Sun Enterprise 10000 IDN Configuration Guide

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

Part No. 806-5230-10
January 2001, Revision A

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

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 configuration of the Sun Enterprise™ 10000 InterDomain Network (IDN) features. For information about how to use these features, refer to the appropriate user guide listed in “Related Documentation” on page xi.

Before You Read This Book

This guide is intended for the InterDomain Networks 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 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 InterDomain Networks User Guide*	806-4131
	TCP/IP and Data Communications Administration Guide	805-4003
Reference	*Sun Enterprise 10000 SSP 3.4 Reference Manual*	806-4871
	Sun Enterprise 10000 IDN Error Messages	806-5231

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.

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://www.fatbrain.com/documentation/sun`

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 (806-5230-10) 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 IDN Configuration Guide*. For information on how to set up and use InterDomain Networks, refer to the *Sun Enterprise 10000 InterDomain Networks User Guide* in the SSP 3.4 AnswerBook2 collection.

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

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 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 2-1 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 (that is, unclaimed) 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 2-1 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 11. 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 12.

▼ 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 17 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 2-2 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 2-2 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 2-2 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 software interrupt, at which cross-domain interrupts are processed. The value is given as a number.

TABLE 2-2 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 2-2 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 2-3 `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 2-3 `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 2-4 `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 (that is, reclaim) 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

O

P

R

S

T