Sun Enterprise 10000 InterDomain Networks User Guide

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300Fax 650 969-9131

Part No.: 805-3534-10
November 1999, Revision A

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

Contents

Figures

Tables

Preface

This guide describes the InterDomain Network (IDN) feature, which enables dynamic system domains on an Sun Enterprise™ 10000 server to communicate with each other through the use of an internal, high-speed, hardware connection, as if they are communicating over a standard network.

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 document contains the following chapters:

Chapter 1 introduces IDN networks and explains their purpose.

Chapter 2 “Configuring InterDomain Networks” describes how to configure the IDN enviroment for better performance and reliability.

Chapter 3 describes how to set up and to use IDN networks.

Appendix A contains the IDN-related error messages, notifications, and panics that occur on the domain.

Appendix B contains the IDN-related error messages, notifications, and panics that occur on the system service processor (SSP).

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

The following table describes the typographic conventions used in this book.

TABLE P-1 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 User Guide*	806-1500-05
	Sun Enterprise Server Alternate Pathing User Guide	805-7985-10
	Sun Enterprise 10000 Dynamic Reconfiguration User Guide	805-5985-10
	TCP/IP and Data Communications Administration Guide	805-4003-10
Reference	*Sun Enterprise 10000 SSP 3.2 Reference Manual*	806-1501-05
	Sun Enterprise Server Alternate Pathing Reference Manual	805-5986-10
	Sun Enterprise 10000 Dynamic Reconfiguration Reference Manual	805-7986-10

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 of your document in the subject line of your email.

CHAPTER 1

Introduction to InterDomain Networks

This chapter contains an overview of IDN and information about Domain IP addresses, dynamic reconfiguration (DR), memory error handling, network-wide arbstops, and system commands and daemons.

Overview of IDN

The InterDomain Network (IDN) feature supports high-speed networking between dynamic system domains (or simply, *domains*) within a single Sun Enterprise 10000 platform. The IDN driver is a DLPI exporting driver that allows domains to communicate with each other using standard networking interfaces, such as Transmission Control Protocol/Internet Protocol (TCP/IP). However, an IDN requires no cabling or special hardware.

InterDomain networks take advantage of the Sun Enterprise 10000 hardware features that enable any set of resident domains to communicate among themselves over the system backplane using shared memory. A shared memory region (SMR) is used as a conduit for network packets. The SMR is maintained in one domain in the IDN and is used by all other domains in that IDN.

There may be multiple, independent IDNs within a single Sun Enterprise 10000 platform. Each network can comprise multiple logical network interfaces or channels, with each channel representing a separate IP subnet. Configure the number of networks, and the domains that make up a particular network, based on the performance considerations of your applications. For example, consider which domains require high-speed connectivity and also have sufficient processing power to effectively take advantage of InterDomain networking.

InterDomain networks can be used for many purposes. For example, IDNs can be used for the following reasons:

- Batch data transfers
- Consolidation of domains

Linking Domains

To link domains to an IDN or to create an IDN, use the `domain_link`(1M) command. The order in which you specify the domain names is not significant. For instructions on how to use the `domain_link`(1M) command, see "To Use the domain_link(1M) Command With Inactive Domains" on page 33.

Whenever an argument to `domain_link`(1M) specifies a domain that is already part of an IDN, all other domains in that IDN are also linked by the `domain_link`(1M) command.

Note that when you link domains together in an IDN, each domain can communicate directly with the other domains in the network by using the shared memory region (SMR). There is no priority given to the domains based on the order in which they were added to an IDN.

Master Domain

Only one domain in an IDN is denoted as the master domain. The master domain maintains the SMR, which is used as a conduit for network traffic. For example, if *domain_a* is the master domain, *domain_b* and *domain_c* communicate with each other using the SMR maintained on *domain_a*.

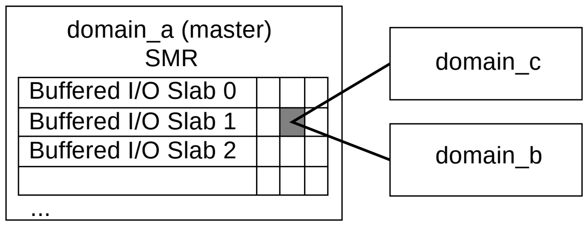

FIGURE 1-1 Domain Communication Using the SMR in the Master Domain

When you create a new IDN out of two domains that do not belong to an existing IDN, the master domain is automatically chosen by the system. After this decision is made, the master domain cannot be changed unless you unlink the master domain or unless the master domain hangs and the network is automatically reconfigured to use an alternate master. An exception to this rule occurs when two existing IDNs are merged by using a single `domain_link`(1M) command. In this case, the system determines which domain from among the two current master domains will become the master domain for the new IDN.

The system chooses the master domain by determining which domain has the greatest processing power and the widest memory bandwidth, which is a function of how many system boards with memory are contained within a domain. The domain with the greatest overall capacity is used as the master domain because it has the responsibility of servicing IDN buffer requests on behalf of other domains.

Unlinking Domains

To unlink a domain from an IDN, use the `domain_unlink`(1M) command, which accepts one or more domains as a parameter. When you unlink a domain, the system broadcasts a message to the remaining domains in the IDN to inform them that they should no longer attempt to communicate with the outgoing domain. Other domains in the network continue to communicate with each other without interruption, both during and after the unlink operation.

Although there is no particular order in which you must deconfigure an IDN link and its associated network interface(s), Sun suggests that you deconfigure the network interface by using the `ifconfig`(1M) command before you unlink the domain to prevent users from unnecessarily using the disconnected link.

By default, the system will not perform an unlink operation on an active domain if *any* domain within the same IDN is in an unknown (AWOL) state, such as halted or hung. The state of the domain is detected and reported when you perform the unlink operation.

Force Options

You can use one of two force options, `-f` or `-F`, to bypass the check for domains in an unknown state and to force the unlink operation to proceed. With the soft force option, `-f`, the `domain_unlink`(1M) command attempts to unlink all of the specified domains in the standard manner; however, if a time-out condition occurs due to the presence of an AWOL domain within the IDN, the `domain_unlink`(1M) command uses the `-F` option to remove the link, forcing the domain to be unlinked.

With the hard force option, `-F`, the `domain_unlink`(1M) command disconnects the specified domain from all of the other domains in the IDN and does so without synchronizing the disconnections. Use this option only when the specified domain is completely nonresponsive (that is, not responding to log in requests) or when it must be isolated from the IDN as part of AWOL recovery.

Caution – The force option must be used only as a recovery mechanism when a domain is known to be in an unknown state (AWOL). It must not be used under standard conditions. It could result in an arbstop if the hardware is reprogrammed while the IDN is active.

Dismantling an IDN

You can dismantle an entire IDN in a single operation, which isolates each domain that is a member of the IDN. Execute the `domain_unlink`(1M) command with at least *n*-1 names of the domains in the IDN, where *n* is the total number of domains within the IDN.

Automated IDN Handling

The IDN subsystem, in conjunction with support from the SSP, can automatically link and unlink domains. Automatic linking occurs at boot time if the domain has been configured as part of the IDN. Automatic unlinking occurs when one or more IDN members detect and report that another IDN member is not responding to IDN requests. If the master domain is nonresponsive, a new master domain wil be elected from the available domains after the master is unlinked. Although the domain is automatically unlinked, the `domain_status`(1M) command still reports the domain as being linked.

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.

Dynamic Reconfiguration and IDNs

DR operations work on individual domains within an IDN. The IDN traffic to and/or from the target domain is paused for only a brief period of time while DR operations are executed on the domain.

Attach Operation

When you attach a board to a domain that is part of an IDN, the following sequence of actions occur:

1. You perform the Init Attach operation.

2. You perform the Complete Attach operation, at which point DR unlinks the domain in which the board resides from the IDN. DR saves the IDN configuration information internally so that DR can relink the domain after the Complete Attach operation.
3. DR then performs the Complete Attach operation.
4. After the Complete Attach operation completes successfully, DR relinks the domain to the IDN.

Detach Operation

During a Detach operation, the following sequence of actions occur:

1. You perform the Drain operation.
2. After the Drain operation has completed and you have selected the Complete Detach operation, DR unlinks the domain from the IDN. DR saves the IDN configuration information internally so that it can automatically relink the domain after the Drain operation.
3. DR then performs the Complete Detach.
4. After the Complete Detach operation, DR relinks the domain to the IDN.

Note – The DR Complete Attach and Complete Detach operations must finish in a timely manner to prevent TCP/IP connections across the IDN from timing out. Typically, the timeout value is two minutes.

If the unlink operation that is performed by the DR subsystem fails for some reason (for example, an AWOL domain exists within the IDN), you must resolve the problem manually (that is, you must manually unlink the AWOL domain) before you attempt the DR operation again.

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.

Network-Wide Arbstops

An arbitration stop, or arbstop, of the domain causes the domain to freeze and all hardware-level transactions to cease. When an arbstop occurs within a domain that is part of an IDN, subsequent arbstops occur in all of the other domains in that same IDN.

Note – Domains that are not members of an arbstopped IDN are *not* affected by the arbstop.

Normally, this is not a problem because arbstops rarely occur. However, if another domain in that IDN is in an unknown state and possibly attempting to communicate with the domain being unlinked, the unlink command can cause arbstops to occur, especially when it is used with the force option.

If a domain or cluster arbstop occurs, the current BBSRAM and arbstop information is dumped to the following files:

- Domain arbstop:

  `$SSPLOGGER/`*domain_name*`/Edd-Arbstop-bbsram-`*time_stamp*

  `$SSPLOGGER/`*domain_name*`/Edd-Arbstop-Dump-`*time_stamp*

- Cluster arbstop:

  `$SSPLOGGER/`*domain_name*`/Edd-MD-Arbstop-bbsram-`*time_stamp*

  `$SSPLOGGER/`*domain_name*`/Edd-MD-Arbstop-Dump-`*time_stamp*

To understand the conditions under which arbstops can occur, consider the hardware architecture that allows system boards to communicate with each other. In the following illustration, three domains exist, none of which are members of an IDN.

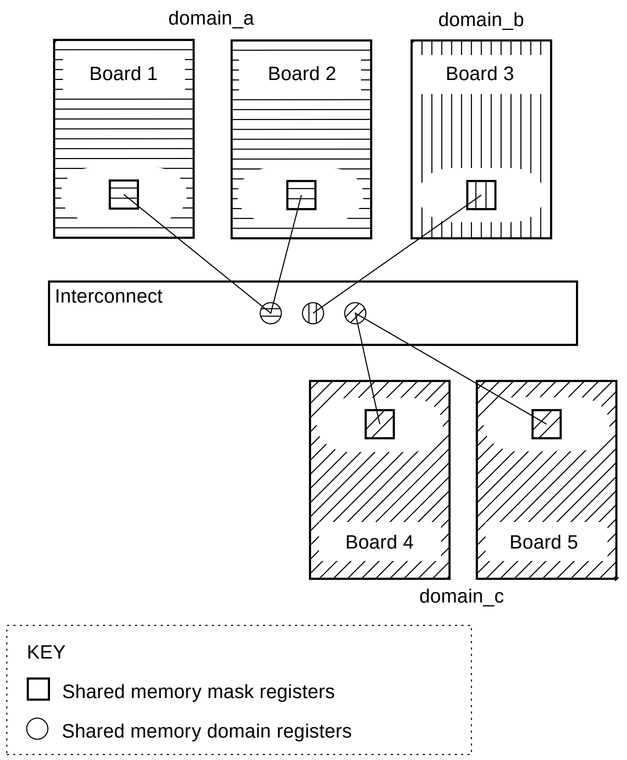

FIGURE 1-2 Three Isolated Domains

The interconnect (that is, the backplane of the system) contains shared memory domain registers, and each board contains shared memory mask registers. These registers work to support interboard communication within a domain, and they facilitate interdomain communication in the IDN environment. The shared memory domain registers on the interconnect allow a message to be forwarded to a particular destination board only if the registers are programmed to allow the originating board to send a message to the specified destination board. Correspondingly, the shared memory mask registers on a destination board allow an incoming message to be accepted only if the registers are programmed to allow that destination board to receive a message from the originating board.

In FIGURE 1-3, the shared memory domain registers on the interconnect for the three domains have been grouped to allow forwarding of messages between the domains. In addition, the boards in each domain now have shared memory mask registers programmed to accept messages from the other domains.

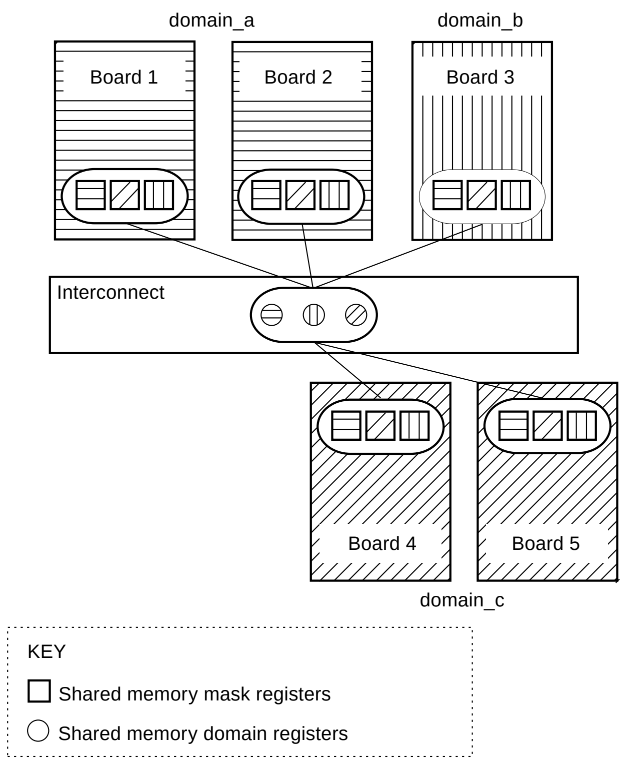

FIGURE 1-3 IDN With Three Domains

In addtion to certain hardware failures, an arbstop can occur if any board attempts to send a message to another board and if *either* the shared memory domain registers, or the shared memory mask registers, do not allow communication between the two boards. During the linking or unlinking process, the `domain_link`(1M) and `domain_unlink`(1M) commands reprogram these registers to enable or disable cross-domain transactions.

If one domain is in an unknown state (for example, partially hung) and if you attempt to unlink another domain in the IDN, the `domain_unlink`(1M) command will fail unless you use one of the force options, `-f` or `-F`, because the `domain_unlink`(1M) command needs to communicate to the hung domain to ensure that all interdomain transactions have ceased before it reprograms the registers that are related to the shared memory. The `domain_unlink`(1M) command does not reprogram the shared memory domain registers if the domain is not responding. If you force the unlink to proceed, the shared memory domain registers are reprogrammed; however, the shared memory mask registers on the hung domain are not reprogrammed, so the IDN software on that domain is not aware that the disconnect has taken place.

After all of the IDN-specific registers have been reprogrammed, if the hung domain attempts to communicate with another domain, the hung domain, and any other domains in that IDN can arbstop. Thus, use the force option only if you are certain that the hung domain will not attempt any further communication with the other domains in the same network. To reduce the potential for such an arbstop, either reboot the hung domain or unlink it from the IDN before you unlink any other domain.

Caution – Do not use the force option on a known active domain unless it is absolutely necessary.

Commands

Interdomain networks affect the behavior of several commands. This section contains an explanation of the behavior of the commands that are affected by IDNs.

SSP Commands

The following table contains a list of the SSP commands affected by IDNs.

TABLE 1-1 SSP Commands Affected by IDNs

Command	Affect
`bringup`(1M)	You must unlink any domain that is in an unknown state (AWOL) before you use the `bringup`(1M) command to reboot any other domain in the IDN. Note that if multiple domains within the same IDN are hung, you must unlink all of the hung domains simultaneously. In addition, you cannot unlink any nonresponsive domain when other nonresponsive domains are present in the same IDN. Finally, the `bringup`(1M), `domain_link`(1M), and `domain_unlink`(1M) commands cannot run concurrently.

TABLE 1-1 SSP Commands Affected by IDNs

Command	Affect
`domain_create`(1M)	You must unlink any domain that is in an unknown state (AWOL) before you use the `bringup`(1M) command to reboot any other domain in the IDN. Note that if multiple domains within the same IDN are hung, you must unlink all of the hung domains simultaneously. In addition, you cannot unlink any nonresponsive domain when other nonresponsive domains are present in the same IDN. Finally, the `bringup`(1M), `domain_link`(1M), and `domain_unlink`(1M) commands cannot run concurrently.
`domain_remove`(1M)	You cannot remove a domain that is currently a member of an IDN. The domain must first be unlinked; then, you can remove it.
`dr`(1M)	DR commands and IDN commands cannot run concurrently. See "Dynamic Reconfiguration and IDNs" on page 5 for more information about the `dr`(1M) command.
`edd`(1M)	When `edd`(1M) produces a dump file as a result of an arbstop or recordstop in a domain that is part of an IDN, the dump file is based on the entire set of boards comprising *all* of the domains that are members of the IDN. The `edd`(1M) daemon is also required to enable automatic linking of domains and AWOL recovery.
`hostint`(1M)	By default, you cannot issue a `hostint`(1M) operation to a domain that is a member of an IDN. You must unlink the domain first, unless you use the force option with the `hostint`(1M) command.
`hpost`(1M)	When `hpost -Wc` is used on a domain that is part of an IDN, it clears recordstops on all of the boards within all of the domains in that IDN.
`power`(1M)	You cannot power off a system board within a domain that is a member of an IDN unless you use the force option with the `power`(1M) command (refer to the `power`(1M) man page for more information on the use of the force option). The domain must first be unlinked from the IDN. Then, you can use the `power`(1M) command to power off the board.
`sigbcmd`(1M)	By default, you cannot issue a `sigbcmd` *obp* or *panic* operation to a domain that is currently a member of an IDN. You must first unlink the domain unless you use the force option with the `sigbcmd`(1M) command.

System Commands

There is only one system command affected by IDNs.

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.

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.

See "Domain IP Addresses" on page 4 for more information about the contents of the `/etc/hostname.idn`*X* file. 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 specified in the `domain_config` file 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.

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.**

```
<#Ø> ok cd /memory
<#Ø> 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 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.

TABLE 2-1 `ndd`(1M) Parameters *(Continued)*

Name	Min.	Max.	Default	Description
`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.
`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.

TABLE 2-1 ndd(1M) Parameters *(Continued)*

Name	Min.	Max.	Default	Description
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 18. 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 19.

▼ 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. See `modinfo`(1M) 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 26 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;
```

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	30	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). The domain must not be linked to an IDN when the value is changed.
`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.

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

Name	Min.	Max.	Default	Description
`idn_hiwat`	1024	536870912	262144	Controls the high-water mark of the IDN STREAM queue. This value is given in bytes. The TCP/IP stack must not be plumbed on top of the driver when the value is changed.
`idn_lowat`	1	524288	1024	Controls the low-water mark of the IDN STREAM queue. This value is given in bytes. The TCP/IP stack must not be plumbed on top of the driver when the value is changed.
`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. The domain must not be linked to an IDN, and the TCP/IP stack must not be plumbed on top of the driver, when the value is changed. This parameter requires entries to be placed in the idn.conf file.
`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. The domain must not be linked to an IDN, and the TCP/IP stack must not be plumbed on top of the driver when the value is changed.
`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_msgwait_cfg`	1	40	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`	1	300	30	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.

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

Name	Min.	Max.	Default	Description
`idn_msgwait_con`	1	300	30	Controls the minimum amount of time to wait for a response to a CON (connection) message. The value is given in seconds.
`idn_msgwait_data`	1	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`	1	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`	1	300	30	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	16	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. It can be changed at any time.
`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	50	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). It can be changed at any time.
`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. It can be changed at any time.

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

Name	Min.	Max.	Default	Description
`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. The domain must not be linked to an IDN, and the TCP/IP stack must not be plumbed on the driver when the value is changed.
`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.
`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. It can be changed at any time.
`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. It can be changed at any time.
`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.

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

Name	Min.	Max.	Default	Description
`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. The domain must not be linked to an IDN.
`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. It can be changed at any time.
`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. It can be changed at any time.
`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. The domain must not be linked to an IDN, and the TCP/IP stack must not be plumbed on the driver when this value is changed.
`idn_smr_size`	0	Entire SMR	0	Used with a non-zero value to override the value indicated by the idn-smr-size OBP parameter. The chosen value cannot exceed the value specified by OBP. A value of zero (0) causes the domain to use the value given by OBP. 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. It can be changed at any time.
`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. It can be changed at any time.

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) command line utility.

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 14556939 reconfigs 1 reconfig_last 839329 reaps 0
reap_last 0
links 2 link_last 840383 unlinks 2 unlink_last 843127 buf_fail 0
buf_fail_last 0 slab_fail 0 slab_fail_last 0 reap_count 0
dropped_intrs 0

# netstat -k idn0
idn0:
ipackets 0 ierrors 0 opackets 0 oerrors 0 collisions 0
rx_collisions 0 crc 0 buff 0 nolink 0 linkdown 0 inits 0 nocanput 0
allocbfail 0 notbufs 0 reclaim 0 smraddr 0 txmax 0 txful 0 xdcall 0
sigsvr 0 mboxcrc 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 interfaces for two fictitious domains. The amounts in the examples are for informational purposes only; the output you receive can differ significantly.

```
# netstat -k idn0 idn1

idn0:
ipackets 3986730 ierrors 0 opackets 3035326 oerrors 24 collisions 0
rx_collisions 0 crc 0 buff 0 nolink 0 linkdown 24 inits 1 nocanput 8
allocbfail 0 notbufs33 reclaim 0 smraddr 0 txmax 0 txfull 0
xdcall 12369 sigsvr 4461 mboxcrc 0

idn1:
ipackets 1973118 ierrors 0 opackets 1571003 oerrors 7 collisions 0
rx_collisions 0 crc 0 buff 0 nolink 0 linkdown 7 inits 1 nocanput 14
allocbfail 0 notbufs 0 reclaim 0 smraddr 0 txmax 0 txfull 0
xdcall 1658 sigsvr 10818 mboxcrc 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.

TABLE 2-3 `kstat`(3K) Statistics Per Interface (`netstat -k idn0`)

Statistic	Description
`ipackets`	Number of packets received by the IDN driver for the respective channel (network interface)
`opackets`	Number of packets transmitted by the IDN driver on the respective channel
`txmax`	Number of attempted packet transmissions that occurred when the outstanding packet count exceeded the value of `idn_window_emax`
`txfull`	Number of attempted packet transmissions that occurred while the receiving mailbox was full
`xdcall`	Number of times the domain had to perform a cross-domain call
`sigsvr`	Number of times after receiving a cross-domain call that the domain had to signal the channel server to start reading its mailbox

TABLE 2-3 `kstat`(3K) Statistics Per Interface (`netstat -k idn0`) *(Continued)*

Statistic	Description
`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.)
`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.)
`txcoll`	n/a (transmit collisions)
`rxcoll`	n/a (receive collisions)
`crc`	Number of times a corrupted data (header)) buffer was encountered during reclamation or was received from a remote domain
`buff`	Number of times incoming data packet size exceeded the expected size of an SMR I/O buffer
`nolink`	Number of times that a specified destination domain did not have a connection established with the local domain.
`linkdown`	Number of times that an existing IDN connection to a specified domain was found not connected
`inits`	Number of times the init function of the IDN driver was called
`nocanput`	Number of times the IDN driver encountered a full STREAMS queue when attempting to push data up the protocol stack
`allocbfail`	Number of times the IDN driver failed to allocate a STREAMS buffer for incoming message
`notbufs`	Number of times the domain failed to allocated an SMR I/O buffer for an outgoing messages
`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.)
`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.)
`mboxcrc`	Number of times the domain encountered a sending or receiving mailbox with a corrupted mailbox header
`txmax`	Number of times that a domain-to-domain connection exceeded `idn_window_max`

The following table includes a list of the global statistics.

TABLE 2-4 `kstat(3K)`Global Statistics (`netstat -k idn`)

Statistic	Description
`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
`reconfigs`	Number of times the domain participated in a reconfiguration
`reconfig_last`	Time stamp of `lbolt` at the most recent time a reconfiguration took place
`reaps`	Number of times the domain was requested to reap some SMR slabs by the master domain
`reap_last`	Time-stamp of `lbolt` at the most recent time that a reap occurred
`links`	Number of connect operations the domain participated in (Each domain connection counts as one link.)
`link_last`	Time-stamp of `lbolt` at the most recent time that a link, or connect, request occurred
`unlinks`	Number of disconnect operations the domain participated in (Each domain disconnect counts as one unlink.)
`unlink_last`	Time-stamp of `lbolt` at the most recent time that a disconnect request occurred
`buffail`	Number of times that the domain failed to allocate an SMR I/O buffer
`buffail_last`	Time stamp of `lbolt` at the most recent time that an SMR buffer allocation failed
`slabfail`	Number of times that the domain failed to allocate an SMR slab from the master domain
`slabfail_last`	Time-stamp of `lbolt` at the most recent time that an SMR slab allocation failed
`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)
`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

CHAPTER 3

Using Inter-Domain Networks

This chapter contains instructions on how to use the IDN commands. Make sure that you read Chapter 1 "Introduction to InterDomain Networks" and Chapter 2 "Configuring InterDomain Networks" before you attempt to use the commands in this chapter.

IDN Requirements

This section contains the general and OpenBoot PROM (OBP) requirements for IDNs.

Domain and SSP

Before you can use IDN commands, your system must have, at the minimum, the following software components:

- The host must have a version of the Solaris operating environment for the Sun Enterprise 10000 server, which contains the IDN driver packages (`SUNWidn.u`, the 32-bit binaries, and `SUNWidnx.u`, the 64-bit binaries).
- The SSP must have SSP 3.2 with all of the current patches.

OpenBoot PROM Variable

OBP has one variable, the shared memory region (SMR) size, `idn-smr-size` that must be set before you link any domains. A value of zero disables the IDN feature. A nonzero value indicates the number of megabytes of kernel space to reserve for the SMR.

To set this variable, boot, or halt, the system to the OBP prompt and set the variable by using the `setenv` command, as in the following example:

```
<#ø> ok setenv idn-smr-size 32
```

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 affect. You can, however, decrease the actual size of the SMR by using the `idn_smr_size` `driver.conf`(4) variable. For more information about the default and suggested sizes for the `idn-smr-size` variable, see “OpenBoot PROM Variable” on page 14.

Using IDN Commands

The following commands support IDN:

- `domain_link`(1M) – Links domains to form or expand an IDN
- `domain_unlink`(1M) – Unlinks one or more domains from an IDN
- `domain_status`(1M) – Displays information about the domains that make up all of the IDNs on the server

---

Note – You must be user `ssp` to run the `domain_link`(1M), `domain_unlink`(1M), and `domain_status`(1M) commands. Refer to the man pages for these commands in the *Sun Enterprise 10000 SSP 3.2 Reference Manual*.

Viewing IDN Status

The domain_status(1M) command returns a listing that provides general information about domains, as well as the list of any IDNs that contain those domains. Here is an example:

```
ssp% domain_status
DOMAIN         TYPE                     PLATFORM  OS     SYSBDS
xf3            Ultra-Enterprise-10000     xf3     5.7    4 6 7
xf3-b8         Ultra-Enterprise-10000     xf3     5.7    8 9 13
xf3-b10-hme0   Ultra-Enterprise-10000     xf3     5.6    10 11
xf3-b2         Ultra-Enterprise-10000     xf3     5.8    2 14
xf3-b5-fddi0   Ultra-Enterprise-10000     xf3     5.7    0 1 5

IDN NETWORKS
0: xf3-b2 xf3-b8
1: xf3 xf3-b5-fddi0
```

The section at the bottom of this listing indicates that two IDNs exist on this server. Each IDN is identified by a number followed by the names of the domains that make up that network. Note that the number associated with the IDN is simply a tag used in the listing; it is not a persistent identifier for that IDN.

Using the domain_link(1M) Command

This section contains procedures for linking domains to create an IDN. The method for creating an IDN depends on the state of the domains that you want to link together. You can link inactive domains or active domains. For more information about the domain_link(1M) command, see “Linking Domains” on page 2.

▼ To Use the domain_link(1M) Command With Inactive Domains

The following procedure contains steps for linking two domains, domain_a and domain_b. If you are linking more than two domains, you must perform the domain-specific steps (that is, those executed at the domain prompt) for all of the domains.

None of the domains that are to be part of an IDN need be up and operational prior to defining an IDN; however, if the domains are not booted, the link operation only updates only the logical IDN information maintained by the SSP. When the domain is brought up by using the `bringup`(1M) command, the information on the SSP about the IDN is used to configure the domain.

Note – Before you perform the steps in this procedure, you must ensure that each domain has an `/etc/hostname.idn`*X* file defined. For more information about this file, refer to "Automatic Activation of the Logical Network Interfaces" on page 13. If this file is not already defined, you must create it for each domain before you proceed with the remaining steps in this section.

---

1. **Use the `domain_switch`(1M) command to ensure that the `SUNW_HOSTNAME` variable is set to the correct domain name.**

   The domain must be running a version of the Solaris operating environment that supports IDNs. Refer to the IDN release notes for version support information. The `domain_link`(1M) command will not succeed if the `SUNW_HOSTNAME` variable is set to a doman that is running the Solaris 2.5.1 or Solaris 2.6 operating environment.

2. **On the SSP, execute the `domain_link`(1M) command to define an IDN.**

```
ssp% domain_link domain_a  domain_b
```

Note – Because domains can be linked when they are not booted, you cannot verify that a given domain supports IDN. If the domain does not support IDN, then upon boot, the domain will not be automatically linked.

---

3. **Bring up the domains to the OpenBoot PROM (OBP) prompt.**

4. **At the OBP prompt, ensure that the IDN driver is enabled.**

   The `idn-smr-size` variable must be set to a valid nonzero value to enable the IDN driver.

```
<#Ø> ok printenv
```

   If the `idn-smr-size` variable is not set properly, see "OpenBoot PROM Variable" on page 31 for instructions on how to set this variable.

5. **Execute the `bringup`(1M) command for each domain.**

6. **Boot the domains.**

   After all domains are booted, the IDN between them is automatically enabled by using the SSP services that detect the booted domains.

▼ To Use the domain_link(1M) Command With Active Domains for TCP/IP

The following procedure contains steps for linking two domains, domain_a and domain_b. If you are linking more than two domains, you must perform the domain-specific steps (that is, those executed at the domain prompt) for all of the domains. In the following procedure, both domains are booted.

1. **Ensure that each domain has an /etc/hostname.idn*X* file defined.**

   For more information about this file, refer to "Automatic Activation of the Logical Network Interfaces" on page 13. If this file is not already defined, you must create it for each domain before you proceed with the remaining steps in this section.

2. **Use the eeprom(1M) command to ensure that the IDN driver is enabled.**

   Note – The OBP variable idn-smr-size must be set prior to boot so that the operating environments will reserve the appropriate amount of memory for the SMR. By default, idn_nwr_size is equal to idn-smr-size, so typically, idn-smr-size must be set to an equivalent value for all of the domains in the IDN.

   The idn-smr-size variable must be set to a valid non-zero value to enable the IDN driver.

   ```
   <#0> ok printenv
   ```

   If the idn-smr-size variable is not set properly, see "OpenBoot PROM Variable" on page 31 for instructions on how to set this variable.

3. **Use the domain_link(1M) command to link the domains.**

   ```
   ssp% domain_link domain_a domain_b
   ```

▼ To Create a Basic IDN

The following procedure contains steps to set up a very basic TCP/IP network. Your configuration can vary; therefore, the examples in the steps may not work for your configuration. Refer to the *TCP/IP and Data Communications Administration Guide* for more specific information on how to set up a TCP/IP network.

1. **Use the** `domain_switch`**(1M) command to ensure that the** `SUNW_HOSTNAME` **variable is set to the correct domain name.**

   The domain must be running a version of the Solaris operating environment that supports IDNs. Refer to the IDN release notes for version support information. The `domain_link`(1M) command will not succeed if the `SUNW_HOSTNAME` variable is set to a doman that is running Solaris 2.5.1 or Solaris 2.6.

2. **Use the** `eeprom`**(1M) command to ensure that the IDN driver is enabled in each domain.**

   The `idn-smr-size` variable must be set to a valid nonzero value to enable the IDN driver. If the `idn-smr-size` variable is not set properly, see "OpenBoot PROM Variable" on page 31 for instructions on how to set this variable.

3. **Plumb and configure the network interfaces within each domain.**

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

   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 (see "Domain IP Addresses" on page 4 and `hosts`(4) for more information).

4. **Use the** `domain_link`**(1M) command to link the domains.**

```
ssp% domain_link domain_a domain_b
```

▼ To Merge IDNs

- **Use the names of two domains in separate IDNs with the `domain_link`(1M) command.**

```
ssp% domain_link domain_a domain_b
```

This command merges the IDN that contains *domain_a* with the IDN that contains *domain_b*. A master domain is chosen for the new IDN from among the domains in both of the existing IDNs.

Using the `domain_unlink`(1M) Command

This section contains instructions for unlinking domains from an IDN. The method of unlinking an IDN depends of the state of the domains that you want to unlink and the state of the other domains in the IDN. For more information about the `domain_unlink`(1M) command, see "Unlinking Domains" on page 3.

▼ To Unlink a Domain From an IDN

1. **Use the `domain_status`(1M) command to check the status of all of the domains in the IDN.**
2. **If desired, dismantle the TCP/IP stack by using the `ifconfig`(1M) command in the same manner as other network interfaces.**

```
# ifconfig idn0 down
# ifconfig idn0 unplumb
```

3. **On the SSP, execute the `domain_unlink`(1M) command to disconnect IDN connections to the domain.**

```
ssp% domain_unlink domain_name
```

If the IDN contains domains that are in an unknown (AWOL) state (halted or hung), you must unlink all of the AWOL domains simultaneously, or use one of the force options on the given domain. For example, if *domain_a* and *domain_c* are in unknown states, you should unlink them simultaneously with the following command:

```
ssp% domain_unlink domain_a  domain_c
```

If a domain is non-responsive, you can use the force option (`-f` or `-F`) to unlink the given domain.

```
ssp% domain_unlink -f domain_b
```

Caution – Try to unlink all AWOL domains first before you attempt to unlink a domain with the force option. For more information about forcing an unlink operation, see "Force Options" on page 3.

At this point, the domain is fully unlinked from the IDN.

You can dismantle the TCP/IP stack and unlink the IDN connection in any order. Unlinking a domain from an IDN does not necessarily require that the TCP/IP stack be dismantled. In the example above, `idn0` 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. Refer to the `hosts`(4) man page for more information about configuring TCP/IP networks.

Note – If you unlink the last pair of domains in an IDN, the IDN will no longer exist, so no information will appear in the `domain_status`(1M) output.

APPENDIX **A**

IDN Error Messages, Notifications, and Panics on the Domain

This chapter contains two tables:

- IDN error numbers that appear on the domain
- IDN error messages, notifications, and panics that occur on the domain

The destination of these messages depends entirely on the location of the individual error or failure. For some errors, both locations must be used to diagnose the error or failure.

Domain IDN Messages

This section contains the IDN messages that occur on the domain from which the IDN command was executed. TABLE A-1 contains the name, number, and description of IDN `errno` numbers. This table describes some of the typographical conventions that are used in the message tables.

Tables A-2 through A-5 contain the common InterDomain Network errors, notifications, and panics that can occur within a dynamic system domain (or *domain* in this publication). These tables contain the text of the error, description of the possible cause of the error, and suggested action. For notifications, the list contains the text of the notice and a possible cause of the message.

An IDN operation can take several minutes to complete successfully. You may need to wait several minutes before network statistics become available. In addition, other information about the IDN may not appear in the output of certain commands until the domain is successfully linked to the IDN.

IDN messages that occur on the domain are sent to the following locations:

- `netcon`(1M) console window

- `/var/adm/messages`
- `$SSPLOGGER/`*domain_name*`/messages`

The following table contains the IDN error numbers that are specific to the domain.

TABLE A-1 IDN Domain-Specific Error Numbers

Message	Number	Description
`IDNKERR_DRV_DISABLED`	`0x100`	The IDN driver is disabled.
`IDNKERR_DATA_LEN`	`0x101`	The IDN region in the signature block (BBSRAM) is misaligned between the IDN driver and the SSP (`P0` represents the length).
`IDNKERR_INFO_FAILED`	`0x102`	SSI_INFO command failed (only in engineering).
`IDNKERR_INVALID_DOMAIN`	`0x103`	An invalid domain ID was specified (`P0` represents the domain ID, and `P1` represents the CPU ID).
`IDNKERR_INVALID_FORCE`	`0x104`	An invalid force option was passed (`P0` represents the force option).
`IDNKERR_INVALID_CMD`	`0x105`	An invalid IDN command was requested (`P0` represents command).
`IDNKERR_INVALID_WTIME`	`0x106`	An invalid wait time was specified for the IDN operation (`P0` represents the wait time).
`IDNKERR_SMR_CORRUPTED`	`0x107`	SMR memory was found corrupted (`P0` represents the domain ID, against which the corruption was found).
`IDNKERR_CPU_CONFIG`	`0x108`	Domain ID is not configured properly for an IDN. Each system board that hosts memory must have at least one CPU (`P0` represents the domain ID).
`IDNKERR_HW_ERROR`	`0x109`	The domain was unable to properly program the hardware to support an IDN connection to domain ID (`P0` represents the domain ID)
`IDNKERR_SIGBINTR_LOCKED`	`0x10a`	The signature block interrupt lock on the host is currently locked.
`IDNKERR_SIGBINTR_BUSY`	`0x10b`	The signature block interrupt handler thread is currently active.
`IDNKERR_SIGBINTR_NOTRDY`	`0x10c`	The signature block interrupt handler thread has not been initialized.
`IDNKERR_CONFIG_FATAL`	`0x10d`	An error occurred during the exchange of configuration information with domain ID, specifically it was missing information (`P0` represents the domain ID).
`IDNKERR_CONFIG_MULTIPLE`	`0x10e`	Multiple conflicts were found between the configuration parameters exchanged during the connection establishment (`P0` represents the domain ID).

TABLE A-1 IDN Domain-Specific Error Numbers *(Continued)*

Message	Number	Description
IDNKERR_CONFIG_MTU	0x10f	The MTU sizes of the domains do not match (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_BUF	0x110	The values of the idn_smr_bufsize variable conflict among the domains (P0 represents the domain ID; P1 presents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_SLAB	0x111	The values of the SMR slab sizes conflict among the domains (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_NWR	0x112	The values of the idn_nwr_size variables conflict among the domains (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_NETS	0x113	The values of the idn_max_nets variables conflict between domains (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_MBOX	0x114	The values of the idn_mbox_per_nets variables conflict between domains (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_NMCADR	0x115	The number of MCADRs received does not match the number of MCADRs that the domain had reported to exist (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_MCADR	0x116	Received an MCADR for a board that the remote domain did not report to exist (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_CKSUM	0x117	The values of the idn_checksum parameter is not consistent among the domains (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).
IDNKERR_CONFIG_SMR	0x118	The master domain SMR is too large for the slave domain (P0 represents the domain ID; P1 represents the expected value, and P2 represents the actual value).

The following table contains the errors, notices, and panics that are specific to the domain.

TABLE A-2 IDN Domain-Specific Messages, 100 through 142

Error	Description	Recovery
`WARNING: IDN: 100: sigblock area misaligned (`*bytes*`) != exp (`*bytes*`)`	This message indicates a mismatch between the version of the sigblock data structure in the IDN driver and the region of the signature block that is reserved for the IDN.	Unlink the domains, then recreate the IDN.
`WARNING:IDN:101: not in expected OFFLINE state for DDI_RESUME`	The IDN driver is not in the expected state for the DR driver to perform the `DDI_RESUME` operation.	Unlink the domain, then relink it.
`NOTICE: IDN: 102 driver disabled - check OBP environment (idn-smr-size)`	The IDN driver was not initialized.	Check the OpenBoot PROM (OBP) variable `idn-smr-size` to ensure that it is set properly. See "ndd(1M) Driver Parameters" on page 16 for more information about this variable.
`WARNING: IDN: 103: unable to reference sigblock area`	The `sgnblk_poll_reference()` routine failed to initialize. The IDN driver may have been loaded too early in the boot sequence.	Reload the module after the operating system boots.
`WARNING: IDN: 104: cannot suspend while active (state = `*GSTATE*`)`	The IDN driver cannot be suspended while it is in use.	Wait for the driver to complete the current transmission, or unlink the domain from the IDN before you initiate a DR operation.
`WARNING: IDN: 105: driver parameter (`*parameter*`) specified (`*number*`) out of range [`*low_value* `-` *high_value*`]`	The value for the specified parameter is outside of the range of values that can be used for the IDN.	Reset the parameter with a value that is within the allowed range.
`WARNING: IDN: 106: idn_nwr_size (`*Mbytes*`) > idn_smr_size(`*Mbytes*`) - Limiting to` *number* `MB`	The value of `idn_nwr_size` variable is greater than the value of `idn_smr_size` variable. The value of `idn_nwr_size` cannot be larger than the value of `idn_smr_size`.	The IDN driver reduces the size of the network region (NWR) to the total size of the shared memory region (SMR). If the IDN must have a NWR that is larger than the current size of the SMR, increase the size of the SMR so that the size of the NWR can be increased to the appropriate value.

TABLE A-2 IDN Domain-Specific Messages, 100 through 142 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 107: memory region(`*bytes*`) < slab size(`*bytes*`)`	The value of the `idn_nwr_size` variable is less than the size of one of the slabs within the SMR.	Increase the value of `idn_smr_size` or `idn_nwr_size` to a value that is larger than the smallest buffer size in the SMR. Or, reset other tunables, such as `idn_slab_bufcount`, until the size of each slab within the IDN is smaller than the value of `idn_nwr_size`.
`WARNING: IDN: 108: idn_lowat(`*bytes*`) >= idn_hiwat(`*bytes*`)`	The specified values for the low-water and high-water marks for IDN STREAMS are not set properly.	Lower the value of `idn_lowat` or increase the value of `idn_hiwat` as appropriate. See the "ndd(1M) Driver Parameters" on page 16 for more information about the appropriate values for these parameters.
`WARNING: IDN: 109: mailbox area(`*bytes*`) + slab size(`*bytes*`) > nwr region(`*bytes*`)`	The specified values for the mailbox area, `idn_mbox_size`, and the slab size, `idn_slab_size`, variables are not set properly.	Increase the size of the NWR or the SMR so that the size of the NWR can be increased. You can also decrease the size of other variables, such as the number of mailboxes per channel, `idn_mbox_per_channel`, or the number of I/O buffers per slab, `idn_slab_bufcount`.
`WARNING: IDN: 110: maximum number of slabs(`*number*`) < minimum required(`*number*`)`	The value of the `idn_slab_maxtotal` variable is less than the required minimum. The value of this variable is calculated by the IDN driver. The driver returns this warning if the value is less than the minimum value for nominal usage of the IDN.	Lower the value of the `idn_slab_mintotal` variable, or increase the value of the SMR so that it can hold more slabs. You may need to adjust the value of other driver parameters, such as `idn_slab_bufcount`, to recover from this error.
`WARNING: IDN: 111: idn_smr_bufsize(`*bytes*`) not on a 64 byte boundary`	The value of the `idn_smr_bufsize` variable must be a multiple of 64.	Reset the `idn_smr_bufsize` variable to a multiple of 64.
`WARNING: IDN: 112: idn_smr_bufsize(`*bytes*`) not a power of 2`	The value of the `idn_smr_buffsize` variable must be set to a value that is a power of two.	Reset the `idn_smr_bufsize` variable to a power of two.
`WARNING: IDN: 113: idn_mbox_per_net(`*number*`) must be an odd number`	For proper hashing, the value of the `idn_mbox_per_net` variable must be an odd number.	Reset the `idn_mbox_per_net` variable to an odd number.

TABLE A-2 IDN Domain-Specific Messages, 100 through 142 *(Continued)*

Error	**Description**	**Recovery**
`WARNING: IDN: 115: idn_netsvr_wait_min (`*seconds*`) cannot be greater than idn_netsvr_wait_max (`*seconds*`)`	The minimum wait time for the IDN net server cannot be greater than the maximum wait time.	Decrease the value of `idn_netsvr_wait_min`, or increase the value of `idn_netsvr_wait_max`.
`WARNING: IDN: 116: failed rmalloc(kernelmap,` *number* `pages)`	The operating system failed to allocate pages *number* of the virtual address space for the mapping of the SMR.	Reboot the domain, then attempt to link the domain to the IDN.
`WARNING: IDN: 117: IDN not enabled`	The IDN driver failed to initialize the IDN because the IDN was not enabled (that is, `idn-smr-size` is set to zero).	At the OBP prompt, use the `ndd`(1M) command to check the value of the `idn-smr-size` variable. If it is set to zero, reset it to the appropriate value to enable the IDN and to set the size of the SMR. You must reboot the domain if you change the value of this variable.
`WARNING: IDN: 118: hardware config not appropriate`	The hardware configuration within the domain that you are trying to link is not appropriate for an IDN, or the IDN driver could not determine the hardware configuration. The hardware configuration within a domain could cause this error if it consists of one of the following configurations: A system board that hosts memory does not host a CPU. System boards in domains must host at least one CPU for the domain to be considered eligible for linking to an IDN. See option `no_non_proc_boards` in the `post`(4) man page on the SSP for information on how to prevent system boards without a CPU from being included in the domain. The shared memory mask on the CIC is not enabled to allow the IDN driver to manipulate the registers.	Ensure that each system board that contains memory has at least one CPU and that SSP 3.2 is running on the SSP.

TABLE A-2 IDN Domain-Specific Messages, 100 through 142 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 119: failed to initialize` *number* `protocol servers`	The IDN driver failed to start up the *number* protocol servers that manage the IDN connections. Typically, a strain on kernel resources causes this error.	Reload the IDN driver. If this error occurs repeatedly, reduce the memory usage by other applications, if possible.
`WARNING: IDN: 120: cannot deinit while active (state =` *GSTATE*`)`	The DR operation tried to deinitialize the IDN driver while it was still in use (that is, not offline).	Ensure that the domain is completely unlinked from all IDNs before you initiate a DR operation on the domain.
`IDN: 121: domain` *domain_ID* `(cpu` *CPUID*`, name "`*host*`", state` *DSTATE*`)`	This error displays the IDN state of each domain connection. This error message occurs in conjunction with messages 104 and 120.	See error messages 104 and 120.
`WARNING: IDN: 123: unexpected M_DATA packets for q_stream` *VADDR*	The STREAMS read procedure in the IDN driver received an unexpected data packet on the respective stream queue. The packet is discarded.	If this error persists, report the problem to your Sun Microsystems, Inc., service representative.
`WARNING: IDN: 124: sigblk for cpuid` *CPUID* `is NULL`	The CPU in question expected to receive messages from the SSP; however, the signature block (BBSRAM) data structure for that CPU is not mapped properly.	Unload, then reload, the IDN driver. If that does not work, unlink the domain, then reboot it. Relink the domain after it has booted successfully.
`WARNING: IDN: 125: op (`*IDNOP*`) failed, returning (`*errno*/*IDNERR* `[`*EPARAM0*`,` *EPARAM1*`,` *EPARAM2*`])`	An attempted IDN operation (link, unlink, or info) failed. In the message, *errno* equals the error number, *IDNERR* equals the IDN error, and *EPARAM2* represents the parameters that are dependent on this type of IDN error.	Ensure that you used the correct parameters. If not, retry the operation with the correct parameters.
`WARNING: IDN: 126: sighandler thread already exists (`*VADDR*`)`	The IDN driver attempted to create an unnecessary, duplicate, sigblock-interrupt-handler thread.	You can safely ignore this message.
`WARNING: IDN: 127: cannot change` *parameter* `while IDN connected`	The IDN tunable parameter that you tried to change cannot be changed while the domain is linked to an IDN.	Unlink the domain before you attempt to change the parameter settings for the specified parameter (*parameter*).
`WARNING: IDN: 128: cannot change` *parameter* `while DLPI attached`	The IDN tunable parameter that you tried to change cannot be changed while the IDN interface (`idn`*X*) is plumbed under TCP/IP.	Unplumb all of the IDN interfaces that are under TCP/IP control before you attempt to change the parameter settings for the specified parameter (*parameter*).

TABLE A-2 IDN Domain-Specific Messages, 100 through 142 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 129:` *IDNOP* `operation timed out`	An IDN operation (link, unlink, or info) exceeded the specified wait-time before it successfully completed.	Check for AWOL domains in the IDN. If present, unlink the AWOL domain before you attempt to unlink the local domain. You can use the force option, `-f` or `-F`; however, you must use the force option with caution.
`WARNING: IDN: 130: IDN DMV handler already initialized`	The IDN driver attempted to initialize the interrupt handler too many times.	Reboot the domain. If this error persists, contact your Sun service representative.
`WARNING: IDN: 131; unable to allocate data area for DMV handler`	The IDN driver was unable to allocate a data area for control structures that are used by the DMV handler.	This error typically occurs when memory usage levels are too high. Retry the IDN command when the usage level decreases, or remove some of the system activity that is causing the high usage level.
`WARNING: IDN: 132: failed to add IDN DMV handler`	The IDN driver failed to register its internal interrupt handler with the DMV-based kernel subsystem.	Reboot the domain, or unload the IDN driver and retry the operation. If this error persists, report the problem to your Sun service representative.
`WARNING: IDN: 133: sigblock event area missing`	The signature block event area that is used between the IDN driver and the SSP may be missing or corrupted.	Reboot the domain, or unload the IDN driver and retry the operation. If this error persists, report the problem to your Sun service representative.
`IDN: 134: unable to mark boardset (`*BMASK*`) AWOL`	This error returns a 16-bit board mask that indicates which boards in the domain could not be marked as AWOL so that they could be handled by the SSP.	Unlink the domain(s) in question, then relink them to the domain. You may need to reboot the domain or to unload the IDN driver before you retry the operation.
`IDN: 135: idn: kstat_create failed`	Unable to create the `kstat` structures. Thus, global kernel statistics for the IDN are not maintained.	Reboot the domain, and retry the command, if necessary.
`WARNING: IDN: 136: "`*property*`" property not found, disabling IDN`	The IDN driver failed to initialize because it did not find the specified OBP property.	Ensure that the SSP 3.2 software is on the SSP.
`WARNING: IDN: 137: SMR size is 0, disabling IDN`	The IDN driver failed to initialize because the OBP variable `idn-smr-size` is set to zero (0).	Reset the `idn-smr-size` variable to the appropriate value. The value must be the same for all of the domains in an IDN. Reboot the domain after you reset the value of the variable.

TABLE A-2 IDN Domain-Specific Messages, 100 through 142 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 138: SMR size (`*number*`MB) is too big (max = `*number*`MB), disabling IDN`	The IDN driver failed to initialize because the value of the OBP variable `idn-smr-size` is too large.	Reset the `idn-smr-size` variable to the appropriate value. The value must be the same for all of the domains in an IDN. Reboot the domain after you reset the value of the variable.
`WARNING: IDN: 139: OBP region for SMR is 0 length`	The IDN driver failed to initialize because the OBP variable `idn-smr-size` is set to zero (`0`).	Reset the `idn-smr-size` variable to the appropriate value. The value must be the same for all of the domains in an IDN. Reboot the domain after you reset the value of the variable.
`WARNING: IDN: 140: OPB region (`*bytes* `B) smaller than requested size (`*bytes* `B)`	The SMR region allocated by the OBP is smaller than the value of the OBP variable `idn-smr-size`.	Ensure that the SSP 3.2 software is running on the SSP, then reset the `idn-smr-size` variable to the appropriate value. The value must be the same for all of the domains in an IDN. Reboot the domain after you reset the value of the variable.
`WARNING: IDN: 141: OPB region (`*PADDR*`) not on (`*hex_number*`) boundary`	The SMR region allocated by OBP is not on the appropriate memory address boundary (64-Kbytes).	Ensure that the SSP 3.2 software is running on the SSP, then reboot the domain and retry the operation.
`NOTICE: IDN: 142: link (domain `*domain_ID*`, cpu `*CPUID*`) connected`	The domain has been linked with domain *domain_ID* that hosts CPU *CPUID*.	Notification only

TABLE A-3 IDN Domain-Specific Messages, 200 through 242

Error	Description	Recovery
`NOTICE: IDN: 200: link (domain `*domain_ID*`, cpu `*CPUID*`) disconnected`	The domain has been unlinked with domain *domain_ID* that hosts CPU *CPUID*.	Notification only
`WARNING: IDN: 201: (`*IDNOP*`) invalid cpu-id (`*CPUID*`)`	During the IDN operation, the IDN driver specified an invalid ID number for the CPU.	Retry the IDN operation.
`WARNING: IDN: 202: (`*IDNOP*`) invalid time-out value (`*seconds*`)`	An invalid time out value was passed to the IDN operation (*IDNOP* = LINK / UNLINK).	Retry the IDN operation.
`WARNING: IDN: 203: (`*IDNOP*`) invalid domain-id (`*domain_ID*`)`	An invalid domain ID was passed to the IDN operation (*IDNOP* = LINK / UNLINK).	Retry the IDN operation.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 204: domain` *domain_ID* `state (`*DSTATE*`) inappropriate`	The domain *domain_ID* was not in the closed state when the IDN operation was executed.	Retry the IDN operation. If this error persists, unlink the domain.
`WARNING: IDN: 205: (`*function*`) failed to open-domain (`*domain_ID*`,`*CPUID*`)`	The internal IDN function *function* failed to open an IDN domain control structure for managing an IDN connection with domain *domain_ID* and CPU *CPUID*.	Wait a few minutes for the connection to be resolved. If the connection cannot be resolved, reboot domain *domain_ID*.
`WARNING: IDN: 206: cannot link domains with equal votes (L(`*domain_ID*`), R(`*domain_ID*`),`*hex_number*`)`	The local domain, *domain_ID*, has the same vote ticket, *hex_number*, as the domain, *domain_ID*, to which it is trying to link. The vote tickets are determined internally and must be unique.	Reboot the local domain, then retry the operation. If the error occurs again, reboot domain *domain_ID*, then retry the operation. If the error persists, contact your Sun service representative.
`WARNING: IDN: 207: local/remote master-id conflict (`*local_domain_ID*`.lmasterid =` *domain_ID_a*`,` *remote_domain_ID*`.rmasterid =` *domain_ID_b*`)`	The local domain, *local_domain_ID*, has a master ID, *domain_ID_a*, that conflicts with the master ID, *domain_ID_b*, of domain *remote_domain_ID*.	Retry the operation. If the error persists, reboot both of the domains, then retry the operation.
`WARNING: IDN: 208: idn_select_master: unknown case (`*number*`)`	This is an internal error. During the selection of the master domain, the IDN driver encountered an unexpected case, *number*.	Retry the operation. If the error persists, reboot both of the domains, then retry the operation.
`WARNING: IDN: 209: remote domain (id` *domain_ID*`, cpu` *CPUID*`) reporting master (id` *master_domain_ID*`) without cpuid`	During the IDN operation, the remote domain, *domain_ID*, returned the ID of the master to the local domain, but not the CPU ID.	Retry the operation. If this error persists, reboot the remote domain.
`WARNING: IDN: 210: failed to init MASTER context`	This is an internal error. The local domain failed to initialize control structures that the domain needs to be a master domain.	Reboot the local domain, then retry the operation.
`WARNING: IDN: 211: disconnect domain` *domain_ID*`, unexpected GSTATE (`*GSTATE*`)`	During the disconnect operation for domain *domain_ID*, the local IDN was in an unexpected global state, *GSTATE*.	Reboot the local domain, then retry the operation.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`PANIC: IDN: 212: disconnect domain` *domain_ID*`, bad GSTATE (`*GSTATE*`)`	During the disconnect operation for domain *domain_ID*, the local IDN was in an unexpected global state, *GSTATE*.	Reboot the local domain, then retry the operation.
`WARNING: IDN: 213: no destination specified (d=`*domain_ID*`, c=`*channel*`, n=0x`*NE*`)`	The local domain attempted to send a data packet to an inappropriate destination, domain ID *domain_ID*, on channel *channel* with the network ID *NE*.	Retry the operation. If this error persists, unlink the local domain, then relink it and retry the operation.
`WARNING: IDN: 214: received message (`*MSG*`[0x`*M_number*`]) from self (domid` *domain_ID*`)`	The local domain received an IDN connection protocol message, *MSG*[0x*M_number*], from itself, domain ID *domain_ID*.	Reboot the local domain, then retry the operation.
`WARNING: IDN: 215: invalid cookie (`*cookie*`) for message (`*M_number*`) from domain` *domain_ID*	The local domain received an IDN connection protocol message, *M_number*, from domain *domain_ID* with an invalid or stale cookie, *cookie*.	Retry the operation. If this error persists, unlink the local domain and/or remote domain, then relink the local and/or remote domain.
`WARNING: IDN: 216: (`*M_number*`)msgtype/ (`*A_number*`)acktype rcvd from domain` *domain_ID*	The local domain received an invalid IDN connection protocol message or acknowledgement (*M_number*/*A_number*) from domain *domain_ID*.	Retry the operation. If this error persists, unlink the local and/or remote domain, then relink the local and/or remote domain.
`WARNING: IDN: 217: unknown CFGARG type (`*type*`) from domain` *domain_ID*	The local domain received an unexpected type, *type*, of configuration message from the remote domain, *domain_ID*.	Retry the operation. If this error persists, unlink the local and/or remote domain, then relink the local and/or remote domain.
`WARNING: IDN: 218: missing some required config items from domain` *domain_ID*	During the connection operation, the local domain did not receive all of the configuration information it expected from domain *domain_ID*.	Retry the operation. If this error persists, unlink the local and/or remote domain, then relink the local and/or remote domain.
`WARNING: IDN: 219: remote domain` *domain_ID* `MTU (`*bytes*`) invalid (local.mtu =` *bytes*`)`	The MTU size, *bytes*, received from domain *domain_ID* is not compatible with the MTU size of the local domain, *bytes*. The value of the IDN tunable parameter `idn_smr_bufsize` must be the same for all of the domains in an IDN.	Reset the value of the `idn_smr_bufsize` parameter on the local domain or the remote domain so that the values are the same.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 220: remote domain` *domain_ID* `BUFSIZE (`*bytes*`) invalid (local.bufsize =` *bytes*`)`	The local domain received an SMR buffer size, *bytes*, from the remote domain that is not compatible with the SMR buffer size of the local domain. The value of the IDN tunable parameter `idn_smr_bufsize` must be the same for all of the domains in an IDN.	Reset the value of the `idn_smr_bufsize` parameter on the local domain or the remote domain so that the values are the same.
`WARNING: IDN: 221: remote domain` *domain_ID* `SLABSIZE (`*bytes*`) invalid (local.slabsize =` *bytes*`)`	The local domain received an SMR slab size, *bytes*, from domain *domain_ID* that is not compatible with the SMR slab size of the local domain. The value of the IDN tunable parameters `idn_slab_bufcount` and `idn_smr_bufsize` must be the same for all of the domains in an IDN.	Reset the slab size parameters on the local and/or remote domain.
`NOTICE: 222: no IDN linkage found (b=`*BMASK_a*`, i=`*BMASK_b*`) upgrading unlink` *FTYPE* `-> FORCE_HARD`	The SSP requested that the local domain be unlinked from the remote domain with boardmask *FTYPE*; however, the SSP was unable to find a hardware link in the IDN hardware register board mask *BMASK_b*. The specified soft force option, `-f`, was upgraded to the hard force option, `-F`, to unlink the domain.	None
`WARNING: IDN: 223: remote domain` *domain_ID* `NWRSIZE (`*Mbytes*`) invalid (local.nwrsize =` *Mbytes*`)`	The local domain received a value of the `idn_nwr_size` variable from the remote domain that does not match the value of `idn_nwr_size` on the local domain. The value of the `idn_nwr_size` variable must be the same for all of the domains in the IDN.	Reset the value of the `idn_nwr_size` variable for the remote domain.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 224: remote domain` *domain_ID* `idn_max_nets (`*number*`) invalid (local.maxnets =` *number*`)`	The local domain received a value of the `idn_max_nets` variable from the remote domain that does not match the value of `idn_max_nets` on the local domain. The value of the `idn_max_nets` variable must be the same for all of the domains in the IDN.	Reset the value of the `idn_max_nets` variable for the remote domain.
`WARNING: IDN: 225: remote domain` *domain_ID* `MBOX_PER_NET (`*number*`) invalid (local.mboxpernet =` *number*`)`	The local domain received a value for the `idn_mbox_per_net` variable from the remote domain that does not match the value of `idn_mbox_per_net` on the local domain. The value of the `idn_mbox_per_net` variable must be the same for all of the domains in the IDN.	Reset the value of the `idn_mbox_per_net` variable for the remote domain.
`WARNING: IDN: 226: remote domain` *domain_ID* `CHECKSUM flag (`*number*`) mismatches local domain's (`*number*`)`	The local domain received a value for the `idn_checksum` variable from the remote domain that does not match the value of `idn_checksum` on the local domain. The value of the `idn_checksum` variable must be the same for all of the domains in the IDN.	Reset the value of the `idn_checksum` variable for the remote domain.
`WARNING: IDN: 227: missing some required config items from domain` *domain_ID*	The local domain did not receive all of the expected configuration information from the remote domain, *domain_ID*.	Retry the link operation. If this error persists, reboot the remote domain, then retry the link operation.
`WARNING: IDN: 228: master's SMR (`*bytes*`) larger than local's SMR (`*bytes*`)`	The size the SMR for the master domain is larger than the virtual space the local domain has available for the SMR. The OBP variable `idn-smr-size` must be the same for all of the domains in an IDN.	Reset the size of the `idn-smr-size` variable for the local domain and/or the remote domains so that the size is the same. Reboot the domain(s), then retry the link operation.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 229: remote domain` *domain_ID* `boardset (`*BMASK*`) conflicts with MCADR(board` *number*`) [`*MCADR*`]`	The local domain received conflicting information about the MCADR register from domain *domain_ID*. Board *number* is reported to have a MCADR setting, but it is not present in the physical board set *BMASK* of the remote domain. *MCADR* represents the actual MCADR register value.	Reboot domain *domain_ID*, then retry the operation.
`WARNING: IDN: 230: remote domain` *domain_ID* `reported number of MCADRs (`*number*`) mixmatches received (`*number*`)`	The local domain received conflicting information about the MCADRs in domain *domain_ID*. The number of MCADRs specified by domain *number* does not match the number of MCADRs reported by that domain.	Reboot domain *domain_ID*, then retry the operation.
`WARNING: IDN: 231: domain` *domain_ID* `boardset (`*BMASK*`) conflicts with existing IDN boardset (`*BMASK*`)`	The set of boards in the board mask *BMASK* for domain *domain_ID* overlap the existing boards *BMASK* in the IDN.	Ensure that the local domain has past the power-on self-test (POST). Unlink the domain, then relink it.
`WARNING: IDN: 232: domain` *domain_ID* `cpuset (`*CPUSET*`) conflicts with existing IDN cpuset (`*CPUSET*`)`	The set of CPUs in the CPU mask *CPUSET* for domain *domain_ID* overlap the existing CPUs *CPUSET* in the IDN.	Ensure that the local domain has past the power-on self-test (POST). Unlink the domain, then relink it.
`WARNING: IDN: 233: domain` *domain_ID* `missing cpu per memory boardset (`*BMASK*`), cpu boardset (`*BMASK*`)`	Each system board in the domain *domain_ID* must have at least one CPU if it hosts memory. Board set *BMASK* represents the board that have memory, and board set *BMASK* represents the boards that have at least one CPU.	Ensure that the domain you want to link has system boards that host at least one CPU on each board that hosts memory.
`WARNING: IDN: 234: failed to program hardware for domain` *domain_ID* `(boardset =` *BMASK*	The IDN driver was unable to program the hardware for the local domain to allow shared memory access with domain *domain_ID*, which contains the board set *BMASK*.	Do not execute additional IDN operations. When appropriate, halt domain *domain_ID*, and run a full diagnostic test by using the `hpost`(1M) command.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 235: [`*MBXTYPE*`] mailbox (domain` *domain_ID*`, channel` *channel*`) SMR CORRUPTED - RELINK` `IDN: 235: [`*MBXTYPE*`] expected (cookie` *cookie*`, cksum` *hex_number*`), actual (cookie` *cookie*`, cksum` *hex_number*`)` `IDN: 235: [`*MBXTYPE*`] activeptr (`*VADDR*`), readyptr (`*VADDR*`)`	The local domain detected that the send and/or receive mailbox *MBXTYPE* control area is corrupted for domain *domain_ID*. These messages indicate the expected and actual values of the cookie and checksum control information. Depending on the condition, an additional message is displayed containing additional control information that is used for synchronization during data transmissions (`activeptr` and `readyptr`).	Unlink the master domain, then relink it. If this error persists, dismantle the entire IDN, then reassemble it.
`WARNING: IDN: 236: domain (`*host*`) [id` *domain_ID*`] not responding to` *IDN_command* `[#`*number*`]` `WARNING: IDN: 236: domain [id` *domain_ID*`, cpu` *CPUID*`] not responding to` *IDN_command* `[#`*number*`]`	The local domain attempted to connect or disconnect domain *host* or domain ID *domain_ID* with CPU ID *CPUID*; however, the domain is not responding. *number* represents the number of detected AWOL messages.	Unlink domain *host*, then retry the link operation.
`WARNING: IDN: 237: invalid number (`*number*`) of protocol servers`	The specified number of IDN protocol servers is invalid. The value of the `idn.conf`(4) tunable parameter `idn_protocol_nservers` must be greater than zero (`0`).	Reset the value of the `idn_protocol_nservers` parameter in the `idn.conf`(4) file to an appropriate number.
`WARNING: IDN: 238: kmem_cache_create (jobcache) failed`	The kernel failed to create an internal cache for allocating IDN job control data structures.	Reboot the local domain, then retry the link operation. If this error persists, remove unused software from the domain, and retry the link operation.
`WARNING: IDN: 239: invalid cpuid (`*CPUID*`) specified for IDN net` *channel*	An invalid CPU ID, *CPUID*, was specified for the local domain. *channel* represents the network interface for the IDN data server thread.	Retry the operation with a valid CPU ID for the local domain.

TABLE A-3 IDN Domain-Specific Messages, 200 through 242 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 240: (channel` *channel*`) SMR CORRUPTED - RELINK` `IDN: 240: (channel` *channel*`) cookie (expected` *cookie*`, actual` *cookie*`)` `IDN: 240: (channel` *channel*`) actv_flg (expected` *hex_number*`, actual` *hex_number*`)` `IDN: 240: (channel` *channel*`) ready_flg (expected` *hex_number*`, actual` *hex_number*`)`	The IDN data server for network interface *channel* encountered corrupted data in the SMR. The expected and actual values for control information that is used by the data server are included (`cookie`, `actv_flg`, and `ready_flg`). Subsequent data transmissions are likely to fail.	Unlink the master domain, then relink it. If this error persists, dismantle the IDN, then reassemble it.
`WARNING: IDN: 241: [`*operation*`] (domain` *domain_ID*`, channel` *channel_ID* `SMR CORRUPTED - RELINK)`	The IDN driver attempted to transmit or receive data to or from an IDN mailbox in the SMR; however, the SMR was corrupted. The operation is designated as operation, `send` or `recv`, in the message. Future data transmissions are likely to fail.	Unlink the master domain, then relink relink it. If this error persists, dismantle the IDN, then relink the domains.
`WARNING: IDN: 242: maximum channels (`*number*`) already open`	You cannot plumb more network interfaces than the IDN driver is configured to support.	Reset the `idn.conf`(4) tunable `idn_max_nets`, then retry the operation.

TABLE A-4 IDN Domain-Specific Messages, 300 through 307

Error	Description	Recovery
`WARNING: IDN: 300: no slab allocations without a master`	The slave domain attempted to allocate a slab of memory without a master domain being present.	If this error persists, unlink the local domain, then relink it.
`WARNING: IDN: 301: (`*SMROP*`) unknown slab state (`*slab_state*`) for domain` *domain_ID*	This is an internal error, indicating that a SMR slab was in an unexpected state for domain ID *domain_ID*, with respect to slab operation *SMROP*.	Unlink the local domain, then relink it.
`WARNING: IDN: 302: no slab free without a master`	The slave domain attempted to free up a slab of memory without the master domain being present.	If this error persists, unlink the local domain, then relink it.

TABLE A-4 IDN Domain-Specific Messages, 300 through 307 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 303: buffer len` *bytes* `> IDN_DATA_SIZE (`*bytes*`)`	The local domain attempted to allocate an SMR buffer with a length greater than the length supported by the IDN configuration.	Unlink the local domain, then relink it.
`WARNING: IDN: 304: buffer (`*VADDR*`) from domain` *domain_ID* `not on a` *hex_number* `boundary`	An SMR buffer at kernel virtual address *VADDR* was received from domain ID *domain_ID*; however, it was not aligned on the expected boundary *hex_number* (in bytes).	Unlink the local domain and/or domain *domain_ID*, then relink the local domain and/or domain *domain_ID*.
`WARNING: IDN: 305: buffer length (`*bytes*`) from domain` *domain_ID* `greater than IDN_DATA_SIZE (`*bytes*`)`	An SMR buffer of length *bytes* was received from domain ID *domain_ID*; however, the length was greater than the length that is supported by the local domain (*bytes* in bytes).	Unlink the local domain and/or domain *domain_ID*, then relink the local domain and/or domain *domain_ID*.
`WARNING: IDN: 306: unknown buffer (`*hex_number*`) from domain` *domain_ID*	The local domain received a request for a domain ID, domain_ID, that was not in the expected range of valid domain IDs (that is, 0 to 15).	If this error persists, unlink the local domain, then relink it.
`WARNING: IDN: 307: domain id (`*domain_ID*`) invalid`	The SMR subsystem received a request for a domain ID *domain_ID* that was not in the expected range of valid domain IDs (that is, 0 to 15).	If this error persists, unlink the local domain, then relink it.

TABLE A-5 IDN Domain-Specific Messages, 400 through 450

Error	Description	Recovery
`WARNING: IDN: 400: corrupted MAC header (exp` *hex_number* `or 0xffff, act` *hex_number*`)`	The MAC header in the SMR data packet contained bad data.	Unlink the local domain, then relink it. If this error persists, dismantle the IDN, and reassemble it.
`IDN: 450: idn`*X*`: kstat_create failed`	Unable to create the `kstat` structures. No per-interface kernel statistics will be maintained for the IDN.	Reboot the local domain if you need the per-instance kernel statistics to be maintained.

TABLE A-6 IDN Domain-Specific Messages, 500 through 516

Error	Description	Recovery
`WARNING: IDN: 500: failed to write sm_bar (lsb/msb) (`*hex_number*`)`	While programming the SMR, the local domain failed to write the (`lsb/msb`) portion of the shared memory base-address-register with value *hex_number*.	Do not perform IDN operations. When appropriate, halt the local domain, and run a full diagnostic test by using the `hpost`(1M) command.
`WARNING: IDN: 501: failed to write sm_lar (lsb/msb) (`*hex_number*`)`	While programming access to the SMR, the local domain failed to write the (`lsb/msb`) portion of the shared memory limit-address-register with value *hex_number*.	Do not perform IDN operations. When appropriate, halt the local domain, and run a full diagnostic test by using the `hpost`(1M) command.
`WARNING: IDN: 502: unable to store data (`*hex_number*`) to CIC buffer (`*PADDR*`)`	While programming access to the SMR, the local domain failed to write to the CIC (Coherency Interface Controller) *prep* buffer with the data *hex_number*.	Do not perform IDN operations. When appropriate, halt the local domain, and run a full diagnostic test by using the `hpost`(1M) command.
`WARNING: IDN: 503: (`*PCPROG*`) failed to update PC madr (expected 0xXXX, actual `*hex_number*`)`	While programming access to the SMR during the *PCPROG* (invalidate or validate) phase, the local domain failed to write a memory address decoding register (MADR) entry to the port controller (PC).	Do not perform IDN operations. When appropriate, halt the local domain, and run a full diagnostic test by using the `hpost`(1M) command.
`WARNING: IDN: 504: (`*PCPROG*`) failed to update IOPC madr (expected `*hex_number*`, actual `*hex_number*`)`	While programming access to the SMR during the *PCPROG* (invalidate or validate) phase, the local domain failed to write a memory address decoding register (MADR) entry of the I/O port controller (IOPC).	Do not perform IDN operations. When appropriate, halt the local domain, and run a full diagnostic test by using the `hpost`(1M) command.
`WARNING: IDN: 505: board `*number*` mising any valid PCs`	Board *number* does not contain valid port controllers (PCs).	Do not perform IDN operations. When appropriate, halt the local domain to ensure that board *XX* hosts the appropriate hardware. You may need to run a full diagnostic test by using the `hpost`(1M) command.
`WARNING: IDN: 506: cic sm_mask is not writable`	The CIC has been programmed by POST to not let the operating system level software to manipulate the shared-memory mask register.	Ensure that the SSP 3.2 software is running on the SSP. Run a full diagnostic test on the board by using the `hpost`(1M) command.

TABLE A-6 IDN Domain-Specific Messages, 500 through 516 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 507: failed to map-in post2obp structure`	The local domain did not successfully map in the `POST2OBP` data structure to the address space of the kernel.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 508: post2obp checksum invalid`	The `POST2OBP` data structure that was passed from POST to OBP appeared invalid. The expected checksum value did not match the value that is reported by the data structure.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 509: cpu` *CPUID* `never responded to CIC update`	While programming the SMR for access during the parallel update of the CIC registers phase, CPU *CPUID* did not respond to the update request.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 510: failed write-smregs (bd=`*number*`, bs=`*bus*`, sm(bar=`*bar*`, lar=`*lar*`)`	The local domain failed to update all of its shared memory registers. The specific failure occurred on system board *number*, interconnect bus *bus*, with base / limit-address-register contents of *bar* / *lar*, respectively.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 511: update-one (cpu=`*CPUID*`, bd=`*number*`) time conflict`	A stale IDN-hardware update operation was encountered during the update of the shared memory registers across the system.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 512: failed [`*add/delete*`] write-madr (bd=`*number*`, rbd=`*remote_number*`, madr=`*hex_number*`)`	The local domain failed to update (to add and / or to delete) the PC memory address decoding registers on board *number* with respect to the remote board, *remote_number*. The *hex_number* value represents the targeted register contents.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 513: sm-mask error (expected =` *hex_number*`, actual =` *hex_number*`)`	The local domain encountered inconsistent or unexpected values in the shared memory mask of the CIC.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 514: sm-base error (expected =` *hex_number*`, actual =` *hex_number*`)`	The local domain encountered inconsistent or unexpected values in the shared memory base register of the CIC.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.

TABLE A-6 IDN Domain-Specific Messages, 500 through 516 *(Continued)*

Error	Description	Recovery
`WARNING: IDN: 515: sm-limit error (expected =` *hex_number*`, actual =` *hex_number*`)`	The local domain encountered inconsistent or unexpected values in the shared memory limit register of the CIC.	Halt the local domain, then reboot it. After it reboots, relink it to the IDN.
`WARNING: IDN: 516: (`*local/remote*`) board` *number* `has memory, but no cpus - CPU-PER-BOARD REQUIRED`	The local domain detected that a local or remote, *local/remote*, system board, *number*, contains memory, but no CPUs. In an IDN, each system board that hosts memory must also host at least one CPU.	Halt the local or remote domain, then check its hardware configuration. If it does not host a CPU, place it in the blacklist, then relink the domain.

APPENDIX **B**

IDN Error Messages, Notifications, and Panics on the SSP

This chapter contains the IDN error messages, notifications, and panics that occur, or are recorded, on the SSP.

IDN messages that occur on the SSP are sent to the following locations:

- `netcon`(1M) console
- `/var/adm/messages`
- `$SSPLOGGER/messages`

Note – When an IDN-related error occurs, you may see several messages that relate to the error or that give further explanation of the error. Those messages are included in the following tables.

Searching this Appendix

Locating specific error messages in this appendix depends entirely on the media type you are using. If you are using this appendix online, . If you are using this appendix in hard-copy form, search the tables alphabetically starting with the first character in the error message.

Online Searching

You can use the search engine provided in the AnswerBook2™ environment or the search engine in your browser to find a specific string of characters from an error message. Before you construct the search string, keep in mind that this appendix contains special typographical conventions. In addition, you may need to search all of the tables individually. If you know the error type (that is, where the error was encountered), use the hypertext links in “Error Type Links” to start your search.

Special Typographical Conventions

The tables in this appendix contain special typographical conventions for the names of words and values that change, depending on the type of error. When you search for an error message, keep in mind that these names appear as generic representations in italic font. The following list contains the commonly used representations used in this appendix.

- *domain_ID* for the value of the domain ID
- *domain_name* for the names of all domains
- *domain_name_a, domain_name_b, domain_name_c* for the names of the domains used with the IDN commands
- *platform_name* for the name of the Sun Enterprise™ 10000 platform
- *process_id* for the value of the process ID (pid number)
- *system_board_number* for the number of a system board (that is, 1 through 15)
- *XX* for numeric values

Error Type Links

This section contains links to each of the major error type tables.

- “IDN Environment Errors”
- “Host Environment Errors”
- “General Host Errors”
- “IDN-Related Command Errors”

Hard-Copy Searching

If you are searching this appendix in hard-copy form, the tables have been sorted alphabetically to help in your search. The messages in this appendix are grouped by their error type. The error types are IDN environment errors, host environment errors, general host errors, and IDN-related command errors. If you know the type of error, start your search in that section of this appendix.

IDN Environment Errors

The following table contains the IDN environment errors that occur on the SSP.

TABLE B-1 IDN Environment Errors Recorded on the SSP

Error	Probable Cause	Suggested Action
`domain_link error: domain_link: File not found in /opt/SUNWssp/release/Ultra-Enterprise-10000/5/5/1/bin or hostobjs. Please check the environment variable SUNW_HOSTNAME.`	The domain name in the `SUNW_HOSTNAME` environment variable does not support the IDN feature. The domain specified in this variable must support the IDN feature.	Set the `SUNW_HOSTNAME` variable to a valid domain name.
`domain_link error:` *domain_name* `is not a valid domain. Please set SUNW_HOSTNAME to a valid domain name.`	The `SUNW_HOSTNAME` environment variable is set to an invalid domain name. This variable must be set to the name of a domain that supports the IDN feature.	Set the `SUNW_HOSTNAME` variable to a valid domain name.
`domain_link: a domain cannot be IDN linked to itself. Usage: domain_link domain_name1 domain_name2`	The domain names specified in the command were the same.	Use the domain names of different domains.
`domain_link: cannot open connection to the snmpd agent` `domain_link: IDN initialization failed (idnerr = ERR_SSP_SNMP(0x200)`	The SSP could not open a connection to the `SNMPD` agent.	Use the `ps`(1) command to ensure that the `SNMPD` process is running. If not, wait until it restarts or until the SSP program starts up successfully, then retry the command.
`domain_link: domain [`*domain_name*`] OS version (5.6) not supported`	The domain, *domain_name*, does not support the IDN feature. Both of the names specified in the `domain_link`(1M) command must support the IDN feature.	Use valid domain names with the `domain_link`(1M) command.
`domain_unlink: domain [`*domain_name*`] is in a different IDN.`	The domain names specified in the command are not in the same IDN.	Use domain names that are in the same IDN.
`domain_unlink: domain [`*domain_name*`] not found`	The domain name specified in the command does not exist.	Select another domain name.

TABLE B-1 IDN Environment Errors Recorded on the SSP *(Continued)*

Error	Probable Cause	Suggested Action
`RPC: Miscellaneous tli error - An event requires attention. No such file or directory domain_unlink: IDN initialization failed (idnerr = ERR_SSP_CBS(0x202))`	The SSP cannot connect to the CBS.	Use the `ps`(1) command to ensure that the CBS daemon is running. If not, wait until it restarts or until the SSP program starts up successfully, then retry the command.

Host Environment Errors

The following table contains the host environment errors. These errors indicate that the host set up is invalid.

TABLE B-2 Host Errors Recorded on the SSP

Error	Probable Cause	Suggested Action
`domain_link: Another IDN operation is currently IN-PROGRESS (pid` *process_id*`)` `Retry when other operation has completed.`	An IDN operation (for example, the `domain_link`(1M) command, the `domain_unlink`(1M) command, or an IDN event) was in progress.	Wait for the IDN process to complete successfully, then retry the `domain_link`(1M) command.
`domain_link: ...Checking IDN state of [`*domain_name*`] : NOTSUPPORTED domain_link: [`*domain_name*`] does not support IDN`	The IDN driver was not initialized on the domain, *domain_name* because the value of the OpenBoot™ PROM (OBP) variable `idn-smr-size` is set to zero.	Set the value of `idn-smr-size` to a size other than zero, and reboot the domain. Then, retry the `domain_link`(1M) command.
`domain_link: Failed to acquire IDN specific lock.`	Another IDN operation is in progress.	Wait for the current IDN operation to complete successfully. Then, retry the `link`(1M) operation.

TABLE B-2 Host Errors Recorded on the SSP *(Continued)*

Error	Probable Cause	Suggested Action
`domain_unlink:` `...Checking IDN state of` [*domain_name*] `: UNKNOWN` `domain_unlink: Cannot proceed without known domain state.` `domain_unlink: IDN UNLINK operation unsuccessful` [*domain_name*] `Retry domain_unlink(1M).`	The SSP cannot determine the state of the IDN domain. The IDN driver may be hung, or the CBE is not running.	Check the domain to ensure that the IDN driver is not hung and/or that the CBE is running. If necessary, reboot the domain to reset the IDN state.
`domain_unlink:` `...Checking IDN state of` [*domain_name*] `: NOTSUPPORTED` `domain_unlink: Domain` [*domain_name*] `does NOT support IDN.` `Must force (-f) to unlink, if necessary.`	The IDN driver was not loaded on the domain.	Ensure that the domain name is correct, then retry the `domain_unlink`(1M) command. If necessary, use the soft force option, `-f`, with the `domain_unlink`(1M) command.
`domain_unlink: Failed to acquire global lock (bringup_dr.lock).` `Possibly critical host operation or IDN operation in-progress.` `Retry domain_unlink(1M).`	A host operation was in progress (for example, `bringup`(1M), DR, EDD, or IDN).	Retry the `domain_unlink`(1M) command after the host operation has completed successfully.

General Host Errors

The following table contains the general host errors.

TABLE B-3 General Host Errors Recorded on the SSP

Error	Probable Cause	Suggested Action
`domain_link: ...Checking IDN state of` [*domain_name*] `: ARBSTOP` `Cannot link with domains that are arbstopped.`	The domain, *domain_name*, has arbitrarily stopped (arbstopped). If a domain arbitrarily stops, the event-detection daemon (EDD) reboots the domain and relinks the IDN member domains.	Wait for the EDD recovery script to complete successfully. The EDD script relinks the IDN member domains as part of its recovery routine. If the EDD is not running, reboot the domain manually.

TABLE B-3 General Host Errors Recorded on the SSP *(Continued)*

Error	Probable Cause	Suggested Action
`domain_link: IDN LINK operation unsuccessful [`*domain_name_a*` + `*domain_name_b*`] domain_link: (IDNKERR 0x112) IDN_NWR_SIZE conflicts (expected `*XX*` MB, actual `*XX*` MB) with domain id `*domain_ID*` domain_link: Unlinking domains. Retry domain_link(1M).`	The values of the IDN parameter `idn_nwr_size` for the specified domains do not match. The domains are automatically relinked to the IDN in which they were members.	Set the values of the `idn_nwr_size` variable on the specified domain(s) to the same value. Reboot the domain(s) so that the variable(s) take affect, then retry the `domain_link`(1M) command.
`domain_link: IDN LINK operation unsuccessful (link `*domain_name_a*` + `*domain_name_b*`) domain_link: (IDNKERR 0x109) error programming hardware with respect to domain id `*domain_ID*`. Retry domain_link(1M).`	The SSP encountered a kernel error `IDNKERR_HW_ERROR` (`0x109`) on the host domain.	Examine the domain-specific error for more information about this error.
`domain_link: WARNING: Some domains possibly failed to LINK: domain_link: ...Failed to LINK: `*domain_name*` domain_link: domain [`*domain_name*`] being AWOL domain_link: IDN LINK operation unsuccessful ([`*domain_name_a*` + `*domain_name_b*`] Retry domain_link(1M).`	The `domain_link`(1M) operation timed out waiting for one of the domains, *domain_name*, to link successfully to the IDN. The domain was in an unknown state (AWOL).	Check the platform log for an AWOL event. If one is present, wait for it to unlink the AWOL domains. Then, retry the `domain_link`(1M) command.
`domain_unlink: domain [`*domain_name*`] being AWOL domain_unlink: IDN UNLINK operation unsuccessful [`*domain_name*`] Retry domain_unlink(1M).`	The IDN driver failed to unlink the domain.	Check the platform log for an AWOL IDN event. If one is present, wait for the event to unlink the AWOL domain. Then, retry the `domain_unlink`(1M) command.

TABLE B-3 General Host Errors Recorded on the SSP *(Continued)*

Error	Probable Cause	Suggested Action
`domain_unlink: domain` *domain_name* `boardset (0x8001) conflicts with MIB (0x1) May need to force (-f) domain_unlink: board configuration conflicts with expected value [`*domain_name*`]`	The physical board set of the domain is inconsistent with the set in the MIB.	Ensure that the domain name is correct, then retry the `domain_unlink`(1M) command. If necessary, use the soft force option, `-f`, with the `domain_unlink`(1M) command.
`domain_unlink: ...Checking IDN state of` *domain_name_a* `: UNKNOWN domain_link: ...Checking IDN state of` *domain_name_b* `: DOWN domain_unlink: ...Checking IDN state of` *domain_name_c* `: DOWN domain_unlink: Verifying IDN UNLINK... Error accessing sigblock fields in bbsram (unlink` *domain_name_a*`)`	The SSP is unable to read the signature block of the domain.	Ensure that the domain is running. If not, reboot the domain. Then, retry the `domain_unlink`(1M) command.
`domain_unlink: IDN UNLINK operation unsuccessful [`*domain_name_a* `-` *domain_name_b*`] Retry domain_unlink(1M). domain_unlink: Failed to resolve IDN linkage and unlink domain (`*domain_name*`) May need to retry or force (-f) domain_unlink: IDN UNLINK operation unsuccessful [`*domain_name*`] Retry domain_unlink(1M).`	The linkage is inconsistent between the specified domain, domain_name, and the other IDN member domains.	Use the soft force option, `-f`, to unlink the domain.

IDN-Related Command Errors

The following table contains command errors that are related to IDN operations.

TABLE B-4 IDN-Related Command Errors Recorded on the SSP

Error	Probable Cause	Suggested Action
`domain_remove: Domain` '*domain_name*' `is linked to an IDN. Domain must be unlinked before it can be removed. See domain_unlink(1M).`	You cannot remove a domain that is a member of an IDN.	Unlink the domain before you use the `domain_remove`(1M) command.
`error: domain` *domain_name* `is an IDN member, cannot disable system board` *system_board_number*`. Powering off a system board of an IDN domain may result in a cluster arbstop. Unlink the domain before power it off. See domain_unlink(1M).`	You cannot power off system boards within a domain that is a member of an IDN.	Unlink the domain from the IDN. If necessary, use the force option, `-f`, with the `power`(1M) command. Use caution with the force option, it could cause a cluster arbstop.
`error: domain` *domain_name_a* `is an IDN member, cannot disable system board` *system_board_number*`.` `error: domain` *domain_name_b* `is an IDN member, cannot disable system board` *system_board_number*`. Powering off a system board of an IDN domain may result in a cluster arbstop. Unlink the domain before power it off. See domain_unlink(1M).`	These errors occur when you try to power off a bulk power supply on a system board in a domain that is a member of an IDN.	Unlink the domain from the IDN. If necessary, use the force option, `-f`, with the `power`(1M) command. Use caution with the force option, it could cause a cluster arbstop.

TABLE B-4 IDN-Related Command Errors Recorded on the SSP *(Continued)*

Error	Probable Cause	Suggested Action
`ERROR: The domain is a member of an IDN. You must unlink the domain before executing this command. See domain_unlink(1M).`	Do not use the `sigbcmd`(1M) command to panic a domain that is a member of an IDN.	Unlink the domain from the IDN before you use the `sigbcmd`(1M) command or before you use the `sync` command in the OBP environment. If the domain is hung, use the force option, `-f`, to force the panic. Use caution in forcing the panic of a domain that is a member of an IDN. It could cause a cluster arbstop.

Glossary

This contains definitions of abbreviations, words, and phrases that are used in the *Sun Enterprise 10000 InterDomain Networks User Guide*.

A

arbstop	A condition in which all ASICs for the given domain cease arbitration for system buses, thus terminating all hardware transactions. Usually, arbstop errors occur when the ASICs detect hardware anomalies such as hardware parity errors or dropped transactions.
AWOL	(absent without leave) When a domain is an unknown state (for example, halted or hung) or when it is in a non-responsive state with respect to IDN requests, it is referred to as being AWOL. If an IDN member domain detects that another IDN member domain is AWOL, that domain sends a warning message to its console and system log. The message indicates only that the domain failed to respond to an IDN message. It does not necessarily indicate that the domain is *hung*. Typically, an AWOL domain is non-responsive when it stops accepting remote logins or `ping`(1M) operations. After a domain has been reported as being AWOL, a recovery event occurs on the SSP to resolve the situation, provided that the Event Detector Daemon has been enabled (see `edd`(1M)). A message is logged on the SSP in the SSP-specific system log files indicating the occurence of the event.

B

backplane	(or centerplane) A hardware component that controls the flow of data to and from the system boards that are connected to it.
boardmask	16-bit mask with each bit representing a system board in the Sun Enterprise 10000 server.
bootbus SRAM (BBSRAM)	256-Kbyte static RAM attached to each processor PC ASIC. The BBSRAM can be accessed through the PC for reading and writing by using JTAG or the processor. BBSRAM is downloaded when `hpost`(1M) or the OpenBoot PROM start up code is executed. It provides shared data between the downloaded code and the SSP.

C

CBE	Control Board Executive.
CFG message	A cross-domain IDN message exchanged during domain linking. The message contains IDN software and hardware configuration information.
CIC	Coherency Interface Controller.
CIC prep buffer	A staging area within the CIC hardware for data targeted for being written to certain CIC registers.
cluster arbstop	An arbstop condition that involves the set of domains in an IDN. See also *arbstop*.
CMD message	A cross-domain IDN message used by IDN member domains to make certain IDN requests, such as SMR slab allocations and domain name, after they are connected to the IDN.
CON message	A cross-domain IDN message that is exchanged during domain linking to synchronize the connection of an incoming domain with other existing IDN member domains.

D

DLPI	(Data Link Provider Interface) A standard defined by the UNIX® International OSI Work Group. DLPI defines the format that STREAMS messages must take when interfacing to the datalink layer.
DMV	(Databearing Mondo Vector) A subsystem used to exchange control messages between IDN member domains.
domain ID	A unique numeric value that is chosen by the IDN driver and used to identify IDN member domains. This value is based on physical attributes of the domains and is guaranteed to be unique across the entire Sun Enterprise 10000 server. The values range from 0 to 15.
DR	(Dynamic Reconfiguration) A software feature that enables you to logically attach and detach system boards to and from the operating system without causing machine downtime.
dynamic system domain	System boards that have been logically grouped together into separate bootable operating environments

E

Ethernet address	The machine address used by the IDN network software to uniquely identify domains in the Sun Enterprise 10000 server.
edd	(event-detector daemon) Initiates event monitoring on the Sun Enterprise 10000 server control board.
errno	UNIX error number (see the Intro(2) man page).

F

FIN message	A cross-domain IDN message that is exchanged during domain unlinking to synchronize the disconnect of an outgoing domain with other existing IDN member domains.

H

header cookie	A unique value that is defined in each SMR mailbox header. The header cookie value uniquely identifies the mailbox header and provides a means for detecting possible data corruption within SMR mailboxes.

I

IDN	InterDomain Network.
IDN info operation	(SSI_INFO) An SSP based sigblock mailbox operation performed by SSP-based IDN commands to query IDN information from the host-side IDN driver.
IP addresses	(Internet Protocol) See the *TCP/IP and Data Communications Administration Guide* for more information.

L

logical interfaces	The IDN driver is composed of multiple instances with each instance representing a separate logical network interface. Each logical interface can serve as a separate IP subnet.
lsb/msb	Least Significant Bits / Most Significant Bits.

M

MAC header	The machine address portion of the Ethernet header that contains the Ethernet address. For the IDN feature, this address is used to uniquely identify the target domain for an IP datagram.
mailbox	(In the context of the IDN feature) Represents the point-to-point interdomain mailboxes that reside in the SMR. They are used to transmit IDN data packets between IDN member domains. See also *SMR*.

master domain	The IDN member domain that contains the physical SMR. The master domain exports the SMR to the slave domains. Slave domains have a logical SMR that maps to the physical SMR of the master domain. You can determine which IDN member domain is the master by using the `ndd`(1M) parameter `idn_global` for the IDN driver. See also *slave domain*.
MCADR	Memory Controller Address Decoding Register.
MTU	Maximum Transfer Unit.

N

netcon	Network console (see the `netcon`(1M) man page).
NEGO message	A cross-domain IDN message that is exchanged to initiate domain linking and to negotiate which domain is, or becomes, the master domain. See also *master domain*.
NetWork Region	The portion of the SMR that is actually used by the IDN driver for data packet (TCP/IP) communication between domains.

O

OBP	OpenBoot™ PROM.

P

plumb	In this guide, plumb means to configure the network by using the `ifconfig`(1M) command.
post	power-on self-test.
POST2OBP data structure	The data structure created by POST. which describes the physical components of the system. Resides in BBSRAM.

S

signature block (BBSRAM)	(or sigblock) See *bootbus SRAM (BBSRAM)*.
shared memory domain registers	Allow a message to be forwarded to a particular system board. The shared memory domain registers are located on the interconnect.
shared memory mask registers	Allow an incoming message to be accepted from a particular system board. The shared memory mask registers are located on the system board.
slab	A unit of allocation of SMR space. Slabs represents an array of fixed-sized SMR buffers to be used for IDN data packets.
slab pool	A structure that is managed by the master domain to administer allocations of the SMR to the slave domains.
slave domain	An IDN member domain that *imports* a logical mapping of the physical SMR from the master domain.
SMR	Shared Memory Region.
snmpd	The SNMP proxy agent listens to a UDP port for incoming requests and services the group of objects specified in `Ultra-Enterprise-1000.mib`. See `snmpd`(1M).
SSP	System Service Processor enables you to monitor and control the Sun Enterprise 10000 server.
STREAMS	A kernel mechanism that supports development of network services and data communications drivers. STREAMS defines interface standards for character input/output within the kernel, and between the kernel and user level. The STREAMS mechanism comprises integral functions, utility routines, kernel facilities, and a set of structures.
subnet	Local networks with large numbers of hosts are sometimes divided into subnets. See the *TCP/IP and Data Communications Administration Guide* for more information.

U

unplumb	In this guide, unplumb refers to the action of deconfiguring the network.

V

vote ticket	A 32-bit quantity that is exchanged by IDN member domains during linking to determine which domain is or will become the master domain. The value is guaranteed to be unique across the entire Sun Enterprise 10000 server.

Index