

Manpages

This Appendix contains the following manpages:

- `u4ftdog` - Netra ft 1800 OSdog driver
- `u4ftdogpat` - user OSdog patting daemon
- `split` - introduction to Split mode
- `splitadm` - to split or merge domains
- `splitconf` - change the split Master or the ownership of a resource
- `splitd.conf` - configuration file for u4ftsplitted system split daemon
- `splitinfo` - displays domain-related information

NAME

u4ftdog - Netra ft 1800 OSdog driver

SYNOPSIS

u4ftdog@*bus_address:port_name*

DESCRIPTION

The **u4ftdog** driver is a multi-threaded, multi-instance, non-STREAMS driver that is responsible for controlling the operating system watchdog. It exports an **ioctl(2)** interface that enables applications to register their ability to report system availability. Once registered, the application is responsible for periodically reporting (via the **ioctl(2)** interface) that the system is available. Failure to do this is classed as a system failure and the **u4ftdog** driver then initiates a system reboot via the **cmn_err(9F)** CE_PANIC function.

The **u4ftdog** driver also allows control of the Netra ft 1800 OSdog hardware, which provides a hard failure recovery mechanism. Failure to regularly pat the OSdog hardware causes a system power cycle/reset to be initiated.

HARDWARE INTERFACE

The Netra ft 1800 hardware OSdog functionality is provided by a number of registers on each motherboard. These registers implement a timer count upwards. In normal system operation this counter is reset periodically by **u4ftdog** driver software writing to the "pat" register. If the register is not patted within a specified time limit, the counter reaches a trigger value, which causes the motherboard hardware to initiate an OSdog timeout.

The actions associated with an OSdog timeout are dependent upon the systems operating mode. In a combined system, the OSdog timeout causes a system power cycle of the side whose OSdog timed out. In split mode, the OSdog timeout causes a cpuset reset on the OSdog's side. The differences in actions are due to the possibility of resource sharing in a split system where one side may be using resources on the other side. If the side was power cycled, the other side's resources would be reset.

The OSdog time has two possible timeout values: normal (10.74s), which is generally selected when Solaris is running, and extended (85.90s), which is generally selected when system firmware is running.

OVERVIEW

The **u4ftdog** driver comprises three instances; a control instance and two hardware instances. The control instance represents the upper layer of the driver. It handles the user patting ioctl interface and performs the patting of the hardware OSdogs.

The hardware instances represents the OSdog hardware on each motherboard and allows the hardware to be brought online/taken offline in response to cpuset integration and out-of-sync events. The hardware instances have private interfaces, which are not intended for end-user use.

The driver performs continuous latent fault checking on the OSdog hardware once the OSdog has been enabled. The driver checks monitor bits in the hardware registers and ensures that they are pulsing.

CONFIGURATION

The **u4ftdog** driver provides a hardware configuration file, **u4ftdog.conf** which contains various properties that are used during driver initialization. These properties are user configurable and are presented below.

name - a string property that identifies the driver. It is the same for all instances.

parent - a string that identifies the parent of a particular instance. This property encodes the absolute path of the parent node. For the control instance, this property is denoted as **pseudo** since it is not responsible for driving any hardware.

node - a single character string property that defines the instance of the current node. "A" and "B" represent the OSdog hardware on the respective sides and "C" represents the control instance.

reg - a 5-tuple integer property that describes the address space that will be mapped by the hardware instances.

The following configuration file properties apply to all three **u4ftdog** instances:

channel_x_limit - a number of integer properties (one per channel - replace x by the channel number) that describe initial limits (in seconds) for each pat channel. These properties are used to provide OSdog protection during initial boot. They should be set to a value that represents the **no-config** - an integer property that tells the framework that the hardware represented is non self-identifying and will provide a configuration file.

non-prom - an integer property that tells the framework that the OBP does not recognize the device, i.e., it is not a hardware node.

u4ft-aware - an integer property that tells the framework that the instances will provide their own routines to process the online/offline state change requests.

trace - a string property that encodes the trace flags that cause certain trace messages to be output to the message log.

IOCTLS

The **u4ftdog** driver provides ioctls that enable applications to configure/perform user OSdog patting and ioctls that allow system management functions to configure the OSdog. Note that the latter should not be used by users and are not described below.

The following system calls are not supported by the **u4ftdog** driver:

read(2), write(2), mmap(2).

During initialization, the control instance creates a number of device nodes (one per pat channel) - **/dev/u4ftdog:pat?** where '?' is the pat channel number from 1 to 8.

In addition, each instance creates a device node **/dev/u4ftdog:[ABC]**, which is used by system control functions. These devices should not be used by users.

USER PAT CHANNEL IOCTLS

The user pat channels allow the user to define a custom system hang detection scheme which monitors aspects of the system appropriate to their needs (see **u4ftdogpat(1M)** for a suitable example).

By default, the user pat channels are active only when a process has the channel device special file open and has configured the limit of the channel. This provides a safety net against problems with the monitoring application that cause it to terminate abnormally and thus cause the OSdog to trigger. This behavior can be overridden using the **LINGER_ON_CLOSE** ioctl described below.

A typical use of a user pat channel is as follows: open the channel device, set the timeout limit to an appropriate value, loop around monitoring for hangs and patting the OSdog if the system is not hung (you must pat more frequently than the limit period), and if you wish to close the channel first set the limit to zero (disable the channel) then close the channel device.

The following ioctls are applicable to the pat channel devices.

U4FTIOC OSDOG SET LIMIT

enable / disable a user patting channel. The argument is a pointer to the a **uint32_t** which represents the desired limit (in seconds) for that pat channel. A pat channel is enabled when a non-zero limit is specified and disabled when a zero limit is specified.

Note that care should be taken when selecting a limit to ensure that the limit is appropriate for the expected pat process. In particular, small values (less than 10) are generally not appropriate.

U4FTIOC OSDOG PAT

Pat the channel to inform the **u4ftdog** driver that the patting process is active and that the driver should reset the pat channel's OSdog timer. The ioctl takes no argument (supply a NULL pointer).

The patting must be performed often enough to ensure that the OSdog does not go off. Take care to ensure that any scheduling delays (if permissible) are accounted for.

U4FTIOC OSDOG LINGER ON CLOSE

Specify that the driver should not automatically disable the pat channel if the channel is closed. This is provided to allow system shutdown operations to be OSdog protected.

The ioctl takes a **uint32_t parameter** whose value is 1 to enable linger-on-close and zero to disable it.

Note that if a device is closed and then re-opened, the linger-on-close attribute will be disabled.

ERRORS

An open() will fail if:

ENXIO

The driver is not installed in the system.

EINVAL

The device minor being opened is the wrong type.

An ioctl() will fail if:

EFAULT

A bad user-space address was specified.

EINVAL

A non-existent control command was requested or invalid parameters were supplied.

EIO

An I/O error occurred.

ENXIO

The driver is not installed in the system or the required hardware instance(s) were not online.

FILES

/platform/SUNW,Ultra-4FT/kernel/drv/u4ftdog # device driver module.

/platform/SUNW,Ultra-4FT/kernel/drv/u4ftdog.conf # hardware configuration file.

/dev/u4ftdog:A # side A hardware device /dev/u4ftdog:B # side B hardware device /dev/u4ftdog:C # generic control device /dev/u4ftdog:pat? # pat channel devices

SEE ALSO

u4ftdogpat(1M), cmn_err(9F), driver.conf(4),

NAME

u4ftdogpat - user OSdog patting daemon

SYNOPSIS

u4ftdogpat **-p** *period* **-l** *limit* **-c** *channel* [**-i**] [**-o** *monitoring options*]

DESCRIPTION

The **u4ftdogpat** daemon provides a software operating system watchdog (OSdog). It communicates with the **u4ftdog**(7D) driver periodically to pat a specific OSdog channel, thus demonstrating that the system is running processes. If a channel is not patted within a specified time limit, the **u4ftdog** driver initiates a operating system panic (see **cmn_err**(9F)) causing the machine to reboot. The panic dump can then be analyzed at a later point in time to determine the cause of the panic.

OPTIONS

-c *channel*

The software OSdog channel to use to protect the system. The **u4ftdog**(7D) driver provides a number of different channels enabling different hang conditions to be monitored.

-i Specify that the daemon should not detach itself from the controlling terminal. This is generally used to allow the daemon to be run as a "respawn" entry in /etc/inittab (see **inittab**(4)).

-l *limit*

The time (in seconds) that the **u4ftdog** driver will wait for a pat command before concluding that the system is hung and, hence, invoking an operating system panic.

-p *period*

Used to specify the period between OSdog pats. This is the time (in seconds) that the daemon waits before issuing a pat command to the **u4ftdog**(7D) driver.

monitoring options

Specify monitoring options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

bind=*processor_id*

This option specifies the *processor_id* (see **processor_bind**(2)) that the daemon process should be bound to. This allows the daemon to monitor a particular processor for hangs. By default, the daemon inherits its parent's binding.

bindset=*pset*

This option specifies the processor set (see **pset_bind**(2)) that the daemon process should be bound to. This allows the daemon to monitor a particular processor set for hangs. By default, the daemon inherits its parent's binding.

RT=*priority*

This option specifies the priority (see **priocntl2**) of the patting daemon as being in the real-time class with the specified numeric priority.

TS=*priority*

This option specifies the priority (see **priocntl2**) of the patting daemon as being in the timesharing class with the specified numeric priority.

unlock

Specify that the daemon's address space should not be locked down in memory (see **mlockall**(3C)). By default, the daemon's address space is locked down in memory.

EXAMPLES

The following example uses software OSdog channel 1 to pat the OSdog every 5 seconds with a time limit of 20 seconds before the OSdog is triggered. All scheduling priorities and processor bindings are inherited from the invoking process.

```
u4ftdogpat -c 1 -p 5 -l 20
```

The following example uses software OSdog channel 4 to pat the OSdog every 5 seconds with a limit of 10 seconds. The process runs with real-time priority 59.

```
u4ftdogpat -c 4 -p 5 -l 10 -o RT=59
```

The following example uses software OSdog channel 1 to pat the OSdog every 10 seconds with a limit of 60 seconds. The daemon does not detach from the controlling terminal. This example is suitable for use in a /etc/inittab entry.

```
u4ftdogpat -c 1 -p 10 -l 60 -i
```

SEE ALSO

u4ftdog(7D), **cmn_err(9F)**, **inittab(4)**, **mlockall(3C)**, **priocntl(2)**, **processor_bind(2)**, **pset_bind(2)**.

NAME

split – introduction to Split mode

DESCRIPTION

The Netra ft 1800 fault-tolerant platform is capable of running in two modes: as a single fault-tolerant system (combined mode), and as two domains, each acting as an independent system (split mode). The split-mode support software manages the orderly transition from one mode of operation to the other and coordinates the proper transfer of system resources between the domains.

When the system is in combined mode, the system is composed of a single domain that encompasses all the system resources. In combined mode, the system has a *combined* attribute alluding to its fault-tolerant nature. In split mode, each independent domain has a *split* attribute implying that it is not part of a fault-tolerant system.

In split mode, each domain contains at least a CPU, MBD, CAF and PSUs, and, possibly, some other modules. Each module is owned by one domain. Ownership can be changed, but can never be shared. Certain modules (CAF, RMM, MBD, CPU, and PSUs) are considered fixed (unmovable) and are statically owned by their "natural" domain when the system is in split mode. All other modules are movable and can be freely assigned to either domain. When the system is in combined mode, the assignment of ownership of modules is meant as a preparatory step for the system to transition to split mode. Therefore, in combined mode the assignment of ownership of a resource to a domain that will exist only after the system is split is taken as potential future ownership of that resource should the system transition from combined mode to split mode.

One domain is defined as the split master. While the system is in combined mode, the split master refers to the future master should the system become split. While the system is in split mode, the split master can always initiate the transfer of ownership of a module from one domain to another. Other domains can initiate a transfer of ownership only if they can successfully negotiate such a transfer with the split master. Note that the split master can be changed by the user, both in combined and split modes.

Mode (or attribute), split master, and module ownership persist across the reboot of any domain.

USER INTERFACE

The split mode support software provides both a command line interface and an application programming interface (API). Both the commands and the API enable management of the mode and resources as well as enquiry functions to get information about the current mode, domain, mastership and ownership of resources.

Commands

```
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/{splitinfo,splitadm,splitconf}
```

The split library

```
/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib/{libu4ftsplit.so,libu4ftsplitmt.so}
```

The split API is defined in `/usr/platform/SUNW,Ultra-4FT/SUNWcms/include/split_api.h`

SPLIT SUPPORT

The software subsystem that implements split mode support consists of:

The split daemon

u4ftsplitd, which executes on each system while split and on the combined system when in fault tolerant mode. It is the daemon that controls all the domain attribute operations and resource ownership. One copy of the daemon runs on each domain of a multi-domain system and the daemons communicate with each other using sockets. The daemons maintain the domain and ownership information for all the resources in the system. This information is kept consistently in several locations. There are in-memory and persistent copies.

The InterCpuset Network driver

icn, used in split mode as the basis of communication between the two daemons.

SEE ALSO

splitadm(1M), splitconf(1M), splitinfo(1M), set_slot_owner(3), set_domain_attribute(3), get_slot_status(3), get_domain_attributes(3), split_lock(3), split_unlock(3), splitd.conf(4)

NAME

`splitadm` – to split or merge domains

SYNOPSIS

```
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitadm [ -f ] [ -t timeout ] [ -w domain ]  
[ -d domain... ] split -w domain  
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitadm [ -d domain... ] merge
```

DESCRIPTION

splitadm is used to split a domain into a number of independent sub-domains, or to merge a number of sub-domains into a single domain.

All the domain names used must be valid and specified in **splitd.conf**.

This command must be executed with super-user privileges.

Split a domain

```
splitadm [ -f ] [ -t timeout ] [ -w domain ] [ -d domain... ] split
```

When a domain splits, a set of independent sub-domains is created, each one owning a subset of the original domain's resources. One sub-domain (called the winner or the surviving domain) may have the system identity of the original domain and may keep running the existing processes. All other sub-domains are given a new system identity.

The following options may be supported for splitting:

-t *timeout*

overrides the default timeout (see **splitd.conf(4)**) within which all the implied (if any) operations on the resources involved in this transition must complete. *timeout* must be expressed in seconds.

-f enforces the success of the command even if some operations on the involved resources fail or do not complete within the timeout.

-w *domain*

specifies the winner domain, which inherits the system identity of the original domain.

-d *domain...*

specifies a list of resulting sub-domains.

The configuration, as reported by **splitinfo(1M)** before splitting, is realized if the command succeeds.

Merge a set of domains

```
splitadm -w domain [ -d domain... ] merge
```

This command merges a set of domains into a single one, which owns all the resources of the original sub-domains. The resulting domain inherits the system identity of one of the original sub-domains (the winner) and keeps running the existing services; all other merged domains are stopped and their resources added to the surviving domain.

The following options may be supported for merging:

-w *domain*

specifies the winner domain.

-d *domain...*

specifies a list of sub-domains to merge together.

Netra ft 1800

On this platform, only two system configurations are supported: one single domain, or two independent domains. When the system is operating in combined mode, a single domain exists called

C. When the system is operating in split mode, two sub-domains exist **A** and **B**. Furthermore, the system identity of the combined domain will be the same as the winner sub-domain.

The only valid domain names, specified in **splitd.conf**, are **A**, **B** and **C**.

On this platform, the **-d** option is ignored since the only valid subdomains are **A** and **B**.

Splitadm(1M) is responsible for the mode transitions on this platform.

Splitting

When the combined domain splits, two independent sub-domains are created (**A** and **B**), each one owning a subset of the total system resources. One domain (the winner or surviving) has the system identity of the combined domain and keeps running the existing processes. The other domain can be rebooted by the user, and will come up with a new system identity.

At first the command tries to unconfigure some modules in the locations belonging to the new domain. The command fails if any disable process fails or does not complete within a certain timeout (unless the **-f** option is used).

If the option **-w** is not specified, the winner (surviving) domain defaults to the split master, as reported by **splitinfo -m**.

Merging

The merge operation moves the system from split to combined mode: the losing domain is reset and its CPU is reintegrated in the combined domain. The specified winner (surviving) sub-domain must be **A** or **B**.

EXAMPLES

Split a domain, surviving domain is split master:

```
example# splitadm split
domain = A
attributes = split,master
master = A
```

Split a domain, surviving domain is B:

```
example# splitadm -w B split
domain = B
attributes = split
master = A
```

Split a domain, surviving domain is A (fails because B-CAF cannot be disabled):

```
example# splitadm -t 30 -w A split
Error: cannot disable B-CAF, operation initiated but not completed
```

Split a split domain:

```
example# splitadm split
already split
```

Merge a domain (surviving domain will be A):

```
example# splitadm -w A merge
domain = C
attributes = combined
master = B
```

Merge a combined domain:

```
# splitadm -w A merge
already merged
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltu
Interface Stability	Unstable

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

splitconf(1M), **splitinfo(1M)**, **splitd.conf(4)**, **split(5)**

NAME

`splitconf` – change the split Master or the ownership of a resource

SYNOPSIS

```
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitconf  -m master_domain  
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitconf  [ -f ] [ -t timeout ]  
-l location -o owner_domain
```

DESCRIPTION

splitconf is responsible for controlled changes in resources ownership and split mastership.

Any domain is allowed to change the split master with no restrictions.

Resource ownership changes can be initiated from any domain. All the domains involved in the operation should have the ability to communicate each other (see **split(5)**) in order to negotiate and realize the change, and to keep the same view of the system configuration.

If, for any reason, the communication between the domains involved in the transition stops, only the split master is allowed to initiate a configuration change.

This command must be executed with super-user privileges.

Assign the split master

```
splitconf -m master_domain
```

Elects *master_domain* as the split master.

master_domain must be a valid domain name, listed in **splitd.conf**.

Assign ownership to a location

```
splitconf [ -f ] [ -t timeout ] -l location -o owner_domain
```

Assign the ownership of the resource in *location* to *owner_domain*.

The following options are supported:

-t *timeout*

overrides the default timeout (see **splitd.conf(4)**) within which all the implied operations (if any) on the resource involved must complete. *timeout* must be expressed in seconds.

-f enforces the success of the command, even if some operations on the involved resource fail or do not complete within the timeout.

-l *location*

specifies the location of the resource whose ownership is changed.

-o *owner_domain*

specifies the domain to which the resource must be assigned. A sub-domain may be specified, before splitting a domain, in order to prepare a target configuration before a split operation (see **splitadm(1M)** for details).

If the timeout expires before the required operations (if any) on the resource are completed, or if some of the operations fail, the command fails and the ownership of the resource is not altered. Otherwise, on success, *location* is assigned to *owner_domain*.

The command always refuses to change owner of Fixed locations.

Netra ft 1800

When in combined mode, the command simply assigns *location* to *owner_domain* (**A** or **B**). The options **-f** and **-t** are ignored.

When in split mode, the command tries to unconfigure the module in *location* before transferring ownership.

On this platform, the allowed resource locations are specified in **cms_location(5)**.

EXAMPLES

Assign mastership to B:

```
example# splitconf -m B
master = B
```

Assign A-PCI2 to B:

```
example# splitconf -l A-PCI2 -o B
A-PCI2: B Movable
```

Assign A-CAF to B (fails because A-CAF is a fixed location):

```
example# splitconf -l A-CAF -o B
splitconf: Error: A-CAF Fixed, cannot change owner to a Fixed slot
```

Assign A-PCI2 to B (fails because A-PCI2 didn't disable within the timeout):

```
example# splitconf -t 10 -l A-PCI2 -o B
splitconf: Error: cannot change owner, split daemon operation timeout
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltu
Interface Stability	Unstable

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

splitinfo(1M), **splitadm(1M)**, **splitd.conf(4)**, **split(5)**, **cms_location(5)**

NAME

splitd.conf – configuration file for u4ftsplitd system split daemon

SYNOPSIS

/etc/splitd.conf

DESCRIPTION

The file **/etc/splitd.conf** contains information used by the split daemon, **u4ftsplitd**. This information is used to initialize the daemon's operation timeout, enable logging, and initialize the IP addresses and port numbers used by the daemons running on multiple domains.

The file can contain an unlimited number of one-line entries. Each entry can consist of up to two space-separated fields:

key value

The recognized keys and their values are as follows:

domain

A character string with the name of a domain. This is used to check for validity of domain operations.

host_prin

Two host names, one for each domain, representing the interface used by the two split daemons to exchange messages with each other. On a Netra ft 1800 system, it is recommended that you specify icn interfaces here.

host_alt

Two alternative host names, one for each domain, representing the interface used by the two split daemons to exchange messages with each other in cases when the primary interface defined in **host_prin** malfunctions.

port_address

A character string that is composed of a 4-tuple separated by space. The 4-tuple appear in the following order: source_domain, which is the name of the domain that will act as the sender for this port, destination_domain, the port_number, and a flag that indicates if this is a bidirectional port or not. Note that several ports may be specified for each pair of domains to be used by the split daemons on various domains to communicate with each other. On a Netra ft 1800 system the split daemon uses unidirectional ports and binds itself to the first available one in the configuration file.

dolog

Used to enable split daemon logging. The value is *yes* or *no*.

timeout

The value is an integer which is the operation timeout in seconds.

hostnames

The two host names for the domains of a split system. These host names should be included in the */etc/hosts* file or available from NIS.

There must be only one identical copy of this file, i.e., split daemons on all domains use identical configuration files. The daemons obtain the initial configuration parameters from the configuration file **splitd.conf**. The value of *timeout*, if specified by a command or API option, overrides the default value.

EXAMPLE

In this example, a Netra fit 1800 system called *treeton* can be split into two nonfault-tolerant systems called *treeton* and *treeton-2*. The two split daemons running on each side use an *icn* interface as their primary communications path. An alternative communication path is defined on an Ethernet interface.

```
# The domain-address section.
# Add the primary and alternative host names
# (used for inter-daemon comms) of the two sides here.
# Example:
# host_prim host-i0 host-2-i0
# host_alt thishost thishost-2
host_prim treeton-i0 treeton-2-i0
host_alt treeton treeton-2

# The port-number section.
port-address a b 3501
port-address b a 3502
port-address a b 3502
port-address b a 3503

# The misc-section
timeout 60
dolog yes

#The hostnames section.
# Add the host names of the two sides here.
# Example:
# hostnames thishost thishost-2
#host_alt thishost thishost-2
hostnames treeton treeton-2
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltr
Interface Stability	Unstable

NAME

splitinfo – displays domain-related information

SYNOPSIS

```
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/splitinfo [ -d ] [ -a ] [ -m ] [ -l location... ]  
[ -o domain ]
```

DESCRIPTION

This command retrieves information concerning domains and their attributes, as well as the resources on the system.

OPTIONS

The following options are supported:

-d prints the current domain name.

-a prints a list of attributes associated with the current domain.

-m prints the current split master.

-l *location...*

prints the owner and status (Fixed or Movable) of all *location* in the list. Use **-l all** to print the info on all the locations in the system.

-o *domain*

prints the resources (list of the locations) owned by *domain*.

splitinfo with no arguments does **-dam -lall**.

Netra ft 1800

On this platform, in combined mode domain **C** owns all the resources. Therefore, **splitinfo** always returns as the owner of a given resource the (actual or prospective) sub-domain **A** or **B**.

The allowed resource locations are specified in **cms_location(5)**

EXAMPLES

Print the current domain:

```
example% splitinfo -d  
domain = A
```

Print the owner (actual or perspective) and status of A-PCI3:

```
example% splitinfo -l A-PCI3  
A-PCI3: A Movable
```

Print the current split Master:

```
example% splitinfo -m  
master = B
```

Print the current domain, attributes and Master:

```
example% splitinfo -dam  
domain = B
```

```
attributes = split, master
master = B
```

Print all the locations currently owned by A:

```
example% splitinfo -o A
A-PCI0
A-PCI7
A-CAF
A-DSK
A-RMM
A-CPU
A-MBD
A-PSU0
A-PSU1
A-PSU2
B-PCI0
B-PCI1
```

Print a complete split information:

```
example% splitinfo
domain = A
attributes = split
master = B
A-PCI0: A Movable
A-PCI1: A Movable
A-PCI2: A Movable
.
.
.
B-PSU1: B Fixed
B-PSU2: B Fixed
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWspltu
Interface Stability	Unstable

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

splitadm(1M), splitconf(1M), split(5), cms_location(5)