified in the tn rhtp database.

Examples EXAMPLE 1 Displaying Remote Host Structures Cached in the Kernel

This example shows the remote host structures cached in the kernel. The output reflects the definition in the tn rhdb database.

```
# tninfo -h machine1
  IP address= 192.168.8.61
  Template = cipso
```

EXAMPLE 2 Displaying Multilevel Ports for the Global Zone

This example shows the kernel-cached MLPs for the global zone. The output reflects the definition in the tnzonecfg database, plus any dynamically allocated MLPs. private indicates zone-specific MLPs.

```
# tninfo -m global
private:23/tcp;111/tcp;111/udp;515/tcp;2049/tcp;6000-6003/tcp;
        32812/tcp;36698/ip;38634/tcp;64365/ip
shared: 6000-6003/tcp
```

EXAMPLE 3 Displaying the cipso Template Definition

This example shows the kernel-cached cipso template definition. The output reflects the definition in the tn rhtp database.

```
# tninfo -t cipso
=====
  Remote Host Template Table Entries:
  -----
  template: cipso
  host_type: CIPSO
  doi: 1
```

EXAMPLE 3 Displaying the cipso Template Definition (Continued)

```

min_sl: ADMIN_LOW
hex: ADMIN_LOW
max_sl: ADMIN_HIGH
hex: ADMIN_HIGH

```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability (Command Line)	Evolving
Stability (Output)	Unstable

Files `/etc/security/tsol/tnrhdb` Trusted network remote-host database
`/etc/security/tsol/tnrhtp` Trusted network remote-host templates
`/etc/security/tsol/tzonecfg` Trusted zone configuration database

See Also `tnd(1M)`, `tnctl(1M)`, `tnrhdb(4)`, `tnrhtp(4)`, `tzonecfg(4)`, `attributes(5)`

“How to Synchronize the Kernel Cache With Trusted Network Databases” in *Solaris Trusted Extensions Administrator’s Procedures*

Name updatehome – update the home directory copy and link files for the current label

Synopsis /usr/bin/updatehome [-cirs]

Description updatehome reads the user's minimum-label copy and link-control files (`.copy_files` and `.link_files`). These files contain a list of files to be copied and symbolically linked from the user's minimum-label home directory to the user's home directory at the current label.

The Solaris Trusted Extensions `dt session` program performs an `updatehome` whenever a newly labeled workspace is created so that the user's favorite files are available for use. For example, the user probably wants a symlink to such files as `.profile`, `.login`, `.cshrc`, `.exrc`, `.mailrc`, and `~/bin`. The `updatehome` command provides a convenient mechanism for accomplishing this symlink. The user can add files to those to be copied (`.copy_files`) and to those to be symbolically linked (`.link_files`).

- Options**
- c Replace existing home-directory copies at the current label. The default is to skip over existing copies.
 - i Ignore errors encountered. The default aborts on error.
 - r Replace existing home-directory copies or symbolic links at the current label. This option implies options -c and -s. The default is to skip over existing copies or symbolic links.
 - s Replace existing home-directory symbolic links at the current label. The default is to skip over existing symbolic links.

Return Values Upon success, `updatehome` returns 0. Upon failure, `updatehome` returns 1 and writes diagnostic messages to standard error.

Examples EXAMPLE 1 A Sample `.copy_files` File

The files that are listed in `.copy_files` can be modified at every user's label.

```
.cshrc
.mailrc
.mozilla/bookmarks.html
```

EXAMPLE 2 A Sample `.link_files` File

The files that are listed in `.link_files` can be modified at the lowest label. The changes propagate to the other labels that are available to the user.

```
~/bin
.mozilla/preferences
.xrc
.rhosts
```

EXAMPLE 3 Updating the Linked and Copied Files

The `.copy_files` and `.link_files` were updated by the user at the minimum label. At a higher label, the user refreshes the copies and the links. No privileges are required to run the command.

```
% updatehome -r
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability	Stable

Files `$HOME/.copy_files` List of files to be copied
`$HOME/.link_files` List of files to be symbolically linked

See Also `attributes(5)`

“.copy_files and .link_files Files” in *Solaris Trusted Extensions Administrator’s Procedures*

REFERENCE

System Calls

Name getlabel, fgetlabel – get file sensitivity label

Synopsis `cc [flags...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
int getlabel(const char *path, m_label_t *label_p);
```

```
int fgetlabel(int fd, m_label_t *label_p);
```

Description `getlabel()` obtains the sensitivity label of the file that is named by *path*. Discretionary read, write or execute permission to the final component of *path* is not required, but all directories in the path prefix of *path* must be searchable.

`fgetlabel()` obtains the label of an open file that is referred to by the argument descriptor, such as would be obtained by an `open(2)` call.

label_p is a pointer to an opaque label structure. The caller must allocate space for *label_p* by using `m_label_alloc(3TSOL)`.

Return Values `getlabel()` and `fgetlabel()` return:

0 On success.

-1 On failure, and set `errno` to indicate the error.

Errors `getlabel()` fails if one or more of the following are true:

EACCES Search permission is denied for a component of the path prefix of *path*. To override this restriction, the calling process can assert the `PRIV_FILE_DAC_SEARCH` privilege.

EFAULT *label_p* or *path* points to an invalid address.

EIO An I/O error occurred while reading from or writing to the file system.

ELOOP Too many symbolic links were encountered in translating *path*.

ENAMETOOLONG The length of the path argument exceeds `PATH_MAX`.

A pathname component is longer than `NAME_MAX` while `_POSIX_NO_TRUNC` is in effect (see `pathconf(2)`).

ENOENT The file referred to by *path* does not exist.

ENOTDIR A component of the path prefix of *path* is not a directory.

`fgetlabel()` fails if one or more of the following are true:

EBADF *fd* is not a valid open file descriptor.

EFAULT *label_p* points to an invalid address.

EIO An I/O error occurred while reading from or writing to the file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcslr
Interface Stability	Stable

See Also `open(2)`, `pathconf(2)`, `m_label_alloc(3TSOL)`, `attributes(5)`, `labels(5)`

“Obtaining a File Label” in *Solaris Trusted Extensions Developer’s Guide*

REFERENCE

Trusted Extensions Library

Name blcompare, blequal, bldominates, blstrictdom, blinrange – compare binary labels

Synopsis cc [*flag...*] *file...* -ltsol [*library...*]

```
#include <tsol/label.h>

int blequal(const m_label_t *label1, const m_label_t *label2);
int bldominates(const m_label_t *label1, const m_label_t *label2);
int blstrictdom(const m_label_t *label1, const m_label_t *label2);
int blinrange(const m_label_t *label, const brange_t *range);
```

Description These functions compare binary labels for meeting a particular condition.

blequal() compares two labels for equality.

bldominates() compares label *label1* for dominance over label *label2*.

blstrictdom() compares label *label1* for strict dominance over label *label2*.

blinrange() compares label *label* for dominance over *range*→*lower_bound* and *range*→*upper_bound* for dominance over level *label*.

Return Values These functions return non-zero if their respective conditions are met, otherwise zero is returned.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
MT-Level	MT-Safe
Interface Stability	Stable

See Also ucured_getlabel(3C), [getplabel\(3TSOL\)](#), [label_to_str\(3TSOL\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#), [labels\(5\)](#)

“Determining the Relationship Between Two Labels” in *Solaris Trusted Extensions Developer’s Guide*

Name blminmax, blmaximum, blminimum – bound of two labels

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
void blmaximum(m_label_t *maximum_label, const m_label_t *bounding_label);
```

```
void blminimum(m_label_t *minimum_label, const m_label_t *bounding_label);
```

Description `blmaximum()` replaces the contents of label *maximum_label* with the least upper bound of the labels *maximum_label* and *bounding_label*. The least upper bound is the greater of the classifications and all of the compartments of the two labels. This is the least label that dominates both of the original labels.

`blminimum()` replaces the contents of label *minimum_label* with the greatest lower bound of the labels *minimum_label* and *bounding_label*. The greatest lower bound is the lower of the classifications and only the compartments that are contained in both labels. This is the greatest label that is dominated by both of the original labels.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
MT-Level	MT-Safe
Interface	Stable

See Also [label_to_str\(3TSOL\)](#), [sbltos\(3TSOL\)](#), [attributes\(5\)](#)

Name bltocolor, bltocolor_r – get character-coded color name of label

Synopsis cc [*flag...*] *file...* -ltsol [*library...*]

```
#include <tsol/label.h>
```

```
char *bltocolor(const m_label_t *label);
```

```
char *bltocolor_r(const m_label_t *label, const int size, char *color_name);
```

Interface Level The bltocolor() and bltocolor_r() functions are obsolete. Use the [label_to_str\(3TSOL\)](#) function instead.

Description The calling process must have PRIV_SYS_TRANS_LABEL in its set of effective privileges to get color names of labels that dominate the current process's sensitivity label.

bltocolor() and bltocolor_r() get the character-coded color name associated with the binary label *label*.

Return Values bltocolor() returns a pointer to a statically allocated string that contains the character-coded color name specified for the *label* or returns (char *)0 if, for any reason, no character-coded color name is available for this binary label.

bltocolor_r() returns a pointer to the *color_name* string which contains the character-coded color name specified for the *label* or returns (char *)0 if, for any reason, no character-coded color name is available for this binary label. *color_name* must provide for a string of at least *size* characters.

Files /etc/security/tsol/label_encodings

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	Obsolete
MT-Level	MT-Safe with exceptions

See Also [label_to_str\(3TSOL\)](#), [attributes\(5\)](#)

Notes These functions are obsolete and retained for ease of porting. They might be removed in a future Solaris Trusted Extensions release.

The function bltocolor() returns a pointer to a statically allocated string. Subsequent calls to it will overwrite that string with a new character-coded color name. It is not MT-Safe.

For multithreaded applications the function bltocolor_r() should be used.

If *label* includes a specified word or words, the character-coded color name associated with the first word specified in the label encodings file is returned. Otherwise, if no character-coded color name is specified for *label*, the first character-coded color name specified in the label encodings file with the same classification as the binary label is returned.

Name bltos, bsltos, bcleartos – translate binary labels to character coded labels

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
int bsltos(const m_label_t *label, char **string, const int str_len, const int flags);
```

```
int bcleartos(const m_label_t *label, char **string, const int str_len, const int flags);
```

Interface Level The `bsltos()` and `bcleartos()` functions are obsolete. Use the `label_to_str(3TSOL)` function instead.

Description The calling process must have `PRIV_SYS_TRANS_LABEL` in its set of effective privileges to perform label translation on labels that dominate the current process' sensitivity label.

These routines translate binary labels into strings controlled by the value of the *flags* parameter.

The generic form of an output character-coded label is:

```
CLASSIFICATION WORD1 WORD2 WORD3/WORD4 SUFFIX PREFIX WORD5/WORD6
```

Capital letters are used to display all `CLASSIFICATION` names and `WORDS`. The ' ' (space) character separates classifications and words from other words in all character-coded labels except where multiple words that require the same `PREFIX` or `SUFFIX` are present, in which case the multiple words are separated from each other by the '/' (slash) character.

string can point to either a pointer to pre-allocated memory, or the value `(char *)0`. If *string* points to a pointer to pre-allocated memory, then *str_len* indicates the size of that memory. If *string* points to the value `(char *)0`, memory is allocated using `malloc()` to contain the translated character-coded labels. The translated *label* is copied into allocated or pre-allocated memory.

flags is 0 (zero), or the logical sum of the following:

<code>LONG_WORDS</code>	Translate using long names of words defined in <i>label</i> .
<code>SHORT_WORDS</code>	Translate using short names of words defined in <i>label</i> . If no short name is defined in the <code>label_encodings</code> file for a word, the long name is used.
<code>LONG_CLASSIFICATION</code>	Translate using long name of classification defined in <i>label</i> .
<code>SHORT_CLASSIFICATION</code>	Translate using short name of classification defined in <i>label</i> .
<code>ACCESS_RELATED</code>	Translate only <i>access-related</i> entries defined in information label <i>label</i> .
<code>VIEW_EXTERNAL</code>	Translate <code>ADMIN_LOW</code> and <code>ADMIN_HIGH</code> labels to the lowest and highest labels defined in the <code>label_encodings</code> file.

VIEW_INTERNAL Translate `ADMIN_LOW` and `ADMIN_HIGH` labels to the `admin low` name and `admin high` name strings specified in the `label_encodings` file. If no strings are specified, the strings “`ADMIN_LOW`” and “`ADMIN_HIGH`” are used.

NO_CLASSIFICATION Do not translate classification defined in *label*.

`bsltos()` translates a binary sensitivity label into a string. The applicable *flags* are `LONG_CLASSIFICATION` or `SHORT_CLASSIFICATION`, `LONG_WORDS` or `SHORT_WORDS`, `VIEW_EXTERNAL` or `VIEW_INTERNAL`, and `NO_CLASSIFICATION`. A *flags* value `0` is equivalent to `(SHORT_CLASSIFICATION | LONG_WORDS)`.

`bcleartos()` translates a binary clearance into a string. The applicable *flags* are `LONG_CLASSIFICATION` or `SHORT_CLASSIFICATION`, `LONG_WORDS` or `SHORT_WORDS`, `VIEW_EXTERNAL` or `VIEW_INTERNAL`, and `NO_CLASSIFICATION`. A *flags* value `0` is equivalent to `(SHORT_CLASSIFICATION | LONG_WORDS)`. The translation of a clearance might not be the same as the translation of a sensitivity label. These functions use different `label_encodings` file tables that might contain different words and constraints.

Return Values These routines return:

- `-1` If the label is not of the valid defined required type, if the label is not dominated by the process sensitivity label and the process does not have `PRIV_SYS_TRANS_LABEL` in its set of effective privileges, or the `label_encodings` file is inaccessible.
- `0` If memory cannot be allocated for the return string, or the pre-allocated return string memory is insufficient to hold the string. The value of the pre-allocated string is set to the NULL string (`*string[0]='\\00'`);).
- `>0` If successful, the length of the character-coded label including the NULL terminator.

Process Attributes If the `VIEW_EXTERNAL` or `VIEW_INTERNAL` flags are not specified, translation of `ADMIN_LOW` and `ADMIN_HIGH` labels is controlled by the label view process attribute flags. If no label view process attribute flags are defined, their translation is controlled by the label view configured in the `label_encodings` file. A value of `External` specifies that `ADMIN_LOW` and `ADMIN_HIGH` labels are mapped to the lowest and highest labels defined in the `label_encodings` file. A value of `Internal` specifies that the `ADMIN_LOW` and `ADMIN_HIGH` labels are translated to the `admin low` and `admin high` name strings specified in the `label_encodings` file. If no such names are specified, the strings “`ADMIN_LOW`” and “`ADMIN_HIGH`” are used.

Files `/etc/security/tsol/label_encodings`

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability	Obsolete
MT-Level	MT-Safe with exceptions

See Also `free(3C)`, `malloc(3C)`, `label_to_str(3TSOL)`, `label_encodings(4)`, `attributes(5)`

Notes These functions are obsolete and retained for ease of porting. They might be removed in a future Solaris Trusted Extensions release.

If memory is allocated by these routines, the caller must free the memory with `free()` when the memory is no longer in use.

Name btohex, bsltoh, bcleartoh, bsltoh_r, bcleartoh_r, h_alloc, h_free – convert binary label to hexadecimal

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
char *bsltoh(const m_label_t *label);
```

```
char *bcleartoh(const m_label_t *clearance);
```

```
char *bsltoh_r(const m_label_t *label, char *hex);
```

```
char *bcleartoh_r(const m_label_t *clearance, char *hex);
```

```
char *h_alloc(const unsigned char type);
```

```
void h_free(char *hex);
```

Interface Level The `bsltoh()`, `bcleartoh()`, `bsltoh_r()`, `bcleartoh_r()`, `h_alloc()`, and `h_free()` functions are obsolete. Use the [label_to_str\(3TSOL\)](#) function instead.

Description These functions convert binary labels into hexadecimal strings that represent the internal value.

`bsltoh()` and `bsltoh_r()` convert a binary sensitivity label into a string of the form:

```
[0xsensitivity_label_hexadecimal_value]
```

`bcleartoh()` and `bcleartoh_r()` convert a binary clearance into a string of the form:

```
0xclearance_hexadecimal_value
```

`h_alloc()` allocates memory for the hexadecimal value *type* for use by `bsltoh_r()` and `bcleartoh_r()`.

Valid values for *type* are:

`SUN_SL_ID` *label* is a binary sensitivity label.

`SUN_CLR_ID` *label* is a binary clearance.

`h_free()` frees memory allocated by `h_alloc()`.

Return Values These functions return a pointer to a string that contains the result of the translation, or `(char *)0` if the parameter is not of the required type.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	Obsolete
MT-Level	MT-Safe with exceptions

See Also [atohexlabel\(1M\)](#), [hextoalabel\(1M\)](#), [label_to_str\(3TSOL\)](#), [attributes\(5\)](#), [labels\(5\)](#)

Notes These functions are obsolete and retained for ease of porting. They might be removed in a future Solaris Trusted Extensions release.

The functions `bsltoh()` and `bcleartoh()` share the same statically allocated string storage. They are not MT-Safe. Subsequent calls to any of these functions will overwrite that string with the newly translated string.

For multithreaded applications, the functions `bsltoh_r()` and `bcleartoh_r()` should be used.

Name getdevicerange – get the label range of a device

Synopsis `cc [flag...] file... -lbsm -ltsol [library...]`
`#include <tsol/label.h>`
`blrange_t *getdevicerange(const char *device);`

Description The `getdevicerange()` function returns the label range of a user-allocatable device.

If label range is not specified for *device*, `getdevicerange()` returns the default values of `ADMIN_LOW` for the lower bound and `ADMIN_HIGH` for the upper bound of *device*.

From the command line, `list_devices(1)` can be used to see the label range of *device*.

Return Values The `getdevicerange()` function returns `NULL` on failure and sets `errno`. On successful completion, it returns a pointer to a `blrange_t` structure which must be freed by the caller, as follows:

```
blrange_t *range;
...
m_label_free(range->lower_bound);
m_label_free(range->upper_bound);
free(range);
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcslr
Stability	Evolving
MT-Level	MT-Safe

Errors The `getdevicerange()` function fails if:

EAGAIN There is not enough memory available to allocate the required bytes. The application could try later.

ENOMEM The physical limits of the system are exceeded by size bytes of memory which cannot be allocated.

ENOTSUP Invalid upper or lower bound for device.

See Also `list_devices(1)`, `free(3C)`, `m_label_free(3TSOL)`, `attributes(5)`

“Validating the Label Request Against the Printer’s Label Range” in *Solaris Trusted Extensions Developer’s Guide*

Name getpathbylabel – return the zone pathname

Synopsis cc [*flags...*] *file...* -ltsol

```
#include <tsol/label.h>
```

```
char *getpathbylabel(const char *path, char *resolved_path, size_t bufsize,
                    const m_label_t *sl);
```

Description getpathbylabel() expands all symbolic links and resolves references to './', '/../', extra '/' characters, and stores the zone pathname in the buffer named by *resolved_path*. The *bufsize* argument specifies the size in bytes of this buffer. The resulting path will have no symbolic links components, nor any './', '/../'. This function can only be called from the global zone.

The zone pathname is relative to the sensitivity label *sl*. To specify a sensitivity label for a zone name which does not exist, the process must assert either the PRIV_FILE_UPGRADE_SL or PRIV_FILE_DOWNGRADE_SL privilege depending on whether the specified sensitivity label dominates or does not dominate the process sensitivity label.

Return Values getpathbylabel() returns a pointer to the *resolved_path* on success. On failure, it returns NULL and sets *errno* to indicate the error.

Errors	EACCES	Search permission is denied for a component of the path prefix of <i>path</i> .
	EFAULT	<i>resolved_path</i> extends outside the process's allocated address space or beyond <i>bufsize</i> bytes.
	ELOOP	Too many symbolic links were encountered in translating <i>path</i> .
	EINVAL	<i>path</i> or <i>resolved_path</i> was NULL, current zone is not the global zone, or <i>sl</i> is invalid.
	EIO	An I/O error occurred while reading from or writing to the file system.
	ENOENT	The named file does not exist.
	ENAMETOOLONG	The length of the path argument exceeds PATH_MAX. A pathname component is longer than NAME_MAX (see sysconf(3C)) while _POSIX_NO_TRUNC is in effect (see pathconf(2)).

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcslr
MT-Level	MT-Safe
Interface Stability	Stable

See Also readlink(2), getzonerootbyid(3TSOL), attributes(5), labels(5)

Warnings `getpathbylabel()` indirectly invokes the `readlink(2)` system call, and hence inherits the possibility of hanging due to inaccessible file system resources.

Name getlabel – get process label

Synopsis `cc [flag...] file... -ltsol [library...]`
`#include <tsol/label.h>`
`int getlabel(m_label_t *label_p);`

Description `getlabel()` obtains the sensitivity label of the calling process.

Return Values `getlabel()` returns:

- 0 On success.
- 1 On failure, and sets `errno` to indicate the error. `label_p` is unchanged.

Errors `getlabel()` fails (and `label_p` does not refer to a valid sensitivity label) if this condition is true:

EFAULT `label_p` points to an invalid address.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
MT-Level	MT-Safe
Interface Stability	Stable

See Also `ucred_getlabel(3C)`, `m_label_alloc(3TSOL)`, `m_label_free(3TSOL)`, `attributes(5)`

“Obtaining a Process Label” in *Solaris Trusted Extensions Developer’s Guide*

Notes This function returns different values for system processes than `ucred_getlabel(3C)` returns.

Name getuserange – get the label range of a user

Synopsis `cc [flags...] file... -ltsol`

```
#include <tsol/label.h>
```

```
m_range_t *getuserange(const char *username);
```

Description The `getuserange()` function returns the label range of *username*. The lower bound in the range is used as the initial workspace label when a user logs into a multilevel desktop. The upper bound, or clearance, is used as an upper limit to the available labels that a user can assign to labeled workspaces.

The default value for a user's label range is specified in [label_encodings\(4\)](#). Overriding values for individual users are specified in `user_attr(4)`.

Return Values The `getuserange()` function returns NULL if the memory allocation fails. Otherwise, the function returns a structure which must be freed by the caller, as follows:

```
m_range_t *range;
...
m_label_free(range->lower_bound);
m_label_free(range->upper_bound);
free(range);
```

Errors The `getuserange()` function fails if:

ENOMEM The physical limits of the system are exceeded by size bytes of memory which cannot be allocated.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcslr
MT-Level	MT-Safe
Interface Stability	See NOTES below

See Also [free\(3C\)](#), [m_label_free\(3TSOL\)](#), [label_encodings\(4\)](#), [user_attr\(4\)](#), [attributes\(5\)](#)

Notes The stability of this function is Stable for systems that implement the Defense Intelligence Agency (DIA) MAC policy of [label_encodings\(4\)](#). Other policies might exist in a future release of Trusted Extensions that might obsolete or supplement [label_encodings\(4\)](#).

Name getzonelabelbyid, getzonelabelbyname, getzoneidbylabel – map between zones and labels

Synopsis `cc [flags...] file... -ltsol`

```
#include <tsol/label.h>
```

```
m_label_t *getzonelabelbyid(zoneid_t zoneid);
```

```
m_label_t *getzonelabelbyname(const char *zonename);
```

```
zoneid_t *getzoneidbylabel(const m_label_t *label);
```

Description The `getzonelabelbyid()` function returns the mandatory access control (MAC) label of *zoneid*.

The `getzonelabelbyname()` function returns the MAC label of the zone whose name is *zonename*.

The `getzoneidbylabel()` function returns the zone ID of the zone whose label is *label*.

All of these functions require that the specified zone's state is at least `ZONE_IS_READY`. The zone of the calling process must dominate the specified zone's label, or the calling process must be in the global zone.

Return Values On successful completion, the `getzonelabelbyid()` and `getzonelabelbyname()` functions return a pointer to a sensitivity label that is allocated within these functions. To free the storage, use `m_label_free(3TSOL)`. If the zone does not exist, `NULL` is returned.

On successful completion, the `getzoneidbylabel()` function returns the zone ID with the matching label. If there is no matching zone, the function returns `-1`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcslr
MT-Level	Safe
Interface Stability	Stable

Errors The `getzonelabelbyid()` and `getzonelabelbyname()` functions fail if:

`ENOENT` The specified zone does not exist.

The `getzonelabelbyid()` function fails if:

`ENOENT` No zone corresponds to the specified label.

See Also `Intro(2)`, `getzonenamebyid(3C)`, `getzoneidbyname(3C)`, `m_label_free(3TSOL)`, `attributes(5)`, `labels(5)`

Name getzoneroobid, getzoneroobylabel, getzoneroobbyname – map between zone root pathnames and labels

Synopsis `cc [flags...] file... -ltsol`
`#include <tsol/label.h>`
`char *getzoneroobid(zoneid_t zoneid);`
`char *getzoneroobylabel(const m_label_t *label);`
`char *getzoneroobbyname(const char *zonename);`

Description The `getzoneroobid()` function returns the root pathname of *zoneid*.

The `getzoneroobylabel()` function returns the root pathname of the zone whose label is *label*.

The `getzoneroobbyname()` function returns the root pathname of *zonename*.

All of these functions require that the specified zone's state is at least `ZONE_IS_READY`. The zone of the calling process must dominate the specified zone's label, or the calling process must be in the global zone. The returned pathname is relative to the root path of the caller's zone.

Return Values On successful completion, the `getzoneroobid()`, `getzoneroobylabel()`, and `getzoneroobbyname()` functions return a pointer to a pathname that is allocated within these functions. To free the storage, use `free(3C)`. On failure, these functions return `NULL` and set `errno` to indicate the error.

Errors

<code>EINVAL</code>	<i>zoneid</i> invalid, or zone not found or not ready.
<code>EFAULT</code>	Invalid argument; pointer location is invalid.
<code>ENOMEM</code>	Unable to allocate pathname.
<code>ENOENT</code>	Zone does not exist.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcslr
MT-Level	Safe
Interface Stability	Stable

See Also `Intro(2)`, `free(3C)`, `getzonenamebyid(3C)`, `attributes(5)`, `labels(5)`

Name hextob, htobsl, htobclear – convert hexadecimal string to binary label

Synopsis `cc [flag...] file... -ltsol [library...]`
`#include <tsol/label.h>`
`int htobsl(const char *s, m_label_t *label);`
`int htobclear(const char *s, m_label_t *clearance);`

Interface Level The `htobsl()` and `htobclear()` functions are obsolete. Use the [str_to_label\(3TSOL\)](#) function instead.

Description These functions convert hexadecimal string representations of internal label values into binary labels.

`htobsl()` converts into a binary sensitivity label, a hexadecimal string of the form:

`0xsensitivity_label_hexadecimal_value`

`htobclear()` converts into a binary clearance, a hexadecimal string of the form:

`0xclearance_hexadecimal_value`

Return Values These functions return non-zero if the conversion was successful, otherwise zero is returned.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	Obsolete
MT-Level	MT-Safe

See Also [str_to_label\(3TSOL\)](#), [attributes\(5\)](#), [labels\(5\)](#)

Notes These functions are obsolete and retained for ease of porting. They might be removed in a future Solaris Trusted Extensions release.

Name labelbuilder, tsol_lbuild_create, tsol_lbuild_get, tsol_lbuild_set, tsol_lbuild_destroy – create a Motif-based user interface for interactively building a valid label or clearance

Synopsis `cc [flag...] file... -ltsol -lDtTsol [library...]`

```
#include <Dt/ModLabel.h>
```

```
ModLabelData *tsol_lbuild_create(Widget widget void (*event_handler)() ok_callback
    lbuild_attributes extended_operation, ..., NULL);
```

```
void *tsol_lbuild_get(ModLabelData *data, lbuild_attributes extended_operation);
```

```
void tsol_lbuild_set(ModLabelData *data lbuild_attributes extended_operation, ...,
    NULL);
```

```
void tsol_lbuild_destroy(ModLabelData *data);
```

Description The label builder user interface prompts the end user for information and generates a valid sensitivity label or clearance from the user input based on specifications in the [label_encodings\(4\)](#) file on the system where the application runs. The end user can build the label or clearance by typing a text value or by interactively choosing options.

Application-specific functionality is implemented in the callback for the OK pushbutton. This callback is passed to the `tsol_lbuild_create()` call where it is mapped to the OK pushbutton widget.

When choosing options, the label builder shows the user only those classifications (and related compartments and markings) dominated by the workspace sensitivity label unless the executable has the `PRIV_SYS_TRANS_LABEL` privilege in its effective set.

If the end user does not have the authorization to upgrade or downgrade labels, or if the user-built label is out of the user's accreditation range, the OK and Reset pushbuttons are grayed. There are no privileges to override these restrictions.

`tsol_lbuild_create()` creates the graphical user interface and returns a pointer variable of type `ModLabelData*` that contains information on the user interface. This information is a combination of values passed in the `tsol_lbuild_create()` input parameter list, default values for information not provided, and information on the widgets used by the label builder to create the user interface. All information except the widget information should be accessed with the `tsol_lbuild_get()` and `tsol_lbuild_set()` routines.

The widget information is accessed directly by referencing the following fields of the `ModLabelData` structure.

`lbuild_dialog` The label builder dialog box.

`ok` The OK pushbutton.

`cancel` The Cancel pushbutton.

`reset` The Reset pushbutton.

`help` The Help pushbutton.

The `tsol_lbuild_create()` parameter list takes the following values:

`widget` The widget from which the dialog box is created. Any Motif widget can be passed.

`ok_callback` A callback function that implements the behavior of the OK pushbutton on the dialog box.

`..., NULL` A NULL terminated list of extended operations and value pairs that define the characteristics and behavior of the label builder dialog box.

`tsol_lbuild_destroy()` destroys the `ModLabelData` structure returned by `tsol_lbuild_create()`.

`tsol_lbuild_get()` and `tsol_lbuild_set()` access the information stored in the `ModLabelData` structure returned by `tsol_lbuild_create()`.

The following extended operations can be passed to `tsol_lbuild_create()` to build the user interface, to `tsol_lbuild_get()` to retrieve information on the user interface, and to `tsol_lbuild_set()` to change the user interface information. All extended operations are valid for `tsol_lbuild_get()`, but the *WORK* operations are not valid for `tsol_lbuild_set()` or `tsol_lbuild_create()` because these values are set from input supplied by the end user. These exceptions are noted in the descriptions.

`LBUILD_MODE` Create a user interface to build a sensitivity label or a clearance. Value is `LBUILD_MODE_SL` by default.

`LBUILD_MODE_SL` Build a sensitivity label.

`LBUILD_MODE_CLR` Build a clearance.

`LBUILD_VALUE_SL` The starting sensitivity label. This value is `ADMIN_LOW` by default and is used when the mode is `LBUILD_MODE_SL`.

`LBUILD_VALUE_CLR` The starting clearance. This value is `ADMIN_LOW` by default and is used when the mode is `LBUILD_MODE_CLR`.

`LBUILD_USERFIELD` A character string prompt that displays at the top of the label builder dialog box. Value is `NULL` by default.

`LBUILD_SHOW` Show or hide the label builder dialog box. Value is `FALSE` by default.

`TRUE` Show the label builder dialog box.

`FALSE` Hide the label builder dialog box.

`LBUILD_TITLE` A character string title that appears at the top of the label builder dialog box. Value is `NULL` by default.

LBUILD_WORK_SL	Not valid for <code>tsol_lbuild_set()</code> or <code>tsol_lbuild_create()</code> . The sensitivity label the end user is building. Value is updated to the end user's input when the end user selects the Update pushbutton or interactively chooses an option.
LBUILD_WORK_CLR	Not valid for <code>tsol_lbuild_set()</code> or <code>tsol_lbuild_create()</code> . The clearance the end user is building. Value is updated to the end user's input when the end user selects the Update pushbutton or interactively chooses an option.
LBUILD_X	The X position in pixels of the top-left corner of the label builder dialog box in relation to the top-left corner of the screen. By default the label builder dialog box is positioned in the middle of the screen.
LBUILD_Y	The Y position in pixels of the top-left corner of the label builder dialog box in relation to the top-left corner of the screen. By default the label builder dialog box is positioned in the middle of the screen.
LBUILD_LOWER_BOUND	The lowest classification (and related compartments and markings) available to the user as radio buttons for interactively building a label or clearance. This value is the user's minimum label.
LBUILD_UPPER_BOUND	The highest classification (and related compartments and markings) available to the user as radio buttons for interactively building a label or clearance. A supplied value should be within the user's accreditation range. If no value is specified, the value is the user's workspace sensitivity label, or if the executable has the <code>PRIV_SYS_TRANS_LABEL</code> privilege, the value is the user's clearance.
LBUILD_CHECK_AR	Check that the user-built label entered in the Update With field is within the user's accreditation range. A value of 1 means check, and a value of 0 means do not check. If checking is on and the label is out of range, an error message is raised to the end user.
LBUILD_VIEW	Use the internal or external label representation. Value is <code>LBUILD_VIEW_EXTERNAL</code> by default.
LBUILD_VIEW_INTERNAL	Use the internal names for the highest and lowest labels in the system: <code>ADMIN_HIGH</code> and <code>ADMIN_LOW</code> .
LBUILD_VIEW_EXTERNAL	Promote an <code>ADMIN_LOW</code> label to the next highest label, and demote an <code>ADMIN_HIGH</code> label to the next lowest label.

Return Values The `tsol_lbuid_get()` returns `-1` if it is unable to get the value.

The `tsol_lbuid_create()` routine returns a variable of type `ModLabelData` that contains the information provided in the `tsol_lbuid_create()` input parameter list, default values for information not provided, and information on the widgets used by the label builder to create the user interface.

Examples EXAMPLE 1 To Create a Label Builder

```
(ModLabelData *)lbldata = tsol_lbuid_create(widget0, callback_function,
    LBUILD_MODE, LBUILD_MODE_SL,
    LBUILD_TITLE, "Setting Sensitivity Label",
    LBUILD_VIEW, LBUILD_VIEW_INTERNAL,
    LBUILD_X, 200,
    LBUILD_Y, 200,
    LBUILD_USERFIELD, "Pathname:",
    LBUILD_SHOW, FALSE,
    NULL);
```

EXAMPLE 2 To Query the Mode and Display the Label Builder

These examples call the `tsol_lbuid_get()` routine to query the mode being used, and call the `tsol_lbuid_set()` routine so the label builder dialog box displays.

```
mode = (int)tsol_lbuid_get(lbldata, LBUILD_MODE );

tsol_lbuid_set(lbldata, LBUILD_SHOW, TRUE, NULL);
```

EXAMPLE 3 To Destroy the ModLabelData Variable

This example destroys the `ModLabelData` variable returned in the call to `tsol_lbuid_create()`.

```
tsol_lbuid_destroy(lbldata);
```

Files `/usr/dt/include/Dt/ModLabel.h`
Header file for label builder functions

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
MT-Level	MT-Safe

See Also [label_encodings\(4\)](#), [attributes\(5\)](#)

Chapter 7, “Label Builder APIs,” in *Solaris Trusted Extensions Developer’s Guide*

Name labelclipping, Xbsltos, Xbcleartos – translate a binary label and clip to the specified width

Synopsis `cc [flag...] file... -ltsol -ldttsol [library...]`

```
#include <Dt/label_clipping.h>
```

```
XmString Xbsltos(Display *display, const m_label_t *senslabel, Dimension width,
               const XmFontList fontlist, const int flags);
```

```
XmString Xbcleartos(Display *display, const m_label_t *clearance,
                   Dimension width, const XmFontList fontlist, const int flags);
```

Interface Level The labelclipping functions, `Xbsltos()` and `Xbcleartos()`, are obsolete. Use the [label_to_str\(3TSOL\)](#) function instead.

Description The calling process must have `PRIV_SYS_TRANS_LABEL` in its set of effective privileges to translate labels or clearances that dominate the current process' sensitivity label.

display The structure controlling the connection to an X Window System display.

senslabel The sensitivity label to be translated.

clearance The clearance to be translated.

width The width of the translated label or clearance in pixels. If the specified width is shorter than the full label, the label is clipped and the presence of clipped letters is indicated by an arrow. In this example, letters have been clipped to the right of: TS<-. See the [sbltos\(3TSOL\)](#) man page for more information on the clipped indicator. If the specified width is equal to the display width (*display*), the label is not truncated, but word-wrapped using a width of half the display width.

fontlist A list of fonts and character sets where each font is associated with a character set.

flags The value of flags indicates which words in the [label_encodings\(4\)](#) file are used for the translation. See the [bltos\(3TSOL\)](#) man page for a description of the flag values: `LONG_WORDS`, `SHORT_WORDS`, `LONG_CLASSIFICATION`, `SHORT_CLASSIFICATION`, `ALL_ENTRIES`, `ACCESS_RELATED`, `VIEW_EXTERNAL`, `VIEW_INTERNAL`, `NO_CLASSIFICATION`. `BRACKETED` is an additional flag that can be used with `Xbsltos()` only. It encloses the sensitivity label in square brackets as follows: [C].

Return Values These interfaces return a compound string that represents the character-coded form of the sensitivity label or clearance that is translated. The compound string uses the language and fonts specified in *fontlist* and is clipped to *width*. These interfaces return `NULL` if the label or clearance is not a valid, required type as defined in the [label_encodings\(4\)](#) file, or not dominated by the process' sensitivity label and the `PRIV_SYS_TRANS_LABEL` privilege is not asserted.

Files /usr/dt/include/Dt/label_clipping.h

Header file for label clipping functions

/etc/security/tsol/label_encodings

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Examples EXAMPLE 1 To Translate and Clip a Clearance

This example translates a clearance to text using the long words specified in the [label_encodings\(4\)](#) file, a font list, and clips the translated clearance to a width of 72 pixels.

```
xmstr = Xbcleartos(XtDisplay(topLevel),
&clearance, 72, fontlist, LONG_WORDS
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
MT-Level	MT-Safe

See Also [bltos\(3TSOL\)](#), [label_to_str\(3TSOL\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#)

See [XmStringDraw\(3\)](#) and [FontList\(3\)](#) for information on the creation and structure of a font list.

Name label_to_str – convert labels to human readable strings

Synopsis cc [*flag...*] *file...* -ltsol [*library...*]

```
#include <tsol/label.h>
```

```
int label_to_str(const m_label_t *label, char **string,
                const m_label_str_t conversion_type, uint_t flags);
```

Description label_to_str() is a simple function to convert various mandatory label types to human readable strings.

label is the mandatory label to convert. *string* points to memory that is allocated by label_to_str() that contains the converted string. The caller is responsible for calling free(3C) to free allocated memory.

The calling process must have mandatory read access to the resulting human readable string. Or the calling process must have the sys_trans_label privilege.

The *conversion_type* parameter controls the type of label conversion. Not all types of conversion are valid for all types of label:

M_LABEL	Converts <i>label</i> to a human readable string based on its type.
M_INTERNAL	Converts <i>label</i> to an internal text representation that is safe for storing in a public object. Internal conversions can later be parsed to their same value.
M_COLOR	Converts <i>label</i> to a string that represents the color name that the administrator has associated with the label.
PRINTER_TOP_BOTTOM	Converts <i>label</i> to a human readable string that is appropriate for use as the top and bottom label of banner and trailer pages in the Defense Intelligence Agency (DIA) encodings printed output schema.
PRINTER_LABEL	Converts <i>label</i> to a human readable string that is appropriate for use as the banner page downgrade warning in the DIA encodings printed output schema.
PRINTER_CAVEATS	Converts <i>label</i> to a human readable string that is appropriate for use as the banner page caveats section in the DIA encodings printed output schema.
PRINTER_CHANNELS	Converts <i>label</i> to a human readable string that is appropriate for use as the banner page handling channels in the DIA encodings printed output schema.

The *flags* parameter provides a hint to the label conversion:

DEF_NAMES The default names are preferred.

SHORT_NAMES Short names are preferred where defined.

LONG_NAMES Long names are preferred.

Return Values Upon successful completion, the `label_to_str()` function returns zero (0). Otherwise, -1 is returned, `errno` is set to indicate the error and the string pointer is set to `NULL`.

Errors The `label_to_str()` function fails if:

EINVAL Invalid parameter.

ENOTSUP The system does not support label translations.

ENOMEM The physical limits of the system are exceeded by size bytes of memory which cannot be allocated.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
MT-Level	MT-Safe
Interface Stability	See NOTES and WARNINGS below

See Also `free(3C)`, `str_to_label(3TSOL)`, `label_encodings(4)`, `attributes(5)`, `labels(5)`

“Using the `label_to_str` Function” in *Solaris Trusted Extensions Developer’s Guide*

Notes `label_to_str()` is Stable. Conversion types that are relative to the DIA encodings schema are Standard. Standard is specified in `label_encodings(4)`. The returned string is Undefined and is dependent on the specific `label_encodings` file. The conversion type `INTERNAL` is Unstable, but is always accepted as input to `str_to_label(3TSOL)`.

Warnings A number of these conversions rely on the DIA label encodings schema. They might not be valid for other label schemata.

Name m_label, m_label_alloc, m_label_dup, m_label_free – m_label functions

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>

m_label_t *m_label_alloc(const m_label_type_t label_type);
int m_label_dup(m_label_t **dst, const m_label_t *src);
void m_label_free(m_label_t *label);
```

Description The `m_label_alloc()` function allocates resources for a new label. `label_type` defines the type for a newly allocated label. The label type can be:

`MAC_LABEL` A Mandatory Access Control (MAC) label.
`USER_CLEAR` A user clearance.

The `m_label_dup()` function allocates resources for a new `dst` label. The function returns a pointer to the allocated label, which is an exact copy of the `src` label. The caller is responsible for freeing the allocated resources by calling `m_label_free()`.

The `m_label_free()` function frees resources that are associated with the previously allocated label.

Return Values Upon successful completion, the `m_label_alloc()` function returns a pointer to the newly allocated label. Otherwise, `m_label_alloc()` returns `NULL` and `errno` is set to indicate the error.

Upon successful completion, the `m_label_dup()` function returns zero (0). Otherwise, -1 is returned and `errno` is set to indicate the error.

Errors `EINVAL` Invalid parameter.
`ENOMEM` The physical limits of the system are exceeded by size bytes of memory which cannot be allocated.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Safe
Interface Stability	Stable

See Also `label_to_str(3TSOL)`, `str_to_label(3TSOL)`, `label_encodings(4)`, `attributes(5)`, `labels(5)`

“Determining Whether the Printing Service Is Running in a Labeled Environment” in *Solaris Trusted Extensions Developer’s Guide*

Name sbltos, sbsltos, sbcleartos – translate binary labels to canonical character-coded labels

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
char *sbsltos(const m_label_t *label, const int len);
```

```
char *sbcleartos(const m_label_t *clearance, const int len);
```

Interface Level The `sbsltos()` and `sbcleartos()` functions are obsolete. Use the [label_to_str\(3TSOL\)](#) function instead.

Description The calling process must have `PRIV_SYS_TRANS_LABEL` in its set of effective privileges to perform label translation on labels that dominate the current process's sensitivity label.

These functions translate binary labels into canonical strings that are clipped to the number of printable characters specified in *len*. Clipping is required if the number of characters of the translated string is greater than *len*. Clipping is done by truncating the label on the right to two characters less than the specified number of characters. A clipped indicator, “<–”, is appended to sensitivity labels and clearances. The character-coded label begins with a classification name separated with a single space character from the list of words making up the remainder of the label. The binary labels must be of the proper defined type and dominated by the process's sensitivity label. A *len* of 0 (zero) returns the entire string with no clipping.

`sbsltos()` translates a binary sensitivity label into a clipped string using the long form of the words and the short form of the classification name. If *len* is less than the minimum number of characters (three), the translation fails.

`sbcleartos()` translates a binary clearance into a clipped string using the long form of the words and the short form of the classification name. If *len* is less than the minimum number of characters (three), the translation fails. The translation of a clearance might not be the same as the translation of a sensitivity label. These functions use different tables of the `label_encodings` file which might contain different words and constraints.

Return Values These routines return a pointer to a statically allocated string that contains the result of the translation, or `(char *)0` if the translation fails for any reason.

Examples

`sbsltos()` Assume that a sensitivity label is:

```
UN TOP/MIDDLE/LOWER DRAWER
```

When clipped to ten characters it is:

```
UN TOP/M<–
```

`sbcleartos()` Assume that a clearance is:

```
UN TOP/MIDDLE/LOWER DRAWER
```

When clipped to ten characters it is:

```
UN TOP/M<-
```

Process Attributes If the `VIEW_EXTERNAL` or `VIEW_INTERNAL` flags are not specified, translation of `ADMIN_LOW` and `ADMIN_HIGH` labels is controlled by the label view process attribute flags. If no label view process attribute flags are defined, their translation is controlled by the label view configured in the `label_encodings` file. A value of `External` specifies that `ADMIN_LOW` and `ADMIN_HIGH` labels are mapped to the lowest and highest labels defined in the `label_encodings` file. A value of `Internal` specifies that the `ADMIN_LOW` and `ADMIN_HIGH` labels are translated to the `admin low` name and `admin high` name strings specified in the `label_encodings` file. If no such names are specified, the strings “`ADMIN_LOW`” and “`ADMIN_HIGH`” are used.

Files `/etc/security/tsol/label_encodings`

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability	Obsolete
MT-Level	Unsafe

See Also [label_to_str\(3TSOL\)](#), [attributes\(5\)](#), [labels\(5\)](#)

Notes These functions are obsolete and retained for ease of porting. They might be removed in a future Solaris Trusted Extensions release.

Warnings All these functions share the same statically allocated string storage. They are not MT-Safe. Subsequent calls to any of these functions will overwrite that string with the newly translated string.

Name setflabel – move file to zone with corresponding sensitivity label

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
int setflabel(const char *path, const m_label_t *label_p);
```

Description The file that is named by *path* is relabeled by moving it to a new pathname relative to the root directory of the zone corresponding to *label_p*. If the source and destination file systems are loopback mounted from the same underlying file system, the file is renamed. Otherwise, the file is copied and removed from the source directory.

The following policy checks are enforced by this function:

- If the sensitivity label of *label_p* equals the existing sensitivity label, then the file is not moved.
- If the corresponding directory does not exist in the destination zone, or if the directory exists, but has a different label than *label_p*, the file is not moved. Also, if the file already exists in the destination directory, the file is not moved.
- If the sensitivity label of the existing file is not equal to the calling process label and the caller is not in the global zone, then the file is not moved. If the caller is in the global zone, the existing file label must be in a labeled zone (not ADMIN_LOW or ADMIN_HIGH).
- If the calling process does not have write access to both the source and destination directories, then the calling process must have PRIV_FILE_DAC_WRITE in its set of effective privileges.
- If the sensitivity label of *label_p* provides read only access to the existing sensitivity label (an upgrade), then the user must have the `solaris.label.file.upgrade` authorization. In addition, if the current zone is a labeled zone, then it must have been assigned the privilege PRIV_FILE_UPGRADE_SL when the zone was configured.
- If the sensitivity label of *label_p* does not provide access to the existing sensitivity label (a downgrade), then the calling user must have the `solaris.label.file.downgrade` authorization. In addition, if the current zone is a labeled zone, then it must have been assigned the privilege PRIV_FILE_DOWNGRADE_SL when the zone was configured.
- If the calling process is not in the global zone, and the user does not have the `solaris.label.range` authorization, then *label_p* must be within the user's label range and within the system accreditation range.
- If the existing file is in use (not tranquil) it is not moved. This tranquility check does not cover race conditions nor remote file access.

Additional policy constraints can be implemented by customizing the shell script `/etc/security/tsol/relabel`. See the comments in this file.

Return Values `setflabel()` returns:

0 On success.

–1 On failure, and sets `errno` to indicate the error.

Errors `setlabel()` fails and the file is unchanged if any of these conditions prevails:

EACCES	Search permission is denied for a component of the path prefix of <i>path</i> . The calling process does not have mandatory write access to the final component of path because the sensitivity label of the final component of path does not dominate the sensitivity label of the calling process and the calling process does not have <code>PRIV_FILE_MAC_WRITE</code> in its set of effective privileges.
EBUSY	There is an open file descriptor reference to the final component of <i>path</i> .
ECONNREFUSED	A connection to the label daemon could not be established.
EEXIST	A file with the same name exists in the destination directory.
EINVAL	Improper parameters were received by the label daemon.
EISDIR	The existing file is a directory.
ELOOP	Too many symbolic links were encountered in translating <i>path</i> .
EMLINK	The existing file is hardlinked to another file.
ENAMETOOLONG	The length of the path argument exceeds <code>PATH_MAX</code> .
ENOENT	The file referred to by <i>path</i> does not exist.
EROFS	The file system is read-only or its label is <code>ADMIN_LOW</code> or <code>ADMIN_HIGH</code> .

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTETYPE	ATTRIBUTEVALUE
Availability	SUNWcslr
MT-Level	MT-Safe
Interface Stability	Stable

See Also `attributes(5)`

“Setting a File Sensitivity Label” in *Solaris Trusted Extensions Developer’s Guide*

Name stobl, stobsl, stobclear – translate character-coded labels to binary labels

Synopsis `cc [flag...] file... -ltsol [library...]`

```
#include <tsol/label.h>
```

```
int stobsl(const char *string, m_label_t *label, const int flags, int *error);
```

```
int stobclear(const char *string, m_label_t *clearance, const int flags,
              int *error);
```

Interface Level The `stobsl()` and `stobclear()` functions are obsolete. Use the [str_to_label\(3TSOL\)](#) function instead.

Description The calling process must have `PRIV_SYS_TRANS_LABEL` in its set of effective privileges to perform label translation on character-coded labels that dominate the process's sensitivity label.

The `stobl` functions translate character-coded labels into binary labels. They also modify an existing binary label by incrementing or decrementing it to produce a new binary label relative to its existing value.

The generic form of an input character-coded label string is:

```
[ + ] classification name [ [ + | - ] word ...
```

Leading and trailing white space is ignored. Fields are separated by white space, a '/' (slash), or a ',' (comma). Case is irrelevant. If *string* starts with + or –, *string* is interpreted a modification to an existing label. If *string* starts with a classification name followed by a + or –, the new classification is used and the rest of the old label is retained and modified as specified by *string*. + modifies an existing label by adding words. – modifies an existing label by removing words. To the maximum extent possible, errors in *string* are corrected in the resulting binary label *label*.

The `stobl` functions also translate hexadecimal label representations into binary labels (see [hextob\(3TSOL\)](#)) when the string starts with `0x` and either `NEW_LABEL` or `NO_CORRECTION` is specified in *flags*.

flags can be the following:

`NEW_LABEL` *label* contents is not used, is formatted as a label of the relevant type, and is assumed to be `ADMIN_LOW` for modification changes. If `NEW_LABEL` is not present, *label* is validated as a defined label of the correct type dominated by the process's sensitivity label.

`NO_CORRECTION` No corrections are made if there are errors in the character-coded label *string*. *string* must be complete and contain all the label components that are required by the `label_encodings` file. The `NO_CORRECTION` flag implies the `NEW_LABEL` flag.

0 (zero) The default action is taken.

`error` is a return parameter that is set only if the function is unsuccessful.

`stobl()` translates the character-coded sensitivity label string into a binary sensitivity label and places the result in the return parameter *label*.

flags can be either `NEW_LABEL`, `NO_CORRECTION`, or 0 (zero). Unless `NO_CORRECTION` is specified, this translation forces the label to dominate the minimum classification, and initial compartments set that is specified in the `label_encodings` file and corrects the label to include other label components required by the `label_encodings` file, but not present in *string*.

`stobclear()` translates the character-coded clearance string into a binary clearance and places the result in the return parameter *clearance*.

flags can be either `NEW_LABEL`, `NO_CORRECTION`, or 0 (zero). Unless `NO_CORRECTION` is specified, this translation forces the label to dominate the minimum classification, and initial compartments set that is specified in the `label_encodings` file and corrects the label to include other label components that are required by the `label_encodings` file, but not present in *string*. The translation of a clearance might not be the same as the translation of a sensitivity label. These functions use different tables of the `label_encodings` file that might contain different words and constraints.

Return Values These functions return:

- 1 If the translation was successful and a valid binary label was returned.
- 0 If an error occurred. `error` indicates the type of error.

Errors When these functions return zero, `error` contains one of the following values:

- 1 Unable to access the `label_encodings` file.
- 0 The label *label* is not valid for this translation and the `NEW_LABEL` or `NO_CORRECTION` flag was not specified, or the label *label* is not dominated by the process's *sensitivity label* and the process does not have `PRIV_SYS_TRANS_LABEL` in its set of effective privileges.
- >0 The character-coded label *string* is in error. `error` is a one-based index into *string* indicating where the translation error occurred.

Files `/etc/security/tsol/label_encodings`

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Stability Level	Obsolete
MT-Level	MT-Safe

See Also `blcompare(3TSOL)`, `hextob(3TSOL)`, `str_to_label(3TSOL)`, `attributes(5)`

Notes These functions are obsolete and are retained for ease of porting. They might be removed in a future release of Solaris Trusted Extensions.

In addition to the `ADMIN_LOW` name and `ADMIN_HIGH` name strings defined in the `label_encodings` file, the strings “`ADMIN_LOW`” and “`ADMIN_HIGH`” are always accepted as character-coded labels to be translated to the appropriate `ADMIN_LOW` and `ADMIN_HIGH` label, respectively.

Modifying an existing `ADMIN_LOW` label acts as the specification of a `NEW_LABEL` and forces the label to start at the minimum label that is specified in the `label_encodings` file.

Modifying an existing `ADMIN_HIGH` label is treated as an attempt to change a label that represents the highest defined classification and all the defined compartments that are specified in the `label_encodings` file.

The `NO_CORRECTION` flag is used when the character-coded label must be complete and accurate so that translation to and from the binary form results in an equivalent character-coded label.

Name str_to_label – parse human readable strings to label

Synopsis cc [*flag...*] *file...* -ltsol [*library...*]

```
#include <tsol/label.h>
```

```
int str_to_label(const char *string, m_label_t **label,
                const m_label_type_t label_type, uint_t flags, int *error);
```

Description str_to_label() is a simple function to parse human readable strings into labels of the requested type.

string is the string to parse. If *string* is the result of a label_to_str() conversion of type M_INTERNAL, *flags* are ignored, and any previously parsed label is replaced.

If **label* is NULL, str_to_label() allocates resources for *label* and initializes the label to the *label_type* that was requested before parsing *string*.

If **label* is not NULL, the label is a pointer to a mandatory label that is the result of a previously parsed label and *label_type* is ignored. The type that is used for parsing is derived from *label* for any type-sensitive operations.

If *flags* is L_MODIFY_EXISTING, the parsed string can be used to modify this label.

If *flags* is L_NO_CORRECTION, the previously parsed label is replaced and the parsing algorithm does not attempt to infer missing elements from string to compose a valid label.

If *flags* is L_DEFAULT, the previously parsed label is replaced and the parsing algorithm makes a best effort to imply a valid label from the elements of *string*.

The caller is responsible for freeing the allocated resources by calling the m_label_free() function. *label_type* defines the type for a newly allocated label. The label type can be:

MAC_LABEL The string should be translated as a Mandatory Access Control (MAC) label.

USER_CLEAR The string should be translated as a label that represents the least upper bound of the labels that the user is allowed to access.

If *error* is NULL, do not return additional error information for EINVAL. The calling process must have mandatory read access to *label* and human readable *string*. Or the calling process must have the sys_trans_label privilege.

The manifest constants ADMIN_HIGH and ADMIN_LOW are the human readable strings that correspond to the Trusted Extensions policy admin_high and admin_low label values. See [labels\(5\)](#).

Return Values Upon successful completion, the str_to_label() function returns zero (0). Otherwise, -1 is returned, errno is set to indicate the error, and *error* provides additional information for EINVAL. Otherwise, *error* is a zero-based index to the string parse failure point.

Errors The str_to_label() function fails if:

- EINVAL** Invalid parameter. `M_BAD_STRING` indicates that *string* could not be parsed. `M_BAD_LABEL` indicates that the label passed in was in error.
- ENOTSUP** The system does not support label translations.
- ENOMEM** The physical limits of the system are exceeded by size bytes of memory which cannot be allocated.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe
Interface Stability	See NOTES and WARNINGS below

See Also [label_to_str\(3TSOL\)](#), [m_label\(3TSOL\)](#), [label_encodings\(4\)](#), [attributes\(5\)](#), [labels\(5\)](#)

“Validating the Label Request Against the Printer’s Label Range” in *Solaris Trusted Extensions Developer’s Guide*

Notes `str_to_label()` is Stable. Parsing types that are relative to Defense Intelligence Agency (DIA) encodings schema are Standard. Standard is specified in [label_encodings\(4\)](#).

Warnings A number of the parsing rules rely on the DIA label encodings schema. The rules might not be valid for other label schemata.

Name tsol_getrhtype – get trusted network host type

Synopsis cc [*flag...*] *file...* -ltsnet [*library...*]

```
#include <libtsnet.h>
```

```
tsol_host_type_t tsol_getrhtype(char *hostname);
```

Description The `tsol_getrhtype()` function queries the kernel-level network information to determine the host type that is associated with the specified *hostname*. The *hostname* can be a regular hostname, an IP address, or a network wildcard address.

Return Values The returned value will be one of the enumerated types that is defined in the `tsol_host_type_t` typedef. Currently these types are UNLABELED and SUN_CIPSO.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl
MT-Level	MT-Safe

Files /etc/security/tsol/tnrhdb Trusted network remote-host database

See Also [tnrhdb\(4\)](#), [attributes\(5\)](#)

“Obtaining the Remote Host Type” in *Solaris Trusted Extensions Developer’s Guide*

REFERENCE

X Library Extensions

Name XTSOLgetClientAttributes – get all label attributes associated with a client

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetClientAttributes(display, windowid, clientattr);
```

```
Display *display;
XID windowid;
XTsolClientAttributes *clientattrp;
```

Description XTSOLgetClientAttributes() is used to get all label attributes that are associated with a client in a single call. The attributes include process ID, user ID, IP address, audit flags and session ID.

Parameters *display* Specifies a pointer to the Display structure. Is returned from XOpenDisplay().

windowid Specifies window ID of X client.

clientattrp Client must provide a pointer to an XTsolClientAttributes structure.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege

BadValue Not a valid client

See Also [XTSOLgetPropAttributes\(3XTSOL\)](#), [XTSOLgetResAttributes\(3XTSOL\)](#)

Name XTSOLgetPropAttributes – get the label attributes associated with a property hanging on a window

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetPropAttributes(display, window, property, propattrp);
```

```
Display *display;
Window window;
Atom property;
XTSOLPropAttributes *propattrp;
```

Description The client requires the PRIV_WIN_DAC_READ and PRIV_WIN_MAC_READ privileges. XTSOLgetPropAttributes() is used to get the label attributes that are associated with a property hanging out of a window in a single call. The attributes include UID and sensitivity label.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- window* Specifies the ID of a window system object.
- property* Specifies the property atom.
- propattrp* Client must provide a pointer to XTSOLPropAttributes.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadAtom Not a valid atom

See Also [XTSOLgetClientAttributes\(3XTSOL\)](#), [XTSOLgetResAttributes\(3XTSOL\)](#)

“Setting Window Polyinstantiation Information” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetPropLabel – get the label associated with a property hanging on a window

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetPropLabel(display, window, property, sl);
```

```
Display *display;
Window window;
Atom property;
m_label_t *sl;
```

Description Client requires the PRIV_WIN_DAC_READ and PRIV_WIN_MAC_READ privileges. XTSOLgetPropLabel() is used to get the sensitivity label that is associated with a property hanging on a window.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- window* Specifies the ID of the window whose property's label you want to get.
- property* Specifies the property atom.
- sl* Returns a sensitivity label that is the current label of the specified property.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadAtom Not a valid atom

See Also [XTSOLgetPropAttributes\(3XTSOL\)](#), [XTSOLsetPropLabel\(3XTSOL\)](#)

“Setting Window Polyinstantiation Information” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetPropUID – get the UID associated with a property hanging on a window

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetPropUID (display, window, property, uidp);
Display *display;
Window window;
Atom property;
uid_t *uidp;
```

Description The client requires the PRIV_WIN_DAC_READ and PRIV_WIN_MAC_READ privileges. XTSOLgetPropUID() gets the ownership of a window's property. This allows a client to get the ownership of an object it did not create.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- window* Specifies the ID of the window whose property's UID you want to get.
- property* Specifies the property atom.
- uidp* Returns a UID which is the current UID of the specified property. Client needs to provide a uid_t type storage and passes the address of this storage as the function argument. Client must provide a pointer to uid_t.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadAtom Not a valid atom

See Also XTSOLgetPropAttributes(3XTSOL), XTSOLsetPropUID(3XTSOL)

“Setting Window Polyinstantiation Information” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetResAttributes – get all label attributes associated with a window or a pixmap

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetResAttributes(display, object, type, winattrp);
```

```
Display *display;
XID object;
ResourceType type;
XTSOLResAttributes *winattrp;
```

Description The client requires the PRIV_WIN_DAC_READ and PRIV_WIN_MAC_READ privileges. XTSOLgetResAttributes() is used to get all label attributes that are associated with a window or a pixmap in a single call. The attributes include UID, sensitivity label, and workstation owner.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- object* Specifies the ID of a window system object. Possible window system objects are windows and pixmaps.
- type* Specifies what type of resource is being accessed. Possible values are IsWindow and IsPixmap.
- winattrp* Client must provide a pointer to XTSOLResAttributes.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadPixmap Not a valid pixmap
- BadValue Not a valid type

See Also [XTSOLgetClientAttributes\(3XTSOL\)](#), [XTSOLgetPropAttributes\(3XTSOL\)](#)

“Obtaining Window Attributes” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetResLabel – get the label associated with a window, a pixmap, or a colormap

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetResLabel(display, object, type, sl);
```

```
Display *display;
XID object;
ResourceType type;
m_label_t *sl;
```

Description The client requires the PRIV_WIN_DAC_READ and PRIV_WIN_MAC_READ privileges. XTSOLgetResLabel() is used to get the label that is associated with a window or a pixmap or a colormap.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- object* Specifies the ID of a window system object whose label you want to get. Possible window system objects are windows, pixmaps, and colormaps.
- type* Specifies what type of resource is being accessed. Possible values are IsWindow, IsPixmap or IsColormap.
- sl* Returns a sensitivity label which is the current label of the specified object.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadPixmap Not a valid pixmap
- BadValue Not a valid type

See Also [XTSOLgetClientAttributes\(3XTSOL\)](#), [XTSOLsetResLabel\(3XTSOL\)](#)

“Obtaining a Window Label” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetResUID – get the UID associated with a window, a pixmap

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetResUID(display, object, type, uidp);
```

```
Display *display;
XID object;
ResourceType type;
uid_t *uidp;
```

Description The client requires the PRIV_WIN_DAC_READ and PRIV_WIN_MAC_READ privileges.

XTSOLgetResUID() gets the ownership of a window system object. This allows a client to get the ownership of an object that the client did not create.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- object* Specifies the ID of a window system object whose UID you want to get. Possible window system objects are windows or pixmaps.
- type* Specifies what type of resource is being accessed. Possible values are IsWindow and IsPixmap.
- uidp* Returns a UID which is the current UID of the specified object. Client must provide a pointer to uid_t.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadPixmap Not a valid pixmap
- BadValue Not a valid type

See Also XTSOLgetClientAttributes(3XTSOL), XTSOLgetResAttributes(3XTSOL), XTSOLgetResLabel(3XTSOL)

“Obtaining the Window User ID” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetSSHeight – get the height of screen stripe

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetSSHeight(display, screen_num, newheight);
```

```
Display *display;
int screen_num;
int *newheight;
```

Description XTSOLgetSSHeight () gets the height of trusted screen stripe at the bottom of the screen. Currently the screen stripe is only present on the default screen. Client must have the Trusted Path process attribute.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay ().
screen_num Specifies the screen number.
newheight Specifies the storage area where the height of the stripe in pixels is returned.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege
BadValue Not a valid *screen_num* or *newheight*

See Also [XTSOLsetSSHeight\(3XTSOL\)](#)

“Accessing and Setting the Screen Stripe Height” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLgetWorkstationOwner – get the ownership of the workstation

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLgetWorkstationOwner(display, uidp);
```

```
Display *display;
uid_t *uidp;
```

Description XTSOLgetWorkstationOwner() is used to get the ownership of the workstation.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().

uidp Returns a UID which is the current UID of the specified Display workstation server. Client must provide a pointer to uid_t.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None.

Errors BadAccess Lack of privilege

See Also [XTSOLsetWorkstationOwner\(3XTSOL\)](#)

“Obtaining the X Window Server Workstation Owner ID” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLIsWindowTrusted – test if a window is created by a trusted client

Synopsis #include <X11/extensions/Xtsol.h>

```
Bool XTSOLIsWindowTrusted(display, window);
```

```
Display *display;
```

```
Window window;
```

Description XTSOLIsWindowTrusted() tests if a window is created by a trusted client. The window created by a trusted client has a special bit turned on. The client does not require any privilege to perform this operation.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().

window Specifies the ID of the window to be tested.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values True If the window is created by a trusted client.

Errors BadWindow Not a valid window.

Name XTSOLMakeTPWindow – make this window a Trusted Path window

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLMakeTPWindow(display, w);
```

```
Display *display;
```

```
Window w;
```

Description XTSOLMakeTPWindow() is used to make a window a trusted path window. Trusted Path windows always remain on top of other windows. The client must have the Trusted Path process attribute set.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
w Specifies the ID of a window.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege
BadWindow Not a valid window
BadValue Not a valid type

Name XTSOLsetPolyInstInfo – set polyinstantiation information

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetPolyInstInfo(display, sl, uidp, enabled);
```

```
Display *display;
m_label_t sl;
uid_t *uidp;
int enabled;
```

Description XTSOLsetPolyInstInfo() sets the polyinstantiated information to get property resources. By default, when a client requests property data for a polyinstantiated property, the data returned corresponds to the SL and UID of the requesting client. To get the property data associated with a property with specific *sl* and *uid* a client can use this call to set the SL and UID with *enabled* flag to TRUE. The client should also restore the *enabled* flag to FALSE after retrieving the property value. Client must have the PRIV_WIN_MAC_WRITE and PRIV_WIN_DAC_WRITE privileges.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().

sl Specifies the sensitivity label.

uidp Specifies the pointer to UID.

enabled Specifies whether client can set the property information retrieved.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege

BadValue Not a valid *display* or *sl*.

See Also “Setting Window Polyinstantiation Information” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLsetPropLabel – set the label associated with a property hanging on a window

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetPropLabel(*display, window, property, *sl);
```

```
Display *display;
Window window;
Atom property;
m_label_t *sl;
```

Description XTSOLsetPropLabel() is used to change the sensitivity label that is associated with a property hanging on a window. The client must have the PRIV_WIN_DAC_WRITE, PRIV_WIN_MAC_WRITE, and PRIV_WIN_UPGRADE_SL privileges.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- window* Specifies the ID of the window whose property's label you want to change.
- property* Specifies the property atom.
- sl* Specifies a pointer to a sensitivity label.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadAtom Not a valid atom
- BadValue Not a valid *sl*

See Also [XTSOLgetPropAttributes\(3XTSOL\)](#), [XTSOLgetPropLabel\(3XTSOL\)](#)

Name XTSOLsetPropUID – set the UID associated with a property hanging on a window

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetPropUID(display, window, property, uidp);
```

```
Display *display;
Window window;
Atom property;
uid_t *uidp;
```

Description XTSOLsetPropUID() changes the ownership of a window's property. This allows another client to modify a property of a window that it did not create. The client must have the PRIV_WIN_DAC_WRITE and PRIV_WIN_MAC_WRITE privileges.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- window* Specifies the ID of the window whose property's UID you want to change.
- property* Specifies the property atom.
- uidp* Specifies a pointer to a uid_t that contains a UID.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadAtom Not a valid atom

See Also [XTSOLgetPropAttributes\(3XTSOL\)](#), [XTSOLgetPropUID\(3XTSOL\)](#)

Name XTSOLsetResLabel – set the label associated with a window or a pixmap

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetResLabel(display, object, type, sl);
```

```
Display *display;
XID object;
ResourceType type;
m_label_t *sl;
```

Description The client must have the PRIV_WIN_DAC_WRITE, PRIV_WIN_MAC_WRITE, PRIV_WIN_UPGRADE_SL, and PRIV_WIN_DOWNGRADE_SL privileges.

XTSOLsetResLabel() is used to change the label that is associated with a window or a pixmap.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- object* Specifies the ID of a window system object whose label you want to change. Possible window system objects are windows and pixmaps.
- type* Specifies what type of resource is being accessed. Possible values are IsWindow and IsPixmap.
- sl* Specifies a pointer to a sensitivity label.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadPixmap Not a valid pixmap
- BadValue Not a valid *type* or *sl*

See Also [XTSOLgetResAttributes\(3XTSOL\)](#), [XTSOLgetResLabel\(3XTSOL\)](#)

“Setting a Window Label” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLsetResUID – set the UID associated with a window, a pixmap, or a colormap

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetResUID(display, object, type, uidp);
```

```
Display *display;
XID object;
ResourceType type;
uid_t *uidp;
```

Description The client must have the PRIV_WIN_DAC_WRITE and PRIV_WIN_MAC_WRITE privileges. XTSOLsetResUID() changes the ownership of a window system object. This allows a client to create an object and then change its ownership. The new owner can then make modifications on this object as this object being created by itself.

Parameters

- display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
- object* Specifies the ID of a window system object whose UID you want to change. Possible window system objects are windows and pixmaps.
- type* Specifies what type of resource is being accessed. Possible values are: IsWindow and IsPixmap.
- uidp* Specifies a pointer to a uid_t structure that contains a UID.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors

- BadAccess Lack of privilege
- BadWindow Not a valid window
- BadPixmap Not a valid pixmap
- BadValue Not a valid type

See Also [XTSOLgetResUID\(3XTSOL\)](#)

Name XTSOLsetSessionHI – set the session high sensitivity label to the window server

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetSessionHI(display, sl);
```

```
Display *display;
m_label_t *sl;
```

Description XTSOLsetSessionHI() After the session high label has been set by a Trusted Extensions window system TCB component, login tool, Xsun will reject connection request from clients running at higher sensitivity labels than the session high label. The client must have the PRIV_WIN_CONFIG privilege.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().
sl Specifies a pointer to a sensitivity label to be used as the session high label.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege

See Also [XTSOLsetSessionLO\(3XTSOL\)](#)

“Setting the X Window Server Clearance and Minimum Label” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLsetSessionLO – set the session low sensitivity label to the window server

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetSessionLO(display, sl);
```

```
Display *display;
m_label_t *sl;
```

Description XTSOLsetSessionLO() sets the session low sensitivity label. After the session low label has been set by a Trusted Extensions window system TCB component, `login1tool`, Xsun will reject a connection request from a client running at a lower sensitivity label than the session low label. The client must have the PRIV_WIN_CONFIG privilege.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().

sl Specifies a pointer to a sensitivity label to be used as the session low label.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege

See Also [XTSOLsetSessionHI\(3XTSOL\)](#)

“Setting the X Window Server Clearance and Minimum Label” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLsetSSHeight – set the height of screen stripe

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetSSHeight(display, screen_num, newheight);
```

```
Display *display;
int screen_num;
int newheight;
```

Description XTSOLsetSSHeight () sets the height of the trusted screen stripe at the bottom of the screen. Currently the screen stripe is present only on the default screen. The client must have the Trusted Path process attribute.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay.
screen_num Specifies the screen number.
newheight Specifies the height of the stripe in pixels.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege
BadValue Not a valid *screen_num* or *newheight*.

See Also [XTSOLgetSSHeight\(3XTSOL\)](#)

“Accessing and Setting the Screen Stripe Height” in *Solaris Trusted Extensions Developer’s Guide*

Name XTSOLsetWorkstationOwner – set the ownership of the workstation

Synopsis #include <X11/extensions/Xtsol.h>

```
Status XTSOLsetWorkstationOwner(display, uidp);
```

```
Display *display;
uid_t *uidp;
XTSOLClientAttributes *clientattrp;
```

Description XTSOLsetWorkstationOwner() is used by the Solaris Trusted Extensions `logintool` to assign a user ID to be identified as the owner of the workstation server. The client running under this user ID can set the server's device objects, such as keyboard mapping, mouse mapping, and modifier mapping. The client must have the Trusted Path process attribute.

Parameters *display* Specifies a pointer to the Display structure; returned from XOpenDisplay().

uidp Specifies a pointer to a uid_t structure that contains a UID.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxwts
MT-Level	MT-Unsafe

Return Values None

Errors BadAccess Lack of privilege

See Also [XTSOLgetWorkstationOwner\(3XTSOL\)](#)

“Accessing and Setting a Workstation Owner ID” in *Solaris Trusted Extensions Developer’s Guide*

REFERENCE

File Formats

Name label_encodings – label encodings file

Synopsis /etc/security/tsol/label_encodings

Interface Level This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other Mandatory policies that might be developed for future releases of Solaris Trusted Extensions software.

Parts of the `label_encodings` file are considered standard and are controlled by Defense Intelligence Agency document DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993. Of that standard, the parts that refer to the `INFORMATION LABELS:` and `NAME INFORMATION LABELS:` sections are Obsolete. However, the `INFORMATION LABELS:` section must be present and syntactically correct. It is ignored. The `NAME INFORMATION LABELS:` section is optional. If present, it is ignored but must be syntactically correct.

The following values in the optional `LOCAL DEFINITIONS:` section are obsolete. These values might only affect the obsolete `bltos(3TSOL)` functions, and might be ignored by the `label_to_str(3TSOL)` replacement function:

```
ADMIN LOW NAME=  
ADMIN HIGH NAME=  
DEFAULT LABEL VIEW IS EXTERNAL  
DEFAULT LABEL VIEW IS INTERNAL  
DEFAULT FLAGS=  
FORCED FLAGS=  
CLASSIFICATION NAME=  
COMPARTMENTS NAME=
```

Description The `label_encodings` file is a standard encodings file of security labels that are used to control the conversion of human-readable labels into an internal format, the conversion from the internal format to a human-readable canonical form, and the construction of banner pages for printed output. On a Solaris Trusted Extensions system, the `label_encodings` file is protected at the label `admin_high`. The file should be edited and checked by the security administrator using the Check Label Encodings action in the `System_Admin` folder in the Application Manager.

Extended Description In addition to the required sections of the label encodings file that are described in *Compartmented Mode Workstation Labeling: Encodings Format*, a Solaris Trusted Extensions system accepts optional local extensions. These extensions provide various translation options and an association between character-coded color names and sensitivity labels.

The optional local extensions section starts with the `LOCAL DEFINITIONS:` keyword and is followed by zero or more of the following unordered statements:

DEFAULT USER SENSITIVITY LABEL= *sensitivity label*

This option specifies the sensitivity label to use as the user's minimum sensitivity label if none is defined for the user in the administrative databases. The default value is the MINIMUM SENSITIVITY LABEL= value from the ACCREDITATION RANGE: section of the label encodings file.

DEFAULT USER CLEARANCE= *clearance*

This option specifies the clearance to use as the user's clearance if none is defined for the user in the administrative databases. The default value is the MINIMUM CLEARANCE= value from the ACCREDITATION RANGE: section of the label encodings file.

The final part of the LOCAL DEFINITIONS: section defines the character-coded color names to be associated with various words, sensitivity labels, or classifications. This section supports the [str_to_label\(3TSOL\)](#) function. It consists of the COLOR NAMES: keyword and is followed by zero or more color-to-label assignments. Each statement has one of the following two syntaxes:

word= *word value*; color= *color value*;

label= *label value*; color= *color value*;

where *color value* is a character-coded color name to be associated with the word *word value*, or with the sensitivity label *label value*, or with the classification *label value*.

The character-coded color name *color value* for a label is determined by the order of entries in the COLOR NAMES: section that make up the label. If a label contains a word *word value* that is specified in this section, the *color value* of the label is the one associated with the first *word value* specified. If no specified word *word value* is contained in the label, the *color value* is the one associated with an exact match of a *label value*. If there is no exact match, the *color value* is the one associated with the first specified *label value* whose classification matches the classification of the label.

Examples EXAMPLE 1 A Sample LOCAL DEFINITIONS: Section

LOCAL DEFINITIONS:

```
DEFAULT USER SENSITIVITY LABEL= C A;
DEFAULT USER CLEARANCE LABEL= S ABLE;
```

COLOR NAMES:

```
label= Admin_Low;    color= Pale Blue;
label= unclassified; color= light grey;
word= Project A;    color= bright blue;
label= c;           color= sea foam green;
label= secret;      color= #ff0000;      * Hexadecimal RGB value
word= Hotel;        color= Lavender;
word= KelO;         color= red;
```

EXAMPLE 1 A Sample LOCAL DEFINITIONS: Section (Continued)

```
label= TS;           color= khaki;
label= TS Elephant; color= yellow;
label= Admin_High;  color= shocking pink;
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsr
Stability Level	Mixed. See <code>INTERFACE LEVEL</code> , above.

Files `/etc/security/tsol/label_encodings`

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system. It is protected at the label `admin_high`.

Diagnostics The following diagnostics are in addition to those found in Appendix A of *Compartmented Mode Workstation Labeling: Encodings Format*:

Can't allocate NNN bytes for color names table.

The system cannot dynamically allocate the memory it needs to process the `COLOR NAMES` section.

Can't allocate NNN bytes for color table entry.

The system cannot dynamically allocate the memory it needs to process a Color Table entry.

Can't allocate NNN bytes for color word entry.

The system cannot dynamically allocate the memory it needs to process a Color Word entry.

Can't allocate NNN bytes for `DEFAULT USER CLEARANCE`.

The system cannot dynamically allocate the memory it needs to process the `DEFAULT USER CLEARANCE`.

Can't allocate NNN bytes for `DEFAULT USER SENSITIVITY LABEL`.

The system cannot dynamically allocate the memory it needs to process the `DEFAULT USER SENSITIVITY LABEL`.

`DEFAULT USER CLEARANCE= XXX` is not in canonical form. Is `YYY` what is intended?

This error occurs if the clearance specified, while understood, is not in canonical form. This additional canonicalization check ensures that no errors are made in specifying the clearance.

DEFAULT USER SENSITIVITY LABEL= XXX is not in canonical form. Is YYY what is intended?

This error occurs if a sensitivity label specified, while understood, is not in canonical form. This additional canonicalization check ensures that no errors are made in specifying the sensitivity label.

Duplicate DEFAULT USER CLEARANCE= ignored.

More than one DEFAULT USER CLEARANCE= option was encountered. All but the first are ignored.

Duplicate DEFAULT USER SENSITIVITY LABEL= ignored.

More than one DEFAULT USER SENSITIVITY LABEL= option was encountered. All but the first are ignored.

End of File not found where expected. Found instead: XXX.

The noted extraneous text was found when the end of label encodings file was expected.

End of File or LOCAL DEFINITIONS: not found. Found instead: XXX.

The noted extraneous text was found when the LOCAL DEFINITIONS: section or end of label encodings file was expected.

Found color XXX without associated label.

The color XXX was found, however it had no label or word associated with it.

Invalid color label XXX.

The label XXX cannot be parsed.

Invalid DEFAULT USER CLEARANCE XXX.

The DEFAULT USER CLEARANCE XXX cannot be parsed.

Invalid DEFAULT USER SENSITIVITY LABEL XXX.

The DEFAULT USER SENSITIVITY LABEL XXX cannot be parsed.

Label preceding XXX did not have a color specification.

A label or word was found without a matching color name.

Word XXX not found as a valid Sensitivity Label word.

The word XXX was not found as a valid word for a sensitivity label.

See Also [chk_encodings\(1M\)](#), [label_to_str\(3TSOL\)](#), [str_to_label\(3TSOL\)](#), [attributes\(5\)](#), [labels\(5\)](#)

Solaris Trusted Extensions Label Administration

Defense Intelligence Agency document DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993.

Warnings Creation of and modification to the label encodings file should only be undertaken with a thorough understanding not only of the concepts in *Compartmented Mode Workstation Labeling: Encodings Format*, but also of the details of the local labeling requirements.

The following warnings are paraphrased from *Compartmented Mode Workstation Labeling: Encodings Format*.

Take extreme care when modifying a label encodings file that is already loaded and running on a Solaris Trusted Extensions system. Once the system runs with the label encodings file, many objects are labeled with sensitivity labels that are well formed with respect to the loaded label encodings file. If the label encodings file is subsequently changed, it is possible that the existing labels will no longer be well-formed. Changing the bit patterns associated with words causes existing objects whose labels contain the words to have possibly invalid labels. Raising the minimum classification or lowering the maximum classification that is associated with words will likely cause existing objects whose labels contain the words to no longer be well-formed.

Changes to a current encodings file that has already been used should be limited only to adding new classifications or words, changing the names of existing words, or modifying the local extensions. As described in *Compartmented Mode Workstation Labeling: Encodings Format*, it is important to reserve extra inverse bits when the label encodings file is first created to allow for later expansion of the label encodings file to incorporate new inverse words. If an inverse word is added that does not use reserved inverse bits, all existing objects on the system will erroneously have labels that include the new inverse word.

Notes This file is only meaningful for the DIA MAC policy. Parts of it are obsolete and retained for ease of porting. The obsolete parts might be removed in a future Solaris Trusted Extensions release.

Defining the label encodings file is a three-step process. First, the set of human-readable labels to be represented must be identified and understood. The definition of this set includes the list of classifications and other words that are used in the human-readable labels, relations between and among the words, classification restrictions that are associated with use of each word, and intended use of the words in mandatory access control and labeling system output. Next, this definition is associated with an internal format of integers, bit patterns, and logical relationship statements. Finally, a label encodings file is created. The *Compartmented Mode Workstation Labeling: Encodings Format* document describes the second and third steps, and assumes that the first has already been performed.

-
- Name** sel_config – selection rules for copy, cut, paste, drag and drop operations
- Synopsis** /usr/dt/config/sel_config
- Description** The sel_config file specifies how a system that is configured with Trusted Extensions behaves when a user transfers data between windows that have different labels. Transfer operations include cut-and-paste, copy-and-paste, and drag-and-drop. There are two types of entries in this file: automatic confirmation and automatic reply.
- Automatic Confirmation** This type of entry specifies whether a confirmation window, the selection confirmer, displays. Each entry has the form:
- relationship: confirmation*
- relationship* identifies the result of comparing the selected data's source and destination windows' labels. There are three allowed values:
- | | |
|-------------|---|
| upgradesl | The source window's label is less than the destination window's label. |
| downgradesl | The source window's label is higher than the destination window's label. |
| disjointsl | The source and destination windows' labels are disjoint. Neither label dominates the other. |
- confirmation* specifies whether to perform automatic confirmation. Allowed values are:
- | | |
|---|--|
| n | Use manual confirmation, that is, display the selection confirmer window. This is the default. |
| y | Use automatic confirmation, that is, do not display the selection confirmer window. |
- Automatic Reply** A single user operation can involve several flows of information between the source and destination windows. The automatic reply set of entries provides a means to reduce the number of confirmations that are required of the user.
- There must be one entry of this form:
- autoreply: value*
- If *value* is y (for yes), then the remaining entries of the set are used as attributes for the selection data (rather than the actual contents) to complete the operation without confirmation. If *value* is n (for no), then the remaining entries are ignored.
- Defaults can be specified for any *type* field that appears in the Confirmer window. Below are some sample entries for defaults.
- ```
replytype: TARGETS
replytype: Pixel Sets
replytype: LENGTH
replytype: Type Of Monitor
```

The TARGETS entry, when used, returns the list of target atoms that are supported by the source window. The Pixel Sets and Type Of Monitor entries are used for animation during a drag-and-drop operation. The LENGTH entry specifies the number of bytes in the selection.

**Attributes** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | SUNWtsu         |

**See Also** attributes(5)

“Rules When Changing the Level of Security for Data” in *Solaris Trusted Extensions Administrator’s Procedures*

**Name** tnrhdb – trusted network remote-host database

**Synopsis** /etc/security/tso1/tnrhdb

**Description** The tnrhdb database specifies which remote-host template to use for each host, including the local host, in the distributed system. tnrhdb works together with the [tnrhtp\(4\)](#) database to enable the administrator to establish the security and network accreditation attributes for each host. If a host's IP address cannot be matched to some entry in the tnrhdb database, communication with the host is not permitted.

The trusted network software uses a network “longest prefix of matching bits” mechanism when looking for a tnrhdb entry for a host. The software looks first for an entry that is specific to the host. If the software does not find a matching entry, the software falls back to searching for an entry with the longest prefix of a matching bit pattern, and so on.

**Note** – The actual numeric value of the subnet address or other subnetting information on the system (for example, from the [netmasks\(4\)](#) file) are not considered by this mechanism.

Using the “longest prefix of matching bits” mechanism, an IPv4 wildcard entry (IPv4 address 0.0.0.0) has a prefix length of 0 and hence can match any IPv4 address.

Each entry in tnrhdb consists of a line of the form *IP-address:template*.

*IP-address* This field is the IP address of the host or network that has the security properties that are specified by the *template* that is defined in the [tnrhtp\(4\)](#) database.

An entry can be a host address, for example, 10.100.100.201 or fe80:::9::20ff::fea0::21f7. Or an entry can be an IPv4 or IPv6 subnet address.

An IPv4 subnet entry can take the form of a subnet address with an explicit prefix length (10.100.128.0/17) or the form of a subnet address with trailing zero octets that imply a prefix length (10.100.0.0).

An IPv6 subnet entry must take the form of a subnet address with a prefix length (fe80:::/10). See NOTES for the use of the backslash in tnrhdb entries.

When IPv4 subnet entries are specified by using the implied prefix length format, the actual prefix length will take the value 0, 8, 16, or 24 when there are 4, 3, 2, or 1 trailing zero octets, respectively. An entry with a non-zero value in the final octet is interpreted as a host address and implies a prefix length of 32. See EXAMPLES for sample IPv4 entries.

*template* This value must be a valid template name in the [tnrhtp](#) database. For information on the security attributes, see [tnrhtp\(4\)](#).

More than one IP address can use the same template. If this database is modified while the network is up, the changes do not take effect until after `tnctl(1M)` is used to update the remote-host entries. Administrators are allowed to add new entries and modify existing entries while the network is up. The *template* field cannot contain any white spaces.

After each modification to the `tnrhdb` database, the administrator should run `tnchkdb(1M)` to check the syntax. If this database is modified while the network is up, the changes do not take effect until `tnctl(1M)` updates the kernel.

#### Examples EXAMPLE 1 Sample IPv4 Entries

| IPv4 Entry   | Host Address<br>or Wildcard? | Implied Prefix<br>Length |
|--------------|------------------------------|--------------------------|
| 0.0.0.0      | Wildcard                     | 0                        |
| 10.0.0.0     | Wildcard                     | 8                        |
| 10.100.0.0   | Wildcard                     | 16                       |
| 10.0.100.0   | Wildcard                     | 24                       |
| 10.0.100.100 | Host Address                 | 32                       |

#### EXAMPLE 2 Sample tnrhdb File

The templates in the following example are first defined in the `tnrhtp`, then used in the `tnrhdb` file. The example shows a host that uses the template `cipso`, a host that uses the template `public`, and a host that uses the template `needtoknow`. There are two subnets. One subnet uses the template `internal`, and the other subnet uses the template `secret`. Every other host uses the template `default-template` that is specified in the wildcard entries for IPv4 hosts and IPv6 hosts.

```
#
Assume that templates default-template, cipso, public,
internal, needtoknow, and secret are defined in the
tnrhtp database.
#
the first two entries are addresses of the IPv4 and
IPv6 loopback interfaces
127.0.0.1:cipso
\:\:1:cipso
10.0.0.1:cipso
192.168.120.6:public
192.168.120.0:internal
192.168.120.7:needtoknow
192.168.121.0:secret
0.0.0.0:default-template
0\:\:0/0:default-template
fe80\:\:a00\:\:20ff\:\:fea0\:\:21f7:cipso
```

**Attributes** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | SUNWtsg         |
| Stability      | Project Private |

**Files** `/etc/security/tsol/tnrhdb` Trusted network remote-host database

**See Also** `smtnrhdb(1M)`, `hosts(4)`, `ipnodes(4)`, `netmasks(4)`, `tnchkdb(1M)`, `tnctl(1M)`, `tnd(1M)`, `tninfo(1M)`, `tnrhtp(4)`, `tnzonecfg(4)`, `attributes(5)`

Chapter 12, “Trusted Networking (Overview),” in *Solaris Trusted Extensions Administrator’s Procedures*

**Warnings** Changing a template while the network is up can change the security view of an undetermined number of hosts.

**Notes** The colon (:) character is a database separation character. If the colon is used as part of a data field, it must be escaped with a backslash (\), as in `fe80\:\:a00\:20ff\:fea0\:21f7`.

The administrator might want to create one `tnrhdb` entry for each host that runs Trusted Extensions software, and make one subnet entry that applies to all unlabeled hosts that have the same security attributes. Then, the administrator can make a separate entry for each host that must be assigned a different set of security attributes.

**Name** tnrhtp – trusted network remote-host templates

**Synopsis** /etc/security/tsol/tnrhtp

**Description** The tnrhtp database of templates is specified by the administrator for convenience when assigning accreditation and security attributes for each host in the distributed system, including the local host and network.

tnrhtp works together with [tnrhdb\(4\)](#). IP addresses in tnrhdb can be assigned only to templates that are defined in the tnrhtp database. After each modification to the tnrhtp database, the administrator should run [tnchkdb\(1M\)](#) to check the syntax.

Each entry in the template database is entered as one long line. The fields of the entry are separated by semicolons (;):

*template\_name:attr*

A pound sign (#) as the first character of a line indicates a comment line, which is ignored.

*template\_name*

Is a character string that names the template that is being defined. The string is case-sensitive. Only the first 31 characters of string are read and interpreted. You can use any printable character in *template\_name* except for field delimiters, newline, or the comment character.

*attr*

Is a list of semicolon (;) separated *key=value* pairs that describe the attributes of the template. All keys are mandatory unless otherwise indicated, even if no value other than none is set. The following keys are currently interpreted by the system.

|                        |                                                                                                                                                                                                                |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>host_type</code> | Takes one of two defined values, <code>unlabeled</code> and <code>cipso</code> . The <code>cipso</code> host type is for hosts that use CIPSO (Common IP Security Options - Tag Type 1 only) to label packets. |
| <code>def_label</code> | Defines the default attributes to be applied to incoming data from remote hosts that do not support these attributes. This key is valid for the <code>unlabeled</code> host type only.                         |
| <code>doi</code>       | Is the domain of interpretation. In the case of the <code>unlabeled</code> host type, this is the domain of interpretation for the <code>def_label</code> .                                                    |

The domain of interpretation defines the set of rules for translating between the external or internal representation of the security attributes and their network representation. When systems that are configured with Trusted Extensions software have the same doi, they share that set of rules. In the case of the `unlabeled` host type, these systems also share the same interpretation for the default attributes that are assigned to the unlabeled templates that have that same doi.

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>min_sl,max_sl</code> | <p>Specifies the label accreditation range for the remote hosts that use this template. All labels are specified in a shortened hexadecimal format, except for the administrative labels <code>ADMIN_LOW</code> and <code>ADMIN_HIGH</code>.</p> <p>For gateway systems, <code>min_sl</code> and <code>max_sl</code> define the default range for forwarding labeled packets. The label range for routes is typically set by using a <code>route(1M)</code> subcommand with the <code>-secattr</code> option. When the label range for routes is not specified, the <code>min_sl</code> to <code>max_sl</code> range in the <code>tnrhttp</code> database is used.</p> |
| <code>sl_set</code>        | <p>Specifies the security label set which is allowed for the remote hosts that use this template. For gateway systems, the labels in <code>sl_set</code> are used for forwarding labeled packets. <code>sl_set</code> is optional. The maximum number of labels in a set is 4.</p>                                                                                                                                                                                                                                                                                                                                                                                     |

If the `tnrhttp` database is modified while the network is up, the changes do not take effect immediately unless `tnctl(1M)` is used to update the template entries. Otherwise, the changes take effect when next polled by the trusted network daemon, `tnd(1M)`. Administrators are allowed to add new templates and modify attributes of existing templates while the network is up.

### Examples EXAMPLE 1 Unlabeled Host Entries

For the sake of clarity on this man page, examples are shown using a continuation character (`\`). In the database file, however, the backslash is not permitted because each entry is made on a single line.

```
Sample ADMIN_LOW template entry for machines or networks.
Note that the doi field is required.
#
admin_low:host_type=unlabeled;\
def_label=ADMIN_LOW;\
min_sl=ADMIN_LOW;\
max_sl=ADMIN_HIGH;\
doi=1;
```

Unless the label at which you want to communicate with an unlabeled host is `ADMIN_LOW`, you should not use the above template. Rather, you should use a template that matches an entry in your label encodings file. The following example matches an entry in the sample `label_encodings` file.

```
Sample PUBLIC template entry
based on the sample label_encodings file.
#
public:host_type=unlabeled;\
def_label=0x0002-08-08;\
min_sl=ADMIN_LOW;\
```

**EXAMPLE 1** Unlabeled Host Entries (Continued)

```
max_sl=ADMIN_HIGH;\
doi=1;
```

**EXAMPLE 2** CIPSO Host Entry

```
Labeled host template
#
hl_allzones:host_type=cipso;\
min_sl=ADMIN_LOW;\
max_sl=ADMIN_HIGH;\
doi=1;
```

**Attributes** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | SUNWtsg         |
| Stability      | Project Private |

**Files** `/etc/security/tsol/tnrhtp` Trusted network remote-host templates

**See Also** `route(1M)`, `smtnrhtp(1M)`, `tnchkdb(1M)`, `tnctl(1M)`, `tnd(1M)`, `tninfo(1M)`, `tnrhdb(4)`, `attributes(5)`

“Network Security Attributes in Trusted Extensions” in *Solaris Trusted Extensions Administrator’s Procedures*

**Warnings** Changing a template while the network is up can change the security view of an undetermined number of hosts.

Allowing unlabeled hosts onto a Solaris Trusted Extensions network is a security risk. To avoid compromising the rest of your network, such hosts must be *trusted* in the sense that the administrator is certain that these unlabeled hosts will not be used to compromise the distributed system. These hosts should also be physically protected to restrict access to authorized individuals. If you cannot guarantee that an unlabeled host is physically secure from tampering, it and similar hosts should be isolated on a separate branch of the network.

**Name** tnzonecfg – trusted network zone configuration database

**Synopsis** /etc/security/tsol/tnzonecfg

**Description** The tnzonecfg database is a list of Solaris Trusted Extensions zone configuration entries for the local host. The database is indexed by zone name. Each configuration entry specifies a zone's label, multilevel port (MLP), and other zone-related information for zone creation.

Each entry in the zone configuration database consists of five fields. Each entry is on one long line, with fields of the entry separated by colons (:).

```
zone-name:label:network-policy:zone-mlp-list:shared-mlp-list
global:ADMIN_LOW:1:6000-6003/tcp:6000-6003/tcp
```

A pound sign (#) as the first character of a line indicates a comment line, which is ignored.

- zone-name* Is the name for the zone. This name is used when the zone is configured. See zonecfg(1M), under the -z zonename option, for the constraints on zone names.
- label* Is the label for the zone. This field is used to label the zone when the zone is booted. The label can be in shortened hexadecimal format or in text format. The labels are defined in the label\_encodings file. Each zone must have a unique label.
- network-policy* Is the policy for handling all non-transport traffic. This field is used to decide for non-MLP traffic if an exact zone label is required or if a label range match is allowed. The value 0 indicates strict zone label matching for inbound packets. If this field is set to 1, the receiving host accepts packets within the host's accreditation range.
- ICMP packets that are received on the global zone IP address are accepted based on the label range of the global zone's tnrtcp entry if the global zone's *network-policy* field is set to 1. When this field is set to 0 for a zone, the zone will not respond to an ICMP echo request from a host with a different label.
- zone-mlp-list* Is the multilevel port configuration entry for a zone on the IP addresses that belong to that zone. *zone-mlp-list* is a list of semicolon-separated MLP configuration entries. Each MLP configuration entry is specified by *port/protocol* or *port-range/protocol*. For example, 6001-6003/tcp means that tcp ports 6001, 6002, and 6003 are all MLPs.
- An MLP is used to provide multilevel service in the global zone as well as in non-global zones. As an example of how a non-global zone can use an MLP, consider setting up two labeled zones, `internal` and `public`. The `internal` zone can access company networks; the `public` zone can access public internet but not the company's internal networks. For safe

browsing, when a user in the `internal` zone wants to browse the Internet, the `internal` zone browser forwards the URL to the `public` zone, and the web content is then displayed in a `public` zone web browser. That way, if the download in `public` zone compromises the web browser, it cannot affect the company's internal network. To set this up, `tcp` port `8080` in the `public` zone is an MLP (`8080/tcp`), and the `tnrhttp` template for the `public` zone has a label range from `PUBLIC` to `INTERNAL`.

#### *shared-mlp-list*

Is the multilevel port configuration entry for shared IP addresses. *shared-mlp-list* is a list of semicolon-separated MLP configuration entries. Each MLP configuration entry is specified by *port/protocol*. Other zones do not have access to this *port/protocol* on shared interfaces. It is a configuration error to specify the same *port/protocol* in the *shared-mlp-list* field of more than one zone.

A shared IP address can reduce the total number of IP addresses that are needed on the system, especially when configuring a large number of zones. If network traffic is received on a shared interface, on a port that is specified in a zone's *shared-mlp-list*, the traffic cannot be received by other zones.

After each modification to the `tnzonecfg` database, the administrator should run `tnchkdb(1M)` to check the syntax. If this database is modified while the network is up, the changes do not take effect until `tnctl(1M)` updates the kernel.

### Examples **EXAMPLE 1** Sample Zone Configuration Entries

In the database file, each zone entry is made on a single line.

In this example, there are four non-global zones: `public`, `internal`, `needtoknow`, and `restricted`. Only the global zone and the `public` zone have MLPs.

In the global entry, the *zone-mlp-list* value of `111/tcp;111/udp;2049/tcp;6000-6003/tcp` specifies these ports as MLPs in the global zone only. The *shared-mlp-list* value of `6000-6003/tcp` specifies these ports as MLPs for the shared IP addresses, that is, for the labeled zones. With a *network-policy* of 1, only the global zone accepts incoming packets from a host whose label is different from its own.

In the `public` entry, the *network-policy* value of 0 restricts it to receiving `public` non-transport traffic. The *zone-mlp-list* value of `8080/tcp` makes the `public` zone's web browser port an MLP.

The `8080 tcp` port in the other zones is a single-level port, so is not listed. Similarly, each labeled zone has a single-level 111 port, 2049 port, and so on.

```
#
Sample global zone configuration file
```

**EXAMPLE 1** Sample Zone Configuration Entries (Continued)

```

#
Multilevel Port (MLP) specification:
#
MLP PURPOSE
--- -
111 Port Mapper
2049 NFSv4 server
6000-6003 Multilevel Desktop
#
global:ADMIN_LOW:1:111/tcp;111/udp;2049/tcp;6000-6003/tcp:6000-6003/tcp
public:PUBLIC:0:8080/tcp:
internal:0x0004-08-48:0::
needtoknow:0x0004-08-68:0::
restricted:0x0004-08-78:0::

```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | SUNWtsg         |
| Stability      | Project Private |

**Files** /etc/security/tsol/tnzonecfg Trusted network zone configuration database

**See Also** [smttnzonecfg\(1M\)](#), [tnchkdb\(1M\)](#), [tnctl\(1M\)](#), [tnd\(1M\)](#), [tninfo\(1M\)](#), [zonecfg\(1M\)](#), [label\\_encodings\(4\)](#), [tnrhdb\(4\)](#), [tnrhtp\(4\)](#), [attributes\(5\)](#)

“Solaris Management Console Tools” in *Solaris Trusted Extensions Administrator’s Procedures*

**Name** TrustedExtensionsPolicy – configuration file for Trusted Extensions X Server Extension

**Synopsis** /usr/X11/lib/X11/xserver/TrustedExtensionsPolicy

/usr/openwin/server/etc/TrustedExtensionsPolicy

**Description** TrustedExtensionsPolicy is the configuration file for Trusted Extensions X Server Extension (SUN\_TSOL). SUN\_TSOL provides security policy enforcement. This enforcement is based on Mandatory Access Control (MAC) and Discretionary Access Control (DAC).

Blank lines and comments in the TrustedExtensionsPolicy file are ignored. Comments start with a pound sign (#). The format of the file is as follows:

*keyword*{space|tab}*value*

where *keyword* can be one of the following:

atom            Label this atom ADMIN\_LOW, so that XGetAtomName(3X11) succeeds.

property        Instantiate this property once. The default is to polyinstantiate a property.

selection       Polyinstantiate this selection. The default is to instantiate the selection once.

extension       Disable this extension.

privilege       Implicitly allow this window privilege on all clients.

For possible keyword values, see the /usr/X11/lib/X11/xserver/TrustedExtensionsPolicy file for the Xorg X server. For Xsun, see the /usr/openwin/server/etc/TrustedExtensionsPolicy file.

**Attributes** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | SUNWxwts        |

**Examples** The following entry in the TrustedExtensionsPolicy file polyinstantiates the Dtpad program:

```
selection Dtpad
```

If the entry is missing, or commented out, the Dtpad program is instantiated once.

Similarly, the following entry instantiates the WM\_ICON\_SIZE property once:

```
property WM_ICON_SIZE
```

If the entry is missing, or commented out, the WM\_ICON\_SIZE property is polyinstantiated.

**Files** /usr/X11/lib/X11/xserver/TrustedExtensionsPolicy  
Configuration file for Trusted Extensions X Server Extension

**See Also** XGetAtomName(3X11), attributes(5)



REFERENCE

Standards, Environments, and Macros

**Name** labels – Solaris Trusted Extensions label attributes

**Description** Labels are attributes that are used in mandatory policy decisions. Labels are associated, either explicitly or implicitly, with all subjects (generally processes) and objects (generally things with data such as files) that are accessible to subjects. The default Trusted Extensions mandatory policy labels are defined by a site's security administrator in [label\\_encodings\(4\)](#).

**Mandatory Policy** Various mandatory policies might be delivered in the lifetime of Solaris Trusted Extensions.

The default mandatory policy of Trusted Extensions is a Mandatory Access Control (MAC) policy that is equivalent to that of the Bell-LaPadula Model of the Lattice, the Simple Security Property, and the \*-Property (Star Property), with restricted write up. The default mandatory policy is also equivalent to the Goguen and Meseguer model of Non-Inteference.

For this MAC policy, two labels are always defined: `admin_low` and `admin_high`. The site's security administrator defines all other labels in [label\\_encodings\(4\)](#). `admin_low` is associated with all normal user readable (viewable) Trusted Extensions objects. `admin_high` is associated with all other Trusted Extensions objects. Only administrative users have MAC read (view) access to `admin_high` objects and only administrative users have MAC write (modify) access to `admin_low` objects or `admin_high` objects.

**Human Readable Labels** Users interact with labels as strings. Graphical user interfaces and command line interfaces present the strings as defined in [label\\_encodings\(4\)](#). Human readable labels are classified at the label that they represent. Thus the string for a label A is only readable (viewable, translatable to or from human readable to `opaque_m_label_t`) by a subject whose label allows read (view) access to that label.

**Internal Text Labels** In order to store labels in publicly accessible (`admin_low`) name service databases, an unclassified internal text form is used. This textual form is not intended to be used in any interfaces other than those that are provided with the Trusted Extensions software release that created this textual form of the label.

**Labels and Applications** Applications interact with labels as opaque (`m_label_t`) structures. The semantics of these opaque structures are defined by a string to `m_label_t` translation. This translation is defined in [label\\_encodings\(4\)](#). Various Application Programming Interfaces (API) translate between strings and `m_label_t` structures. Various APIs test access of subject-related labels to object-related labels.

**Attributes** See [attributes\(5\)](#) for description of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | See NOTES below |

**See Also** [chk\\_encodings\(1M\)](#), [blcompare\(3TSOL\)](#), [label\\_to\\_str\(3TSOL\)](#), [m\\_label\\_alloc\(3TSOL\)](#), [m\\_label\\_dup\(3TSOL\)](#), [m\\_label\\_free\(3TSOL\)](#), [str\\_to\\_label\(3TSOL\)](#), [label\\_encodings\(4\)](#), [attributes\(5\)](#)

Bell, D. E., and LaPadula, L. J. *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR-2997 Rev. 2, MITRE Corp., Bedford Mass., March 1976. NTIS AD-A023 588/7.

Goguen, J. A., and Mesegeur, J.: *Security Policies and Security Models*, Proceedings 1982 Symposium on Security and Privacy, IEEE Computer Society Press, 1982, p 11-20.

Goguen, J. A., and Mesegeur, J.: *Unwinding and Interference Control*, Proceedings 1984 Symposium on Security and Privacy, IEEE Computer Society Press, 1984, p 75-86.

*Compartmented Mode Workstation Labeling: Encodings Format*

**Notes** The stability of the labels implementation is Stable for systems that implement the Defense Intelligence Agency (DIA) MAC policy of `label_encodings(4)`. Other policies might exist in a future release of Trusted Extensions that might obsolete or supplement `label_encodings`.

Internal text labels are not an interface and might change with any release of Trusted Extensions. They are only intended for input and generation on the same release of Trusted Extensions software.

As a potential porting aid for Trusted Solaris 8 applications, the opaque structure names `bslabel_t`, `blevel_t`, and `bclear_t` are defined to be equivalent to `m_label_t`. Like `m_label_t`, these types must be ported as opaque pointers. The same must be done with the various Trusted Solaris 8 label interfaces. These Trusted Solaris 8 structures and interfaces are Obsolete and might be removed from a future release of Trusted Extensions.

**Name** pam\_tsol\_account – PAM account management module for Trusted Extensions

**Synopsis** /usr/lib/security/pam\_tsol\_account.so.1

**Description** The Solaris Trusted Extensions service module for PAM, /usr/lib/security/pam\_tsol\_account.so.1, checks account limitations that are related to labels. The pam\_tsol\_account.so.1 module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

pam\_tsol\_account.so.1 contains a function to perform account management, pam\_sm\_acct\_mgmt(). The function checks for the allowed label range for the user. The allowable label range is set by the defaults in the [label\\_encodings\(4\)](#) file. These defaults can be overridden by entries in the user\_attr(4) database.

By default, this module requires that remote hosts connecting to the global zone must have a CIPSO host type. To disable this policy, add the allow\_unlabeled keyword as an option to the entry in pam.conf(4), as in:

```
other account required pam_tsol_account allow_unlabeled
```

**Options** The following options can be passed to the module:

allow\_unlabeled Allows remote connections from hosts with unlabeled template types. See [tnrhtp\(4\)](#).

debug Provides debugging information at the LOG\_DEBUG level. See [syslog\(3C\)](#).

**Return Values** The following values are returned:

PAM\_SUCCESS The account is valid for use at this time and label.

PAM\_PERM\_DENIED The current process label is outside the user's label range, or the label information for the process is unavailable, or the remote host type is not valid.

Other values Returns an error code that is consistent with typical PAM operations. For information on error-related return values, see the [pam\(3PAM\)](#) man page.

**Attributes** See [attributes\(5\)](#) for description of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE         |
|---------------------|-------------------------|
| Interface Stability | Evolving                |
| MT Level            | MT-Safe with exceptions |

**See Also** [keylogin\(1\)](#), [syslog\(3C\)](#), [libpam\(3LIB\)](#), [pam\(3PAM\)](#), [pam\\_sm\\_acct\\_mgmt\(3PAM\)](#), [pam\\_start\(3PAM\)](#), [label\\_encodings\(4\)](#), [pam.conf\(4\)](#), [ttrhpt\(4\)](#), [user\\_attr\(4\)](#), [attributes\(5\)](#)

Chapter 17, “Using PAM,” in *System Administration Guide: Security Services*

**Notes** The interfaces in [libpam\(3LIB\)](#) are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.



# Index

---

## A

- account management for Trusted Extensions, 138
- add entries to device allocation mechanism,
  - add\_allocatable, 22
- administer tnrhdb, smtnrhdb, 28
- administer tnrhttp, smtnrhttp, 32
- administer tnzonecfg, smtnzonecfg, 36
- allocate resources for a new destination label, 82
- allocate resources for a new label, 82

## B

- bounds of two labels, 57

## C

- change file label, setlabel, 19
- change sensitivity label of file, setflabel, 85
- change user interface for labels, tsol\_lbuild\_set, 73
- check file syntax of trusted network databases,
  - tnchkdb, 40
- check label encodings file syntax, chk\_encodings, 25
- compare labels for dominance, 56
- compare labels for equality, 56
- compare labels for strict dominance, 56
- configuration file for Trusted Extensions X Server Extension, TrustedExtensionsPolicy, 132
- configure Trusted Extensions network parameters,
  - tnctl, 42

- convert a human readable label to its internal text equivalent, atohexlabel, 24
- convert an internal text label to its human readable equivalent, hextoalabel, 26
- convert labels to strings, label\_to\_str, 80
- convert strings to labels, str\_to\_label, 90
- .copy\_files, 48
- copy home directory files, updatehome, 48
- create user interface for labels, tsol\_lbuild\_create, 73

## D

- databases
  - administer tnrhdb from Solaris Management Console, 28
  - administer tnrhttp from Solaris Management Console, 32
  - administer tnzonecfg from Solaris Management Console, 36
  - check file syntax of network databases, 40
  - configure trusted network databases, 42
  - get remote host type from trusted network databases, 92
  - label encodings, 116
  - tnrhdb, 123
  - tnrhttp, 126
  - tnzonecfg, 129
- destroy user interface for labels,
  - tsol\_lbuild\_destroy, 73
- determine if label is within label range, 56

**devices**

- add to device allocation mechanism, 22
- remove entries from device allocation mechanism, 27

**E****/etc/security/tsol files**

- label encodings, 116
- tnrhdb, 123
- tnrhtp, 126
- tnzonecfg, 129

**F****file label**

- fgetlabel, 52
- getlabel, 52
- setflabel, 85
- setlabel, 19

**files**

- .copy\_files, 48
  - description of labels model, 136
  - label encodings, 116
  - .link\_files, 48
  - pam\_tsol\_account, 138
  - tnrhdb, 123
  - tnrhtp, 126
  - tnzonecfg, 129
  - TrustedExtensionsPolicy, 132
- free resources that are associated with the previously allocated label, 82

**G**

- get all label attributes associated with a client, XTSOLgetClientAttributes, 94
- get all label attributes associated with a window or a pixmap, XTSOLgetResAttributes, 98
- get remote host type, tsol\_getrhtype, 92
- get the height of screen stripe, XTSOLgetSSHeight, 101

- get the label associated with a property hanging on a window, XTSOLgetPropLabel, 96
- get the label associated with a window, a pixmap, or a colormap, XTSOLgetResLabel, 99
- get the label attributes associated with a property hanging on a window, XTSOLgetPropAttributes, 95
- get the label range of a user, 69
- get the ownership of the workstation, XTSOLgetWorkstationOwner, 102
- get the UID associated with a property hanging on a window, XTSOLgetPropUID, 97
- get the UID associated with a window, a pixmap, XTSOLgetResUID, 100
- get trusted network information, tsol\_getrhtype, 92
- get user interface for labels, tsol\_lbuild\_get, 73
- greatest lower bound of two labels, blminimum, 57

**I**

- initialize network databases, tnd, 44

**L**

- label appearance, label builder, 73
- label builder
  - tsol\_lbuild\_create, 73
  - tsol\_lbuild\_destroy, 73
  - tsol\_lbuild\_get, 73
  - tsol\_lbuild\_set, 73
- label interface, label builder, 73
- label library
  - bldominates, 56
  - blequal, 56
  - blinrange, 56
  - blmaximum, 57
  - blminimum, 57
  - blstrictdom, 56
  - getdevicerange, 65
  - getpathbylabel, 66
  - getlabel, 68
  - getuserrange, 69
  - getzoneidbylabel, 70
  - getzonelabelbyid, 70

label library (*Continued*)

- getzonelabelbyname, 70
- getzonerootbyid, 71
- getzonerootbylabel, 71
- getzonerootbyname, 71
- label\_to\_str, 80
- m\_label\_alloc, 82
- m\_label\_dup, 82
- m\_label\_free, 82
- setflabel, 85
- str\_to\_label, 90
- tsol\_lbuild\_create, 73
- tsol\_lbuild\_destroy, 73
- tsol\_lbuild\_get, 73
- tsol\_lbuild\_set, 73

## label mappings

- getzoneidbylabel, 70
- getzonelabelbyid, 70
- getzonelabelbyname, 70
- getzonerootbyid, 71
- getzonerootbylabel, 71
- getzonerootbyname, 71
- label\_to\_str, 80
- str\_to\_label, 90

## label of file

- fgetlabel, 52
- getlabel, 16, 52
- setflabel, 85
- setlabel, 19

## label of process, plabel, 18

## label ranges

- blcompare, 56
- getdevicerange, 65
- getuserrange, 69

## labels

- appearance, 73
- bounds of two labels, 57
- build labels, 73
- check label encodings file syntax, 25
- compare labels, 56
- convert to human readable equivalent, 26
- convert to internal text equivalent, 24
- convert to strings, 80
- copy home directory files to different labels, 48

labels (*Continued*)

- description, 136
- encodings file, 116
- get file label, 16, 52
- get process label, 18
- interface, 73
- label builder, 73
- link home directory files to different labels, 48
- model for Solaris Trusted Extensions, 136
- parse from strings, 90
- set file label, 19, 85
- user interface, 73

## labels file, 136

## labels model, 136

## least upper bound of two labels, blmaximum, 57

## .link\_files, 48

## link home directory files, updatehome, 48

**M**

- make this window a Trusted Path window, XTSOLMakeTPWindow, 104

- manage entries in the tnrhdb, smtnrhdb, 28

- manage entries in the tnrtcp database, smtnrtcp, 32

- manage entries in the tnzonecfg database, smtnzonecfg, 36

## map between zone root pathnames and labels

- getzonerootbyid, 71
- getzonerootbylabel, 71
- getzonerootbyname, 71

## map between zones and labels

- getzoneidbylabel, 70
- getzonelabelbyid, 70
- getzonelabelbyname, 70

## move files to zone with corresponding label, 19

**N**

## network commands

- check syntax of databases, 40
- configure trusted network databases, 42
- initialize network databases, 44
- print network information, 46

network commands (*Continued*)

smtnrhdb, 28

smtnrhtp, 32

## network databases

administer tnrhdb from Solaris Management

Console, 28

administer tnrhtp from Solaris Management

Console, 32

check syntax, 40

configure trusted network, 42

get remote host type from, 92

initialize the kernel with, 44

print information about, 46

tnrhdb, 123

tnrhtp, 126

## network information

get remote host type from trusted network

databases, 92

tninfo, 46

**O**

## obsolete functions

bclearth, 63

bclearth\_r, 63

bclearthos, 60

bltcolor, 58

bltcolor\_r, 58

bslth, 63

bslth\_r, 63

bslthos, 60

h\_alloc, 63

h\_free, 63

htobclear, 72

htobsl, 72

sbclearthos, 83

sbslthos, 83

stobclear, 87

stobsl, 87

Xbclearthos, 78

Xbslthos, 78

**P**

parse strings to labels, str\_to\_label, 90

process label, getlabel, 68

**R**

remote administration, dtappsession, 14

return zone pathname, getpathbylabel, 66

**S**

selection rules for copy, cut, paste, drag and drop operations, sel\_config file, 121

sensitivity label of file

fgetlabel, 52

getlabel, 52

set polyinstantiation information,

XTSOLsetPolyInstInfo, 105

set sensitivity label of file, setlabel, 85

set the height of screen stripe, XTSOLsetSSHeight, 112

set the label associated with a property hanging on a window, XTSOLsetPropLabel, 106

set the label associated with a window or a pixmap, XTSOLsetResLabel, 108

set the ownership of the workstation, XTSOLsetWorkstationOwner, 113

set the session high sensitivity label, XTSOLsetSessionHI, 110

set the session low sensitivity label, XTSOLsetSessionLO, 111

set the UID associated with a property hanging on a window, XTSOLsetPropUID, 107

set the UID associated with a window, a pixmap, or a colormap, XTSOLsetResUID, 109

smf services

tnctl, 42

tnd, 44

Solaris Management Console

administer tnrhdb database, 28

administer tnrhtp database, 32

administer tnzonecfg database, 36

start a new Application Manager session, 14

SUN\_TSOL, 132

**T**

test if a window is created by a trusted client,  
    XTSOLIsWindowTrusted, 103  
tnrhdb, get remote host type from, 92  
Trusted Extensions X Server Extension, 132  
trusted network daemon, tnd, 44  
trusted network databases, *See* network databases  
trusted network remote-host database, tnrhdb, 123  
trusted network remote-host templates, tnrhtp, 126  
trusted network zone configuration database,  
    tnzonecfg, 129

**U**

user interface, 73  
/usr/openwin/server/etc/TrustedExtensionsPolicy, 132  
/usr/X11/lib/X11/xserver/TrustedExtensionsPolicy, 132

**Z**

zone configuration database, 129  
zone pathname, 66  
zones  
    getpathbylabel, 66  
    getzoneidbylabel, 70  
    getzonelabelbyid, 70  
    getzonelabelbyname, 70  
    getzonepath, 17  
    getzonerootbyid, 71  
    getzonerootbylabel, 71  
    getzonerootbyname, 71  
    tnzonecfg, 129

