Sun Enterprise™ 10000 Domain Configuration Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto,CA 94303-4900
U.S.A. 650-960-1300

Part No. 806-4121-10
February 2000, Revision 01

Send comments about this document to: docfeedback@sun.com

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

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. For Netscape Communicator™, the following notice applies: (c) Copyright 1995 Netscape Communications Corporation. All rights reserved.

Sun, Sun Microsystems, the Sun logo, AnswerBook2, docs.sun.com, Sun Enterprise, SunFIDDI, Sun StorEdge, OpenBoot, and Solaris are trademarks, registered trademarks, or service marks 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.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6 / 87) and FAR 52.227-19(6 / 87), or DFAR 252.227-7015(b)(6 / 95) and DFAR 227.7202-3(a).

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 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X / Open Company, Ltd. La notice suivante est applicable à Netscape Communicator™: (c) Copyright 1995 Netscape Communications Corporation. Tous droits réservés.

Sun, Sun Microsystems, le logo Sun, AnswerBook2, docs.sun.com, Sun Enterprise, SunFIDDI, Sun StorEdge, OpenBoot, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-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 Etats-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.

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

Please
Recycle

Sun Enterprise 10000 SSP Attributions:

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim inany distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

RESTRICTED RIGHTS: Use, duplication or disclosure by the government is subject to the restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software Clause as DFARS 252.227-7013 and FAR 52.227-19.

This is scotty, a simple tcl interpreter with some special commands to get information about TCP / IP networks. Copyright (c) 1993, 1994, 1995, J. Schoenwaelder, TU Braunschweig, Germany, Institute for Operating Systems and Computer Networks. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that this copyright notice appears in all copies. The University of Braunschweig makes no representations about the suitability of this software for any purpose. It is provided “as is" without express or implied warranty.

Contents

Tables

Preface

This guide describes the domain-side configuration of the Sun Enterprise™ 10000 server InterDomain Network (IDN) and Dynamic Reconfiguration (DR) features. For information about how to use these features, refer to the appropriate user guide listed in “Related Documentation” on page xiii.

Before You Read This Book

This guide is intended for the Sun Enterprise 10000 server system administrator who has a working knowledge of UNIX® systems, particularly those based on the Solaris™ operating environment. If you do not have such knowledge, first read all of the books in the Solaris System Administration collection in AnswerBook2™ format provided with your server and consider UNIX system administration training.

Also read and be familiar with the *TCP/IP and Data Communications Administration Guide* that is provided with your server in AnswerBook2 format.

How This Book Is Organized

This guide contains the following chapters:

Chapter 1 contains an introduction to this guide.

Chapter 2 contains descriptions on how to configure and reconfigure a Sun Enterprise 10000 domain before and after a DR operation.

Chapter 3 contains descriptions on how to configure an IDN for better performance and reliability.

Using UNIX Commands

This document does not contain information on basic UNIX commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or more of the following sources for this information:

- AnswerBook2 online documentation for the Solaris 2.x software environment, particularly those dealing with Solaris system administration
- Other software documentation that you received with your system

Typographic Conventions

Typeface or Symbol	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename.*

Shell Prompts

Shell	Prompt
C shell	*machine_name*%
C shell superuser	*machine_name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Related Documentation

Application	Title	Part Number
User	*Sun Enterprise 10000 SSP 3.3 User Guide*	806-2887
	Sun Enterprise 10000 Dynamic Reconfiguration User Guide	806-4122
	Sun Enterprise 10000 InterDomain Networks User Guide	806-4131
	TCP/IP and Data Communications Administration Guide	805-4003
Reference	*Sun Enterprise 10000 SSP 3.3 Reference Manual*	806-2888
	Sun Enterprise 10000 Dynamic Reconfiguration Reference Manual	806-4123
	Sun Enterprise 10000 Domain Error Messages	806-4120

Ordering Sun Documentation

Fatbrain.com, an Internet professional bookstore, stocks select product

documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatrain.com at:

`http://www1.fatrain.com/documentation/sun`

Accessing Sun Documentation Online

The `docs.sun.com`SM web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

`http://docs.sun.com`

Caution – The output of the AnswerBook2 collections depends on the font families you have chosen in your browser. Sun Microsystems suggests that you use a common san-serif font face for regular text and a common fixed-width face for screen text.

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

`docfeedback@sun.com`

Please include the part number (806-4121-10) of your document in the subject line of your email.

CHAPTER 1

Domain Configuration Introduction

This chapter contains an introduction to the *Sun Enterprise 10000 Domain Configuration Guide*. For information on how to use the Dynamic Reconfiguration (DR) feature, refer to the *Sun Enterprise 10000 Dynamic Reconfiguration User Guide* in the SSP 3.3 AnswerBook2™ collection. For information on how to set up and use InterDomain Networks, refer to the *Sun Enterprise 10000 InterDomain Networks User Guide* in the SSP 3.3 AnswerBook2 collection.

Introduction to DR Configuration

This guide contains all of the configuration and reconfiguration information for the DR feature. The following is a list of topics covered in this book. If you are viewing this book online, you can use the following list to link to a specific topic.

Introduction to IDN Configuration

This guide contains all of the configuration information for IDNs. The following is a list of topics covered in this book. If you are viewing this book online, you can use the following list to link to a specific topic.

Memory Error Handling

Memory errors within the SMR are reported by the processors that encounter them within the context of their respective domain. If a slave domain experiences a memory error in the SMR, that error is not reported to the master domain. Thus, it is possible that the master domain can export memory that is experiencing errors without being aware of the errors.

System Commands

This section contains descriptions of system commands that are affected by IDNs and how they are affected.

snoop(1M) Command

The snoop(1M) command supports only a limited number of network maximum transfer unit (MTU) sizes, all of which are significantly smaller than what IDN can support. The IDN driver appears to the system as a standard Ethernet device. For this reason, if you wish to use the snoop(1M) command to capture IDN data transfers, you must use the `-s` option with a specification of 1500 bytes, or less, as in the following example:

```
# snoop -d idn0 -s 1500
```

Due to the point-to-point nature of an IDN, only traffic directed to, or from, the local domain can be captured by the snoop(1M) command.

CHAPTER 2

DR Configuration Issues

This chapter describes how to configure a domain for all DR operations and capabilities.

Caution – Be careful when choosing the slot into which a board is inserted to prevent disk controller renumbering. For more information, see “Reconfiguration After a DR Operation” on page 8.

`dr-max-mem` Variable

With the Solaris 7 and Solaris 8 operating environments, `dr-max-mem` is no longer used. Instead, the DR feature, specifically DR Detach, must be enabled by using the `system`(4) variable `kernel_cage_enable`. A caged kernel confines the nonpageable memory to a minimal (most often one) number of systems boards. By default, the kernel cage is disabled, preventing DR Detach operations.

Note – DR Attach is enabled regardless of the setting of `kernel_cage_enable`.

▼ To Enable the Kernel Cage

1. **Edit the `/etc/system` file so that `kernel_cage_enable` equals 1.**

```
set kernel_cage_enable=1
```

2. **Reboot the domain.**

After the reboot completes successfully, you can verify that the kernel cage is enabled by reviewing the `/var/adm/messages` file for the following message.

```
NOTICE: DR Kernel Cage is ENABLED
```

Configuration for DR Detach

This section describes how to configure DR before you perform a detach operation.

I/O Devices

The DR Detach feature works with Alternate Pathing (AP) or Solstice™ DiskSuite™ mirroring when you detach a board that hosts I/O controllers that are attached to vital system resources. If, for example, the root (`/`) or `/usr` partition is on a disk attached to a controller on the board, the board cannot be detached unless there is a hardware alternate path to the disk, and AP has been configured to take advantage of it, or the disk is mirrored. The alternate path or the mirrors must be hosted by other boards in the domain. The same applies to network controllers. The board that hosts the Ethernet controller that connects the SSP to the Sun Enterprise 10000 platform cannot be detached unless an alternate path exists to an Ethernet controller on another board for this network connection.

To enable device suspension for the `soc` and `pln` drivers, you must edit the `/etc/system` file so that the `pln_enable_detach_suspend` and `soc_enable_detach_suspend` variables are set to 1, as in the following example:

```
set pln:pln_enable_detach_suspend=1
set soc:soc_enable_detach_suspend=1
```

The domain swap space should be configured as multiple partitions on disks attached to controllers hosted by different boards. With this kind of configuration, a particular swap partition is not a vital resource because swap partitions can be added and deleted dynamically (refer to the `swap`(1M) man page for more information).

Note – When memory (`swapfs`) or swap space on a disk is detached, there must be enough memory or swap space remaining in the domain to accommodate currently running programs.

A board that hosts non-vital system resources can be detached whether or not there are alternate paths to the resources. All of the devices on the board must be closed before the board can be detached; all of its file systems must be unmounted; and, its swap partitions must be deleted. You may have to kill processes that have open files or devices, or place a hard lock on the file systems (using `lockfs`(1M)) before you unmount the boards.

All I/O device drivers involved with I/O devices on the board(s) must support the `DDI_DETACH` option in the detach entry-point of the driver. This option releases all system resources associated with that device or adapter.

Driver Parameters

If you use the `ndd`(1M) command to set the configuration parameters for network drivers, the parameters may not persist after a DR Detach or DR Attach operation. Use the `/etc/system` file or the `driver.conf` file for a specific driver to set the parameters permanently.

Target Memory Constraints

When detaching a board with nonpageable memory, DR must locate an alternative (target) memory board to which to copy the nonpageable memory. In the Solaris 7 5/99 version, if no target board is found, the detach operation is refused, and DR displays the following warning message on the system console:

```
WARNING: sfdr: sfdr_pre_release_mem: no available target for mem-
unit (board.0)
```

Swap Space

The domain swap configuration consists of the swap devices and `swapfs` (memory). The domain must contain enough swap space so that it can flush pageable memory. For example, if you want to remove 1 Gbyte of memory from a 2-Gbyte domain, you will need 1 Gbyte of swap space, depending on the load. Insufficient swap space prevents DR from completing the detach of a board that contains memory. If this happens, the memory drain phase does not complete, so you must abort the detach operation.

Network Devices

DR automatically terminates usage of all network interfaces on the board that is being detached. When you complete the detach operation, the `dr_daemon`(1M) identifies all configured interfaces on the board being detached and issues the following `ifconfig`(1M) commands on each such interface.

```
ifconfig interface down
ifconfig interface unplumb
```

Additionally, if FDDI interfaces are detached, DR kills the FDDI network monitoring daemon before you perform the detach operation. DR then restarts it after the detach is complete. Note that the `/usr/sbin/nf_snmd` daemon for `nf` devices is neither started nor stopped when a board that contains a FDDI interface is attached.

DR does not execute these commands on a board that contains a network interface that fits any of the following conditions. In these cases, the detach operation fails and DR displays an error message.

- The interface is the primary network interface for the domain; that is, the interface whose IP address corresponds to the network interface name contained in the file `/etc/nodename`. Note that bringing down the primary network interface for the domain prevents network information name services from operating, which results in the inability to make network connections to remote hosts using applications such as `ftp`(1), `rsh`(1), `rcp`(1), `rlogin`(1). NFS client and server operations are also affected.
- The interface is on the same subnet as the SSP host for the system; that is, the subnet of the IP address that corresponds to the SSP host name found in `/etc/ssphostname`. Bringing down this interface interrupts communication between the host and SSP. Since DR operations are initiated on the SSP, control of the detach process would be lost. (Note that the `/etc/ssphostname` file contains the name of the SSP that controls the host; therefore, if you rename the SSP, the `/etc/ssphostname` must be manually updated.)
- The interface is the active alternate for an Alternate Pathing (AP) metadevice when the AP metadevice is plumbed. Interfaces used by AP should not be the active path when the board is being detached. AP 2.1 performs the switch automatically; however, you can manually switch the active path to an interface that is not on the board being detached. If no such path exists, manually execute the `ifconfig down` and `ifconfig unplumb` commands on the AP interface. (To manually switch an active path, use the `apconfig`(1M) command.)

Caution – Detaching network interfaces may affect NFS client systems.

Non-Network Devices

All non-network devices must be closed before they are detached. In the Hostview device display and in the `drshow`(1M) I/O listing, there is an open count field that indicates how many processes have opened particular devices. To see which processes have these devices open, use the `fuser`(1M) command on the domain.

You must perform certain tasks for non-network devices. Although the following list of tasks implies a sequence of order, strict adherance to the order is not necessary.

1. If the redundancy features of Alternate Pathing or Solstice DiskSuite mirroring are used to access a device connected to the board, reconfigure these subsystems so that the device or network is accessible using controllers on other system boards. Note that for Alternate Pathing 2.1, the system automatically switches the disk devices to an alternate interface if one is available.
2. Unmount file systems, including Solstice DiskSuite metadevices that have a board-resident partition (for example, `umount /partit`).
3. Remove Alternate Pathing or Solstice DiskSuite databases from board-resident partitions. The location of Alternate Pathing or Solstice DiskSuite databases is explicitly chosen by the user and can be changed.
4. Remove any private regions used by Sun Enterprise Volume Manager™ or Veritas Volume Manager. Volume manager by default uses a private region on each device that it controls, so such devices must be removed from volume manager control before they can be detached.
5. Remove disk partitions from the swap configuration by using `swap`(1M).
6. Either kill any process that directly opens a device or raw partition, or direct it to close the open device on the board.
7. If a detach-unsafe device is present on the board, close all instances of the device and use `modunload`(1M) to unload the driver.
8. Kill all of the real-time processes that are open if the operating environment must be suspended.

Caution – Unmounting shared file systems by using the `share`(1M) utility may affect NFS client systems.

Processes

You must perform certain tasks for processes. Although the following list of tasks implies a sequence of order, strict adherance to the order is not necessary.

1. If the operating environment must be suspended, kill all of the real-time processes that are running.
2. Kill, or unbind, any processes that are bound to on-board processors.

Processes bound to the processors of a board prevent that board from being detached. You can use `pbind`(1M) to rebind them to other processors.

Processors

The boot processor is responsible for maintaining the netcon BBSRAM buffer. Before detaching a board on which the boot processor resides, the `dr_daemon`(1M) must assign the boot processor role to another active (online) processor.

Reconfiguration After a DR Operation

This section describes how to reconfigure your domain after you have attached or detached a system board.

Note – As of the Solaris 8 GA release, manual reconfiguration is no longer needed. A new DDI subsystem, `devfsadm`, completes all of the reconfiguration tasks.

The DR user interface enables you reconfigure the domain after a DR Attach or DR Detach operation. The reconfiguration sequence is the same as the reconfiguration boot sequence (`boot -r`):

```
drvconfig; devlinks; disks; ports; tapes;
```

When you execute the reconfiguration sequence after you attach a board, device path names not previously seen by the domain are written to the `/etc/path_to_inst` file. The same path names are also added to the `/devices` hierarchy, and links to them are created in the `/dev` directory.

When to Reconfigure

You should reconfigure the domain if any of the following conditions occur:

- Board Addition – When you add a board to a domain, you must execute the reconfiguration sequence to configure the I/O devices that are associated with the board.
- Board Deletion – If you remove a board that is not to be replaced, you may, but do not have to, execute the reconfiguration sequence to clean up the `/dev` links.
- Board Replacement – If you remove a board then reinsert it in a different slot or if you replace a board with another board that has different I/O devices, you must execute the reconfiguration sequence to configure the I/O devices that are associated with the board. However, if you replace a board with another board that hosts the *same* set of I/O devices, inserting the replacement into the *same* slot, you do not need to execute the reconfiguration sequence. But, be sure to insert a replacement board into the same slot that was vacated to retain the original mapping of `/dev` links to physical names.

Disk Devices

Disk controllers are numbered consecutively as the `disks`(1M) program encounters them. All disk partitions are assigned `/dev` names according to the disk controller number that `disks`(1M) assigns. For example, all disk partitions that are accessible using disk controller 1 are named `/dev/dsk/c`*`X`*`t`*`Y`*`dZs`*`W`*

where:

`X` is the disk controller number,

Y, in most cases, corresponds to the disk target number,

Z corresponds to the logical unit number, and

W corresponds to the partition number.

When the reconfiguration sequence is executed after a board is detached, the `/dev` links for all of the disk partitions on that board are deleted. The remaining boards retain their current numbering. Disk controllers on a newly inserted board are assigned the next available lowest number by `disks`(1M).

Note – The disk controller number is part of the `/dev` link name used to access the disk. If that number changes during the reconfiguration sequence, the `/dev` link name also changes. This change may affect file system tables and software, such as Solstice DiskSuite™, which use the `/dev` link names. Update `/etc/vfstab` files and execute other administrative actions to change the `/dev` link names.

DR and AP Interaction

DR notifies the AP subsystem when system boards are attached, detached, or placed in the drain state. In addition, DR queries AP about which controllers are in the AP database and their status (active or inactive). This communication occurs between the `dr_daemon`(1M) and `ap_daemon`(1M). If the `ap_daemon`(1M) is not present, an error message is placed in the syslog messages buffer of the domain and DR operations continue without error. To disable this interaction, use the `-a` option when you invoke `dr_daemon`(1M). Refer to the `dr_daemon`(1M) man page in the *Sun Enterprise 10000 Dynamic Reconfiguration Reference Manual*.

If you are using AP version 2.1, the operating environment automatically switches off the active disk controllers on outgoing boards during the complete-detach phase of DR. If you are using AP version 2.0, you need to manually switch off the active disk controllers before you start the complete-detach phase. For the Solaris 8 operating environment, you must upgrade to AP version 2.3. For more information about DR and AP interaction, see the *Sun Enterprise Servers Alternate Pathing 2.3 User Guide*. For more information about AP and SDS, refer to the *RAS Companion*.

RPC Time-Out or Loss of Connection

The `dr_daemon`(1M), which runs in each domain, communicates with Hostview and the `dr`(1M) shell application (both of which run on the SSP) by way of Remote Procedure Calls (RPCs). If an RPC time-out or connection failure is reported during a DR operation, check the domain. The daemon must be configured in the `/etc/inetd.conf` file of the domain. The following line (which appears on a single line) must be present in the file:

```
300326/4 tli rpc/tcp wait root \
/platform/SUNW,Ultra-Enterprise-10000/lib/dr_daemon/ dr_daemon
```

If the DR daemon is configured in `/etc/inetd.conf`, kill the `dr_daemon`(1M) if it is currently running. In addition, send a HUP signal to the `inetd`(1M) daemon to cause it to re-read the `inetd.conf`(4) configuration file:

```
# kill dr_daemon_pid
# kill -HUP inetd_pid
```

In the first command, *dr_daemon_pid* is the process ID of the DR daemon. In the second command, *inetd_pid* is the process ID of the `inetd`(1M) daemon. You can check `/var/adm/messages` for possible error messages from `inetd`(1M) if it is having trouble starting the `dr_daemon`(1M). The DR daemon executable file should exist in the `/platform/SUNW,Ultra-Enterprise-10000/lib` directory.

At this point, try the DR operation again, starting from the beginning.

System Quiescence Operation

During a DR Detach operation on a system board with nonpageable OBP or kernel memory, the operating environment is briefly quiesced; that is, all operating environment and device activity on the domain centerplane must cease during the critical phase of the operation. The quiescence only affects the target domain; other domains in the system are not affected.

Before a board can be detached, the operating environment must temporarily suspend all processes, processors, and device activities. If the operating environment cannot quiesce, it displays its reasons, which may include the following:

- Real-time processes are running in the domain.
- A device that cannot be quiesced by the operating environment (that is, a suspend-unsafe device) is open.

The conditions that cause processes not to suspend are generally temporary in nature. You can retry the operation until the quiescence succeeds.

A failure to quiesce due to real-time processes or open suspend-unsafe devices is known as a forcible condition. You have the option of performing either a retry or forced retry. When you force the quiescence, you give the operating environment permission to continue with the quiescence even if forcible conditions are still present.

Caution – Exercise care when using the `force` option.

If a real-time process is running, determine if suspending the process would produce an adverse effect on the functions performed by the process. If not, you can force the operating environment to quiesce. (To force a quiescence, you can either click the `Force` button within Hostview as described in “To Detach a Board With Hostview” in the *Sun Enterprise 10000 Dynamic Reconfiguration Guide* in the SSP 3.3 AnswerBook2 collection, or enter the `complete_detach`(1M) command with the `force` option within the `dr`(1M) shell application. Otherwise, you can abort the operation and try again later.

If any suspend-unsafe device is open and cannot be closed, you can manually suspend the device, and then force the operating environment to quiesce. After the operating environment resumes, you can manually resume the device (see “Suspend-Safe/Suspend-Unsafe Devices” on page 12).

Suspend-Safe/Suspend-Unsafe Devices

A suspend-safe device is one that does not access the domain centerplane (for example, it does not access memory or interrupt the system) while the operating environment is quiesced. A driver is considered suspend-safe if it supports operating environment quiescence (suspend/resume) and guarantees that when a suspend request is successfully completed, the device that the driver manages will not attempt to access the domain centerplane, even if the device is open when the suspend request is made. All other I/O devices are suspend-unsafe when open.

Note – At the time of this printing, the drivers released by Sun Microsystems™ that are known to be suspend-safe are `st`, `sd`, `isp`, `esp`, `fas`, `sbus`, `pci`, `pci-pci`, `qfe`, `hme` (SunFastEthernet™), `nf` (NPI-FDDI), `qe` (Quad Ethernet), `le` (Lance Ethernet), the SSA drivers (`soc`, `pln`, and `ssd`), and the Sun StorEdge™ A5000 drivers (`sf`, `socal`, `ses`).

To enable device suspension for the `soc` and `pln` drivers, you must edit the `/etc/system` file so that the `pln_enable_detach_suspend` and `soc_enable_detach_suspend` variables are set to 1, as in the following example:

```
set pln:pln_enable_detach_suspend=1
set soc:soc_enable_detach_suspend=1
```

The operating environment refuses a quiesce request if a suspend-unsafe device is open. If you can manually suspend the device, you can force the operating environment to quiesce. To manually suspend the device, you may have to close the device by killing the processes that have it open, ask users not to use the device, or disconnect the cables. For example, if a device that allows asynchronous unsolicited input is open, you can disconnect its cables prior to quiescing the operating environment, preventing traffic from arriving at the device and the device from accessing the domain centerplane. You can reconnect the cables after the operating environment resumes. If you cannot make a device suspend its access to the domain centerplane, you should not force the operating environment to quiesce. Doing so could cause a domain to crash or hang. Instead, postpone the DR operation until the suspend-unsafe device is no longer open.

Caution – If you attempt a forced quiesce operation while activity is occurring on a suspend-unsafe device, the domain may hang. However, if the domain hangs, it will not affect other domains that are running on the Sun Enterprise 10000 system.

Special Handling for Tape Devices

For the Solaris 8 operating environment, tape devices that are natively supported by Sun Microsystems™ are suspend-safe and detach-safe (refer to the `st`(7D) man page for a list of natively-supported drives). If a system board that you are detaching contains a natively-supported tape device, you can safely detach the board without suspending the device. If you want to use a tape device that is not natively supported by Sun Microsystems, you can use it, but you should make it detach-safe. To ensure correct input/output and DR operations, you need to make a suitable entry in `/kernel/drv/st.conf` with the `ST_UNLOADABLE` (0x0400) flag set in the entry (refer to the `st`(7D) man page for more information). After you update `st.conf`, you must reboot the domain to process the new entry.

Special Handling of Sun StorEdge A3000

The Sun StorEdge™ A3000 (formerly known as the RSM Array 2000) has dual controller paths with automatic load balancing and automatic failover. To detach a system board that has one or both of the StorEdge A3000 controllers, the controllers on the board that is being detached must be idle or offline. You can take these controllers offline manually by using the `rm6` or `rdacutil` programs before you attempt to detach the system board.

DR and DDI

Not all drivers support the Sun Enterprise 10000 system Dynamic Reconfiguration (DR) feature. To support DR, a driver must be able to perform three basic DDI/DKI (Device Driver Interface/Device Kernel Interface) functions, `DDI_DETACH` and `DDI_SUSPEND`/`DDI_RESUME`. These functions impact DR in different ways.

DR and DDI_DETACH

You can detach a system board that hosts a device only if the driver for that device supports the DDI_DETACH interface, or is not currently loaded. DDI_DETACH provides the ability to detach a particular instance of a driver without impacting other instances that are servicing other devices. A driver that supports DDI_DETACH is called *detach-safe*; a driver that does not support DDI_DETACH is called *detach-unsafe*.

Detaching a detach-unsafe driver that is loaded involves the following process.

- Stopping all usage of the controller for the detach-unsafe device and all other controllers of the same type on all of the boards in the domain.

  Because the detach-unsafe driver must be unloaded, you must stop usage of that controller type on *all* of the system boards in the domain. The remaining controllers can be used again after the DR Detach is complete.

- Using standard Solaris interfaces to manually close and to unload all such drivers on the board.

  Refer to the modunload(1M) man page in the *SunOS Reference Manual*.

- Detaching the system board in the normal fashion.

If you cannot accomplish the above process, you can reboot the domain with the board blacklisted (refer to the blacklist(4) man page), so the board can be removed later.

Note – Many third-party drivers (those purchased from vendors other than Sun Microsystems) do not support the standard Solaris modunload(1M) interface. Conditions that invoke the functions occur infrequently during normal operation and the functions are sometimes missing or work improperly. Sun Microsystems suggests that you test these driver functions during the qualification and installation phases of any third-party device.

DR and DDI_SUSPEND/DDI_RESUME

To perform a DR Detach of a board that contains nonpageable memory, the domain must be quiesced. Memory can be detached only when all of the drivers throughout the entire domain (not just on the board being detached) either support the DDI_SUSPEND/DDI_RESUME driver interface, or are closed. Drivers that support these DDI functions are called *suspend-safe*; drivers that do not are called *suspend-unsafe*.

The most straightforward way to quiesce a domain is to close any suspend-unsafe devices. For each network driver you must execute the `ifconfig`(1M) command with its `down` parameter, then again with its `unplumb` parameter (refer to the `ifconfig`(1M) man page for more information).

Note – It should be possible to unplumb all network drivers. However, this action is rarely tested in normal environments and may result in driver error conditions. If you use DR, Sun Microsystems suggests that you test these driver functions during the qualification and installation phases of any *suspend-unsafe* device.

If the system refuses to quiesce because a *suspend-unsafe* driver is open, you can force the operating domain to quiesce. Doing so forces the operating environment to permit the detach. Note that, although a detach can be forced to proceed when there are open *suspend-unsafe* devices in the system, it is not possible to force a detach when a *detach-unsafe* device resides on the board and its driver is loaded.

To successfully force the operating environment to quiesce, you must manually quiesce the controller. Procedures to do that, if any, are device-specific. The device must not transfer any data, reference memory, or generate interrupts during the operation. Be sure to test any procedures used to quiesce the controller while it is open prior to executing them on a production system.

Caution – Using the `force` option to quiesce the operating environment, without first successfully quiescing the controller, can result in a domain failure and subsequent reboot.

CHAPTER 3

Configuring InterDomain Networks

This chapter contains information about the automatic activation of the logical network interfaces, the tunable parameters that affect the operation and performance of an IDN, and the instructions for setting the tunable parameters.

Domain IP Addresses

Any standard Transmission Control Protocol/Internet Protocol (TCP/IP) network interface must have an assigned IP address so that the domains can communicate through the interface. To establish an IDN connection, a set of domains must also have assigned IP addresses that are unique among any addresses or subnets you expect to access from within the domain. These addresses need to be visible only to the domains within that IDN. If you want to use a domain as a router between external hosts and other domains to which it is connected by way of an IDN, you must choose the IP addresses with consideration for the network configuration in which the Sun Enterprise 10000 server resides. Typically, each logical IDN interface is configured as a separate IP subnet. The IDN software makes no association between IDN member domains and IP addresses, so you are free to choose any IP address that is appropriate for your network environment. The associated host names for the assigned IP addresses must be entered in the `/etc/hostname.idn`*X* file, where `idn`*X* represents the logical IDN interface to which a particular IP address has been assigned. This enables the network to come up automatically upon bootup of the domain.

Note that to enable the IDN driver and to permit a domain to become an IDN member, you must create at least one `/etc/hostname.idn`*X* file so that the IDN driver is automatically loaded when the domain is booted. Only after the IDN driver is loaded will the SSP recognize the domain as an IDN candidate.

Note – By default, there are eight possible logical interfaces, `idn0` through `idn7`. This value can be tuned to a maximum of 32 (`idn0` through `idn31`) by using the IDN tunable parameters and the `idn.conf`(4) file. Only domains with the same active `idn`*X* interface can communicate with each other on the same IDN subnet.

Ethernet and Physical Addresses

The `ifconfig`(1M) command allows you to dynamically change the Ethernet address or the physical address of a network interface. However, due to the point-to-point nature of IDNs, the system must maintain identification information in the Ethernet address to determine where to direct packets. As a result, the IDN driver does not allow you to change Ethernet or physical addresses of IDN interfaces. This is not a problem because an IDN is a private subnet. This assumption remains valid even if a network interface card is installed with the same physical address as an IDN interface.

Automatic Activation of the Logical Network Interfaces

The logical network interface of an IDN (for example, `idn0`, `idn1`, and so forth) is treated the same way as network interfaces of more traditional network interface cards. Although all IDN interfaces use the same physical link, the interfaces are logically separate network interfaces; therefore, each IDN interface requires a unique `/etc/hostname.idn`*X* file to invoke automatic network plumbing when the domain is booted.

The `/etc/hostname.idn`*X* file contains only one entry: the hostname or IP address associated with the IDN interface. If `idn0` is the logical network interface for the IDN, `/etc/hostname.idn`*X* would be named `/etc/hostname.idn0`, and the file would contain a unique hostname that is associated with the IDN interface.

For more information about the contents of the `/etc/hostname.idn`*X* file, refer to the *Sun Enterprise 10000 InterDomain Networks User Guide*. Also, refer to the *TCP/IP and Data Communications Administration Guide* for more information on TCP/IP configuration files.

▼ To Enable Automatic Activation of Logical Network Interfaces

Perform the following steps to create the `/etc/hostname.idn`*X* file:

1. **Open a new file in your text editor.**
2. **Type in the name or IP address of the IDN logical network interface.**
3. **Save the file as** `/etc/hostname.idn`*X* **where** *X* **corresponds to the instance of the IDN driver that you want to activate at boot time.**

   If a domain is a member of an IDN, the domain is automatically linked at boot time with the other IDN members that are booted, as displayed by the `domain_status`(1M) command on the SSP. In conjunction with the `/etc/hostname.idn`*X* files, the Solaris `rc` scripts enable the logical network interfaces over the IDN. The IDN can then be used as a standard TCP/IP network between the domains.

Note – Automatic linking of the IDN requires services provided by the SSP. The SSP event detection daemon, `edd`(1M), is responsible for recognizing that a domain has booted and executes the IDN event handler to perform the actual linking. Depending on the load on the SSP, there may be latencies in the time required for the boot event to be recognized and for the IDN event handler to process the link. As a result, it is possible that the domain may complete its boot cycle before the IDN link to that domain is fully operational. This latency should be no more than a matter of seconds.

Plumbing IDN Interfaces

You plumb IDN interfaces the same way you plumb any other network interface. The information is contained here for convenience only.

▼ To Plumb an IDN Interface

You must perform the following steps for each IDN interface in each domain that is linked to the IDN. Note that the domain does not need to be linked to the IDN before you perform these steps.

1. **Plumb the IDN interface within each domain.**

```
# ifconfig idn0 plumb
```

In the example above, `idn0` is the IDN interface name that is based on the IPv4 usage. Refer to the IPv6 documentation for the correct usage for IPv6. Note that IPv6 is not supported in the Solaris 7 operating environment.

The *IP_address* is defined as the IP address assigned to the given IDN interface for the respective host (refer to "Domain IP Addresses" in the *Sun Enterprise 10000 InterDomain Networks User Guide* and the `hosts`(4) man page for more information).

2. **Configure the IDN interface.**

```
# ifconfig idn0 IP_address netmask 255.255.255.0 \
broadcast IP_subnet_address up
```

The example above assumes that you are setting up a basic IDN. If you plan to use a site-specific netmask, replace the netmask value with the site-specific value.

▼ To Unplumb an IDN Interface

You do not need to unplumb the IDN interfaces in a domain that you are unlinking from an IDN. However, to dismantle an entire IDN, you must perform the following steps for each IDN interface in each domain in the IDN.

1. **Unconfigure the IDN interface.**

```
# ifconfig idn0 down
```

This step dismantles the TCP/IP stack for the specified IDN interface.

2. **Unplumb the IDN interface.**

```
# ifconfig idn0 unplumb
```

Tunable Variables and Parameters

There are several variables and parameters that affect the performance and resource usage of IDNs. This section explains how to set the variables and parameters and includes the minimum, maximum, and default values.

OpenBoot PROM Variable

The OpenBoot™ PROM (OBP) has one IDN-related variable that you must modify to enable IDNs: the shared memory region (SMR) size variable, `idn-smr-size`. This variable specifies the size of the SMR in megabytes. A value of zero disables IDN networking. A nonzero value indicates the number of megabytes of kernel space to reserve for the SMR. The default value of `idn-smr-size` is zero (0).

The larger the SMR, the greater the number of available buffers for data transfers. However, past a certain threshold, no additional benefit is gained by having a larger SMR. The suggested value for `idn-smr-size` is 32 megabytes, which should be adequate for most usages. The maximum value is 96 megabytes.

The value of `idn-smr-size` can be set only at the OBP prompt. You must reboot the domain before the new value can take effect. You can, however, reduce the actual size of the SMR by using the `idn_smr_size` variable in the `idn.conf` file.

Note – All domains within an IDN must have the same value for `idn-smr-size`. If any domain does not have the proper `idn-smr-size` value, or if you want to change the value for the entire IDN, you must reboot the affected domains to the OBP prompt and reset this variable.

▼ To Set OBP Variables

1. **In a `netcon`(1M) window, log in to the domain as superuser.**
2. **Boot, or halt, the domain to the OBP prompt and set the variable by using the `setenv` command, as in the following example:**

```
<#Ø> ok setenv idn-smr-size size
```

3. **Reboot the domain.**

4. **After the reboot has succeeded, check the OBP settings.**

```
<#0> ok cd /memory
<#0> ok .properties
```

The second command produces a list of the OBP variables with their associated settings, as in the following example:

```
idn-smr-size              00 00 00 20
idn-smr-addr              00 00 00 0a 7d 3f 00 00 00 00 00 00 02 00 00 00
dr-max-mem                00 00 9c 40
reg                       0000000a 00000000 00000000 80000000
available                 0000000a 7fff0000 00000000 00004000
                          0000000a 7fcd8000 00000000 00016000
                          0000000a 00000000 00000000 7189e000
name                      memory
```

If the SMR has been properly allocated, the value of `idn-smr-addr` should be non-zero, representing the base physical address of the SMR (for example, `0xA7D3F0000`) and the size in bytes (for example, `0x2000000`).

ndd(1M) Driver Parameters

You can change ndd(1M) driver parameters to tune the system for optimal performance and resource usage. This section explains which parameters you can change, shows you how to change the parameters, and lists the ranges of values you can use with each parameter.

▼ To Set the ndd(1M) Driver Parameters

1. **Read the current parameter setting.**

```
# ndd /dev/idn parameter
```

Use the following command to view a list of all of the ndd(1M) parameters that are supported by the IDN driver.

```
# ndd /dev/idn "?"
```

2. **Change the driver parameter.**

```
# ndd -set /dev/idn parameter value
```

You must use the `-set` syntax to modify the driver parameters mentioned in this section. Also, unless otherwise mentioned, all of the driver parameters in this section can be changed at any time.

The following table includes the name of the parameters that can be read by using the ndd(1M) command and a short description of the parameters. For more information about ndd(1M) usage, see the ndd(1M) man page.

TABLE 3-1 ndd(1M) Parameters

Name	Min.	Max.	Default	Description
idn_modunloadable	0	1	0	Is the binary flag that indicates whether the IDN driver is unloadable or not (assuming that it is not in use). The flag is turned off with a value of zero (0), and it is turned on with a value of one (1). The value can be changed at any time.
idn_slabpool	n/a	n/a	n/a	If the domain is connected and if it is the master of the IDN, this parameter displays the IDN slab pool, indicating the number of slabs that are available and which slabs have been allocated for each domain. The value is read only.
idn_buffers	n/a	n/a	n/a	Displays the number of outstanding SMR I/O buffers that the domain has with respect to the domains with which it is connected. The value is read only.
idn_mboxtbl	n/a	n/a	n/a	Displays the mailbox table allocated to the domain. If the domain is not a member of an IDN, then no table is displayed. The information displayed includes the mailbox header cookie, the value of the ready and/or active pointers, and an indication of whether or not the respective channel server is ready and/or active. The value is read only.

TABLE 3-1 ndd(1M) Parameters *(Continued)*

Name	Min.	Max.	Default	Description
idn_mboxtbl_all	n/a	n/a	n/a	Displays the same information as idn_mboxtbl; however, it displays it for the entire IDN. This parameter is pertinent only when it is performed within the context of the master domain because it maintains a pointer to the global mailbox area.
idn_mainmbox	n/a	n/a	n/a	Contains the detailed information of the mailbox management structures that are maintained by the domain for send and receive mailboxes to the other IDN member domains. The value is read only.
idn_global	n/a	n/a	n/a	Displays the global state information pertaining to the domain (for example, the active channels, the number of domains to which it is connected, and the physical address of the SMR). It also displays a summary of the connection state of each domain in the IDN. The value is read only.
idn_domain	n/a	n/a	n/a	Displays domain specific state information pertaining to the domain (for example, the outstanding timer count, the vote ticket, and the outstanding buffer count). The value is read only.
idn_domain_all	n/a	n/a	n/a	Displays information that is similar to idn_domain, but the information includes all of the domains to which the domain is connected. The value is read only.
idn_bind_net	n/a	n/a	n/a	Allows the user to bind specific channel servers (interfaces) to specific processors within the domain, allowing finer control over which processors within the domain actually drive the reception of IDN data. By default, the servers are unbound; thus, they compete directly for processing time with normal threads. The argument is given in the form channel=cpuid. For example, 0=25 would bind the channel server that is responsible for processing data received on the idn0 interface to cpuid 25. The value can be changed at any time.

driver.conf(4) Parameters

IDNs permit certain tunable and/or configuration parameters to be set by using the driver.conf(4) file for the IDN driver. The file is located in the following path:

/platform/SUNW,Ultra-Enterprise-10000/kernel/drv/idn.conf

You must edit the driver.conf(4) file to change these parameters. Most of the parameters are considered global. Only the bind_cpu parameter is considered per instance (interface). The values of the parameters take affect when the driver is loaded by using the modload(1M) command.

The procedure you use to set the IDN parameters depends on the current state of the domain. If the domain is up and running, but not linked to an IDN, you can set the IDN parameters without rebooting the domain by following the instructions in "To Set IDN Parameters Without a Reboot" on page 25. If the domain is not running, or if you will be rebooting the domain, you can set the IDN parameters by following the instructions in "To Set IDN Parameters With a Reboot" on page 26.

▼ To Set IDN Parameters Without a Reboot

1. **Make sure that the domain is not linked to an IDN.**
2. **In a** netcon**(1M) window, change directories to the directory that contains the** idn.conf **file.**

```
% cd /platform/SUNW,Ultra-Enterprise-10000/kernel/drv/
```

3. **Edit the** idn.conf **file so that it reflects the new values that you want to use.**
4. **Unplumb all of the IDN network interfaces.**
5. **Use the** ndd**(1M) command to set the** idn_modunloadable **parameter to the proper value.**

```
% ndd -set /dev/idn idn_modunloadable 1
```

6. **Use the** modunload**(1M) command to unload the IDN driver module.**

```
% modunload -i id
```

The value of id must correspond with the ID of the IDN module ID number. Refer to the modinfo(1M) man page for more information on how to obtain the module ID number.

7. **Replumb the IDN network interfaces.**

▼ To Set IDN Parameters With a Reboot

1. **In a** `netcon`**(1M) window, change directories to the directory that contains the** `idn.conf` **file.**

```
% cd /platform/SUNW,Ultra-Enterprise-10000/kernel/drv/
```

2. **Use a text editor to edit the file so that it contains the parameters and the values for the IDN.**

   The following example contains a sample `idn.conf` file.

```
name="idn" parent="pseudo" instance=0 bind_cpu=10;
name="idn" parent="pseudo" instance=1;
name="idn" parent="pseudo" instance=2 bind_cpu=35;
idn_pil=4;
idn_protocol_nservers=2;
```

   For all of the required parameters, you must edit the `idn.conf` file for each of the domains in the same IDN. For all other parameters, you can edit the `idn.conf` file of that domain only.

   An entry can use multiple lines; however, it must be terminated by a semicolon. In the example, the instance 0 channel server (`idn0`) will be bound to CPU 10, assuming it is in the system. The instance 1 channel server for (`idn1`) will not be bound to any CPU in the system, and the instance 2 channel server for (`idn2`) will be bound to CPU 35, assuming it is in the system.

3. **Reboot the domain(s).**

   If you changed the settings of the parameters that are required to match, you must reboot each domain in the IDN. If you changed the settings of the requirements that do not need to match, you can reboot a single domain in the IDN. See Section "Required Parameter Matching" on page 31 for a list of the parameters that must match.

idn.conf(4) File

You can define the values of certain parameters in the idn.conf(4) file so that they are set when the IDN is loaded by using the modload(1M) command. You can also add IDN instances to this file. Edit the idn.conf(4) file for each IDN instance with the following line in which *n* equals the number of the instance.

```
name=”idn” parent=”pseudo” instance=n;
```

Note – All idn.conf(4) file parameters can be changed while the domain is linked to the IDN; however, the domain must be rebooted before the values take affect.

The following table contains the name of the parameters; the minimum, maximum, and default values of the parameters; and the units in which they are given.

Caution – The parameters in the following table are meant to be used only by trained IDN users. Modification of some of the values could negatively affect the behavior of the IDN.

TABLE 3-2 IDN idn.conf(4) File Parameters

Name	Min.	Max.	Default	Description
bind_cpu	n/a	n/a	-1	Specifies which cpuid to bind the respective channel server after it is brought online. This parameter must be associated with a particular CPU instance. If the specified cpuid is not a valid CPU in the domain, the channel server will remain unbound. The value is given as the ID of the CPU (-1 equals unbound).
idn_awolmsg_interval	0	3600	60	Controls the frequency with which AWOL messages are displayed on the console on a per domain basis. The value is given in seconds.
idn_checksum	0	1	1	Is the binary flag that indicates whether or not checksum validation is turned on for SMR mailboxes. The flag is turned off with a value of zero (0), and it is turned on with a value of one (1).

TABLE 3-2 IDN `idn.conf`(4) File Parameters *(Continued)*

Name	Min.	Max.	Default	Description
`idn_dmv_pending_max`	8	512	128	Controls the maximum number of outstanding DMV interrupts that a single processor can have pending to the IDN driver. It also describes the number of queue structures used to encapsulate the data of an incoming cross-domain interrupt. The value is given as a number.
`idn_history`	0	1	0	Is the binary flag that indicates whether or not IDN should turn on internal logging of certain IDN events. This is intended only for problem analysis to gather information for later debugging by support personnel. A value of zero (0) turns off the flag, and a value of one (1), turns on the flag.
`idn_hiwat`	1024	1048576	262144	Controls the high-water mark of the IDN STREAM queue. This value is given in bytes.
`idn_lowat`	1	524288	1	Controls the low-water mark of the IDN STREAM queue. This value is given in bytes.
`idn_max_nets`	1	32	8	Controls the maximum number of network channels or interfaces that can be plumbed onto the IDN driver. The value is given in general units or counts.
`idn_mbox_per_net`	31	511	127	Controls the number of mailbox entries per mailbox table (channel and/or interface). The value must be an odd number. It is given in general units or counts.
`idn_msgwait_cfg`	10	300	40	Controls the minimum amount of time to wait for a response to a CFG (configuration) message. The value is given in seconds.
`idn_msgwait_cmd`	10	300	40	Controls the minimum amount of time to wait for a response to a CMD (command) message (typically to the master domain). The value is given in seconds.
`idn_msgwait_con`	10	300	20	Controls the minimum amount of time to wait for a response to a CON (connection) message. The value is given in seconds.

TABLE 3-2 IDN `idn.conf`(4) File Parameters *(Continued)*

Name	Min.	Max.	Default	Description
`idn_msgwait_data`	10	300	30	Controls the minimum amount of time to wait for a response to a DATA (disconnect) wake-up call. The value is given in seconds.
`idn_msgwait_fin`	10	300	40	Controls the minimum amount of time to wait for a response to a FIN (disconnect) message. The value is given in seconds.
`idn_msgwait_nego`	10	300	20	Controls the minimum amount of time to wait for a response to a NEGO (negotiation) message. The value is given in seconds.
`idn_netsvr_spin_count`	0	10000	500	Controls the iterative count that a channel server will poll for incoming packets before it gives up the processor. The value is given in general units or counts.
`idn_netsvr_wait_max`	0	6000	1600	Controls the maximum number of clock ticks that channel server will sleep before it enters a hard sleep.
`idn_netsvr_wait_min`	0	3000	40	Controls the initial clock-tick value that a channel server will sleep when no incoming data packets have been found. The value is given in clock ticks (100 ticks equals one second).
`idn_netsvr_wait_shift`	1	5	1	Represents how much the sleep time of the channel server increases each time it awakes and does not find packets. A value of one (1) causes the time to be doubled on each interval. The sleep time increases until it reaches the maximum value designated by `idn_netsvr_wait_max`. The value is given in general units or counts.
`idn_nwr_size`	0	Entire SMR	Entire SMR	Controls the size of the network region (NWR) portion of the SMR that is used for network-based communication. The value is given in megabytes.
`idn_pil`	1	9	8	Controls the priority level of the soft interrupt, at which cross-domain interrupts are processed. The value is given as a number.

TABLE 3-2 IDN `idn.conf`(4) File Parameters *(Continued)*

Name	Min.	Max.	Default	Description
`idn_protocol_nservers`	1	16	4	Controls the number of threads that are delegated to processing IDN connection management messages from remote IDN member domains. The value is given as a number.
`idn_reclaim_max`	0	128	0	Controls the maximum number of outstanding, unclaimed buffers the domain attempts to reclaim. A value of zero (0) causes the domain to reclaim as many as possible after the minimum threshold (`idn_reclaim_min`) is reached. The value is given in buffers.
`idn_reclaim_min`	1	128	5	Controls the threshold of outstanding (unclaimed) buffers, past which the domain attempts to reclaim the buffers. The value is given in buffers.
`idn_retryfreq_con`	1	60	2	Controls the minimum amount of time between retries to confirm that an incoming domain has reached the CON (connect) phase. The value is given in seconds.
`idn_retryfreq_fin`	1	60	3	Controls the minimum amount of time between retries to confirm that an outgoing domain has reached the FIN (disconnect) phase. The value is given in seconds.
`idn_retryfreq_nego`	1	60	2	Controls the minimum amount of time between retries to initiate an IDN connection. The value is given in seconds.
`idn_sigbpil`	1	9	3	Controls the priority level of the soft interrupt, at which SSP sigblock requests are processed. The value is given as a number.
`idn_slab_bufcount`	4	1024	32	Controls the number of buffers to allocate per slab. The value is given in buffers.
`idn_slab_mintotal`	2	16	8	Controls the minimum number of available slabs that the master domain maintains. The master domain requests the slave domains to return the unused slabs if the total of available slabs falls below the value of this variable. The value is given in slabs.

TABLE 3-2 IDN `idn.conf`(4) File Parameters *(Continued)*

Name	Min.	Max.	Default	Description
`idn_slab_prealloc`	0	10	0	Controls the number of slabs to pre-allocate when the domain is linked to an IDN. The value is given in slabs.
`idn_smr_bufsize`	512	524288	16384	Controls the size of an SMR I/O buffer, which translates to the IDN MTU size. The value is given in bytes and as a power of 2.
`idn_smr_size`	0	Entire SMR	0	The size of the SMR is limited by the value of the OBP variable `idn-smr-size`. The size of the SMR is determined by the minimum value of the `idn-smr-size` variable and by the minimum value of the `idn_smr_size` parameter. If `idn-smr-size` is set to zero, the OBP variable overrides the value of the `idn.conf`(4) parameter. This value is given in megabytes.
`idn_window_incr`	0	32	8	Controls the value by which `idn_window_max` is increased for each additional active channel and/or interface. The value is given in buffers.
`idn_window_max`	8	256	64	Controls the base threshold of outstanding buffers, past which the domain stops sending additional data packets to the respective domain. The value is given in buffers.

Required Parameter Matching

Certain IDN parameters must be the same across all of the domains in the same IDN. During the exchange of configuration information when the domain is linked, each domain verifies that the received information matches the local parameters before it allows the link operation to proceed. The following list contains the name of the parameters that must be the same across all of the domains in an IDN.

- `idn_nwr_size`
- `idn_smr_bufsize`
- `idn_slab_bufcount`
- `idn_max_nets`
- `idn_mbox_per_net`
- `idn_checksum`

Kernel Statistics

The IDN driver supports the standard Solaris kernel statistics mechanism, `kstat`(3K). In addition to the minimum set required to support `netstat`(1M) reporting, the IDN driver reports additional statistics that can be useful for either performance tuning or configuration management. These statistics are most easily available through the standard `netstat`(1M) or `kstat`(1M) command line utilities.

You can request all of the statistics by using the syntax in the following example. The example includes a sample of the statistics you will receive by using the `idn` and `idn0` arguments.

```
# netstat -k idn
idn:
curtime 2048474 reconfigs 0 reconfig_last 0 reaps 0 reap_last 0
links 1 link_last 2042885 unlinks 1 unlink_last 2045246 buf_fail 1
buf_fail_last 2042935 slab_fail 1 slab_fail_last 2042935
reap_count 0 dropped_intrs 0

# netstat -k idn0
idn0:
ipackets 3 ierrors 0 opackets 0 oerrors 0 collisions 0
rx_collisions 0 crc 0 buff 0 nolink 0 linkdown 0 inits 5 nocanput 0
allocbfail 0 notbufs 0 reclaim 0 smraddr 0 txmax 0 txfull 0 xdcall 3
sigsvr 10 mboxcrc 0 rbytes 238 obytes 238 multircv 0 multixmt 0
brdcstrcv 0 brdcstxmt 4 norcvbuf 0 noxmtbuf 0 ipackets64 3
opackets64 3 rbytes64 238 obytes64 238 fcs_errors 0
macxmt_errors 0 toolong_errors 0 macrcv_errors 0
```

You can request the statistics for an individual name or interface, as in the following example, which includes `idn0` and `idn1` as the logical network interfaces. The amounts in the examples are for informational purposes only; the output you receive can differ significantly.

```
# netstat -k idn0 idn1

idn0:
ipackets 1386286 ierrors 0 opackets 1312137 oerrors 0 collisions 0
rx_collisions 0 crc 0 buff 0 nolink 0 linkdown 3561 inits 3
nocanput 131735 allocbfail 0 notbufs 0 reclaim 0 smraddr 0 txmax 0
txfull 0 xdcall 68783 sigsvr 63444 mboxcrc 0 rbytes 291362843
obytes 4225747350 multircv 0 multixmt 0 brdcstrcv 0 brdcstxmt 21
norcvbuf 131735 noxmtbuf 0 ipackets64 1386286 opackets64 1312131
rbytes64 13176264731 obytes64 12816667818 fcs_errors 0
macxmt_errors 16315 toolong_errors 0 macrcv_errors 0

idn1:
ipackets 189387 ierrors 0 opackets 136365 oerrors 0 collisions 0
rx_collisions 0 crc 0 buff 0 nolink 0 linkdown 0 inits 3
nocanput 54938 allocbfail 0 notbufs 0 reclaim 0 smraddr 0 txmax 0
txfull 0 xdcall 11788 sigsvr 453 mboxcrc 0 rbytes 1797429854
obytes 1226840176 multircv 0 multixmt 0 brdcstrcv 0 brdcstxmt 10
norcvbuf 54938 noxmtbuf 0 ipackets64 189387 opackets64 136364
rbytes64 1797429854 obytes64 1226840176 fcs_errors 0
macxmt_errors 0 toolong_errors 0 macrcv_errors 0
```

`kstat`(3K) Statistics

This section contains the `kstat`(3K) variables that pertain to the `netstat`(1M) command when it is executed against the IDN driver. Note that for `idn`*X* entries, there are separate instances of the variable reported for each network interface provided. (In this table, n/a means not applicable to IDN.)

The following table includes a list of the per-instance statistics that are available by using `netstat -k idn0` or `kstat -n idn0`.

TABLE 3-3 `kstat`(3K) Statistics Per Interface

Statistic	Description
`allocbfail`	Number of times the IDN driver failed to allocate a STREAMS buffer for incoming message
`brdcstrcv`	Total number of broadcast packets received by the interface
`brdcstxmt`	Total number of broadcast packets transmitted by the interface
`buff`	Number of times incoming data packet size exceeded the expected size of an SMR I/O buffer
`collisions`	n/a (transmit collisions); always zero (0)
`crc`	Number of times a corrupted data (header)) buffer was encountered during reclamation or was received from a remote domain
`fcs_errors`	Number of received packets that failed the CRC check for the IDN packet header
`ierrors`	Total number of input errors (For example, it was unable to allocate STREAMS buffer. The mailbox was corrupted, or the specified buffers were invalid.)
`inits`	Number of times the init function of the IDN driver was called
`ipackets`	Number of packets received by the IDN driver for the respective channel (network interface)
`ipackets64`	64-bit counter of the total number of packets received by the interface
`linkdown`	Number of times that an existing IDN connection to a specified domain was found not connected
`macrcv_errors`	Number of packets received that had a destination address that was different than the address of the receiving interface
`macxmt_errors`	Number of times the interface failed to transmit a packet due to internal IDN transmit errors (for example, a broken connection)
`mboxcrc`	Number of times the domain encountered a sending or receiving mailbox with a corrupted mailbox header
`multircv`	Total number of multicast packets received by the interface
`multixmt`	Total number of multcast packets transmitted by the interface
`nocanput`	Number of times the IDN driver encountered a full STREAMS queue when attempting to push data up the protocol stack
`nolink`	Number of times that a specified destination domain did not have a connection established with the local domain

TABLE 3-3 `kstat`(3K) Statistics Per Interface *(Continued)*

Statistic	Description
`norcvbuf`	Number of times a buffer could not be allocated to receive an incoming packet
`notbufs`	Number of times the domain failed to allocated an SMR I/O buffer for an outgoing messages
`noxmtbuf`	Number of times a transmit buffer could not be allocated to transmit an outgoing packet
`obytes`	Total number of bytes transmitted by the interface
`obytes64`	64-bit counter of the total number of bytes transmitted by the interface
`oerrors`	Total number of output errors (For example, the sending mailbox was corrupted. It was unable to allocate an SMR I/O buffer, or the header of the data packet was corrupted.)
`opackets`	Number of packets transmitted by the IDN driver on the respective channel
`opackets64`	64-bit counter of the total number of packets transmitted by the interface
`rbytes`	Total number of bytes received by the interface
`rbytes64`	64-bit counter of the total number of bytes received by the interface
`reclaim`	Number of times the domain attempted to reclaim an outgoing buffer, but found an error in the buffer (For example, the header was corrupted, or a bad SMR offset was encountered.)
`rx_collisions`	n/a (receive collisions); always zero (0)
`sigsvr`	Number of times after receiving a cross-domain call that the domain had to signal the channel server to start reading its mailbox
`smraddr`	Number of times the domain encountered an SMR I/O buffer that specified an invalid offset into the SMR (This pertains specifically to incoming buffers found in the mailboxes of the receiving domain.)
`toolong_errors`	Number of packets received that were larger than the expected IDN MTU size
`txfull`	Number of attempted packet transmissions that occurred while the receiving mailbox was full
`txmax`	Number of attempted packet transmissions that occurred when the outstanding packet count exceeded the value of `idn_window_emax`
`xdcall`	Number of times the domain had to perform a cross-domain call to notify the receiver of the incoming packets

The following table includes a list of the global statistics that are available by using `netstat -k idn` or `kstat -n idn`.

TABLE 3-4 `kstat`(3K) Global Statistics

Statistic	Description
`buf_fail`	Number of times that the domain failed to allocate an SMR I/O buffer
`buf_fail_last`	Time stamp of `lbolt` at the most recent time that an SMR buffer allocation failed
`curtime`	Snapshot of `lbolt` at the time the kstats were gathered to use as a reference for other time stamps saved in the global kstats
`dropped_intrs`	Total number of dropped cross-domain calls (DMV interrupts) by the domain due to either an unknown message (protocol) type or an inappropriate IDN version
`link_last`	Time-stamp of `lbolt` at the most recent time that a link, or connect, request occurred
`links`	Number of connect operations the domain participated in (Each domain connection counts as one link.)
`reap_count`	Total number of slabs the domain was able to successfully reap on behalf of a reap request from the master domain (the count is cumulative over the life of the domain)
`reap_last`	Time-stamp of `lbolt` at the most recent time that a reap occurred
`reaps`	Number of times the domain was requested to reap some SMR slabs by the master domain
`reconfig_last`	Time stamp of `lbolt` at the most recent time a reconfiguration took place
`reconfigs`	Number of times the domain participated in a reconfiguration
`slab_fail`	Number of times that the domain failed to allocate an SMR slab from the master domain
`slab_fail_last`	Time-stamp of `lbolt` at the most recent time that an SMR slab allocation failed
`unlink_last`	Time-stamp of `lbolt` at the most recent time that a disconnect request occurred
`unlinks`	Number of disconnect operations the domain participated in (Each domain disconnect counts as one unlink.)

Index

E

F

H

I

L

M

N

P

Q