System Administration Guide: IP Services

Part VI IPMP

This part introduces IP network multipathing (IPMP) and contains tasks for administering IPMP. IPMP provides failure detection and failover for interfaces on a system that are attached to the same link.

Chapter 30 Introducing IPMP (Overview)

IP network multipathing (IPMP) provides physical interface failure detection and transparent network access failover for a system with multiple interfaces on the same IP link. IPMP also provides load spreading of packets for systems with multiple interfaces.

This chapter contains the following information:

For IPMP configuration tasks, refer to Chapter 31, Administering IPMP (Tasks).

Why You Should Use IPMP

IPMP provides increased reliability, availability, and network performance for systems with multiple physical interfaces. Occasionally, a physical interface or the networking hardware attached to that interface might fail or require maintenance. Traditionally, at that point, the system can no longer be contacted through any of the IP addresses that are associated with the failed interface. Additionally, any existing connections to the system using those IP addresses are disrupted.

By using IPMP, you can configure one or more physical interfaces into an IP multipathing group, or **IPMP group**. After configuring IPMP, the system automatically monitors the interfaces in the IPMP group for failure. If an interface in the group fails or is removed for maintenance, IPMP automatically migrates, or **fails over**, the failed interface's IP addresses. The recipient of these addresses is a functioning interface in the failed interface's IPMP group. The failover feature of IPMP preserves connectivity and prevents disruption of any existing connections. Additionally, IPMP improves overall network performance by automatically spreading out network traffic across the set of interfaces in the IPMP group. This process is called **load spreading**.

Oracle Solaris IPMP Components

Oracle Solaris IPMP involves the following software:

- The `in.mpathd` daemon, which is explained fully in the in.mpathd(1M) man page.
- The `/etc/default/mpathd` configuration file, which is also described in the `in.mpathd` (1M) man page.
- `ifconfig` options for IPMP configuration, as described in the ifconfig(1M) man page.

Multipathing Daemon, `in.mpathd`

The `in.mpathd` daemon detects interface failures, and then implements various procedures for failover and failback. After `in.mpathd` detects a failure or a repair, the daemon sends an `ioctl` to perform the failover or failback. The `ip` kernel module, which implements the `ioctl` , does the network access failover transparently and automatically.

Note –

Do not use Alternate Pathing while using IPMP on the same set of network interface cards. Likewise, you should not use IPMP while you are using Alternate Pathing. You can use Alternate Pathing and IPMP at the same time on different sets of interfaces. For more information about Alternate Pathing, refer to the *Sun Enterprise Server Alternate Pathing 2.3.1 User Guide*.

The `in.mpathd` daemon detects failures and repairs by sending out probes on all the interfaces that are part of an IPMP group. The `in.mpathd` daemon also detects failures and repairs by monitoring the `RUNNING` flag on each interface in the group. Refer to the in.mpathd(1M) man page for more information.

Note –

DHCP is not supported to manage IPMP data addresses. If you attempt to use DHCP on these addresses, DHCP eventually abandons control of these addresses. Do not use DHCP on data addresses.

IPMP Terminology and Concepts

This section introduces terms and concepts that are used throughout the IPMP chapters in this book.

IP Link

In IPMP terminology, an **IP link** is a communication facility or medium over which nodes can communicate at the data-link layer of the Internet protocol suite. Types of IP links might include simple Ethernets, bridged Ethernets, hubs, or Asynchronous Transfer Mode (ATM) networks. An IP link can have one or more IPv4 subnet numbers, and, if applicable, one or more IPv6 subnet prefixes. A subnet number or prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated local area network (LAN). With the Address Resolution Protocol (ARP), the scope of the ARP protocol is a single IP link.

Note –

Other IP-related documents, such as RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, use the term **link** instead of **IP link**. Part VI uses the term **IP link** to avoid confusion with IEEE 802. In IEEE 802, **link** refers to a single wire from an Ethernet network interface card (NIC) to an Ethernet switch.

Physical Interface

The **physical interface** provides a system's attachment to an IP link. This attachment is often implemented as a device driver and a NIC. If a system has multiple interfaces attached to the same link, you can configure IPMP to perform failover if one of the interfaces fails. For more information on physical interfaces, refer to IPMP Interface Configurations.

Network Interface Card

A **network interface card** is a network adapter that can be built in to the system. Or, the NIC can be a separate card that serves as an interface from the system to an IP link. Some NICs can have multiple physical interfaces. For example, a `qfe` NIC can have four interfaces, `qfe0` through `qfe3` , and so on.

IPMP Group

An IP multipathing group, or **IPMP** group, consists of one or more physical interfaces on the same system that are configured with the same IPMP group name. All interfaces in the IPMP group must be connected to the same IP link. The same (non-null) character string IPMP group name identifies all interfaces in the group. You can place interfaces from NICs of different speeds within the same IPMP group, as long as the NICs are of the same type. For example, you can configure the interfaces of 100-megabit Ethernet NICs and the interfaces of one gigabit Ethernet NICs in the same group. As another example, suppose you have two 100-megabit Ethernet NICs. You can configure one of the interfaces down to 10 megabits and still place the two interfaces into the same IPMP group.

You cannot place two interfaces of different media types into an IPMP group. For example, you cannot place an ATM interface in the same group as an Ethernet interface.

Failure Detection and Failover

Failure detection is the process of detecting when an interface or the path from an interface to an Internet layer device no longer works. IPMP provides systems with the ability to detect when an interface has failed. IPMP detects the following types of communication failures:

- The transmit or receive path of the interface has failed.
- The attachment of the interface to the IP link is down.
- The port on the switch does not transmit or receive packets.

- The physical interface in an IPMP group is not present at system boot.

After detecting a failure, IPMP begins failover. **Failover** is the automatic process of switching the network access from a failed interface to a functioning physical interface in the same group. Network access includes IPv4 unicast, multicast, and broadcast traffic, as well as IPv6 unicast and multicast traffic. Failover can only occur when you have configured more than one interface in the IPMP group. The failover process ensures uninterrupted access to the network.

Repair Detection and Failback

Repair detection is the process of detecting when a NIC or the path from a NIC to an Internet layer device starts operating correctly after a failure. After detecting that a NIC has been repaired, IPMP performs **failback**, the process of switching network access back to the repaired interface. Repair detection assumes that you have enabled failbacks. See Detecting Physical Interface Repairs for more information.

Target Systems

Probe-based failure detection uses **target systems** to determine the condition of an interface. Each target system must be attached to the same IP link as the members of the IPMP group. The `in.mpathd` daemon on the local system sends ICMP probe messages to each target system. The probe messages help to determine the health of each interface in the IPMP group.

For more information about target system use in probe-based failure detection, refer to Probe-Based Failure Detection.

Outbound Load Spreading

With IPMP configured, outbound network packets are spread across multiple NICs without affecting the ordering of packets. This process is known as **load spreading**. As a result of load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections.

Dynamic Reconfiguration

Dynamic reconfiguration (DR) is the ability to reconfigure a system while the system is running, with little or no impact on existing operations. Not all Sun platforms support DR. Some Sun platforms might only support DR of certain types of hardware. On platforms that support DR of NIC's, IPMP can be used to transparently fail over network access, providing uninterrupted network access to the system.

For more information on how IPMP supports DR, refer to IPMP and Dynamic Reconfiguration.

Basic Requirements of IPMP

IPMP is built into Oracle Solaris and does not require any special hardware. Any interface that is supported by Oracle Solaris can be used with IPMP. However, IPMP does impose the following requirements on your network configuration and topology:

- All interfaces in an IPMP group must have unique MAC addresses.

  Note that by default, the network interfaces on SPARC based systems all share a single MAC address. Thus, you must explicitly change the default in order to use IPMP on SPARC based systems. For more information, refer to How to Plan for an IPMP Group.

- All interfaces in an IPMP group must be of the same media type. For more information, refer to IPMP Group.

- All interfaces in an IPMP group must be on the same IP link. For more information, refer to IPMP Group.

  Note –

  Multiple IPMP groups on the same link layer (L2 or layer 2) broadcast domain are unsupported. A L2 broadcast domain typically maps to a specific subnet. Therefore, you must configure only one IPMP group per subnet.

- Depending on your failure detection requirements, you might need to either use specific types of network interfaces or configure additional IP addresses on each network interface. Refer to Link-Based Failure Detection and Probe-Based Failure Detection.

IPMP Addressing

You can configure IPMP failure detection on both IPv4 networks and dual-stack, IPv4 and IPv6 networks. Interfaces that are configured with IPMP support two types of addresses: data addresses and test addresses.

Data Addresses

Data addresses are the conventional IPv4 and IPv6 addresses that are assigned to an interface of a NIC at boot time or manually, through the `ifconfig` command. The standard IPv4 and, if applicable, IPv6 packet traffic through an interface is considered to be **data traffic**.

Test Addresses

Test addresses are IPMP-specific addresses that are used by the `in.mpathd` daemon. For an interface to use probe-based failure and repair detection, that interface must be configured with at least one test address.

Note –

You need to configure test addresses only if you want to use probe-based failure detection.

The `in.mpathd` daemon uses test addresses to exchange ICMP probes, also called **probe traffic**, with other targets on the IP link. Probe traffic helps to determine the status of the interface and its NIC, including whether an interface has failed. The probes verify that the send and receive path to the interface is working correctly.

Each interface can be configured with an IP test address. For an interface on a dual-stack network, you can configure an IPv4 test address, an IPv6 test address, or both IPv4 and IPv6 test addresses.

After an interface fails, the test addresses remain on the failed interface so that `in.mpathd` can continue to send probes to check for subsequent repair. You must specifically configure test addresses so that applications do not accidentally use them. For more information, refer to Preventing Applications From Using Test Addresses.

For more information on probe-based failure detection, refer to Probe-Based Failure Detection.

IPv4 Test Addresses

In general, you can use any IPv4 address on your subnet as a test address. IPv4 test addresses do not need to be routeable. Because IPv4 addresses are a limited resource for many sites, you might want to use non-routeable RFC 1918 private addresses as test addresses. Note that the `in.mpathd` daemon exchanges only ICMP probes with other hosts on the same subnet as the test address. If you do use RFC 1918-style test addresses, be sure to configure other systems, preferably routers, on the IP link with addresses on the appropriate RFC 1918 subnet. The `in.mpathd` daemon can then successfully exchange probes with target systems.

The IPMP examples use RFC 1918 addresses from the `192.168.0/24` network as IPv4 test addresses. For more information about RFC 1918 private addresses, refer to RFC 1918, Address Allocation for Private Internets.

To configure IPv4 test addresses, refer to the task How to Configure an IPMP Group With Multiple Interfaces.

IPv6 Test Addresses

The only valid IPv6 test address is the link-local address of a physical interface. You do not need a separate IPv6 address to serve as an IPMP test address. The IPv6 link-local address is based on the Media Access Control (MAC) address of the interface. Link-local addresses are automatically configured when the interface becomes IPv6-enabled at boot time or when the interface is manually configured through `ifconfig` .

To identify the link-local address of an interface, run the `ifconfig` *interface* command on an IPv6-enabled node. Check the output for the address that begins with the prefix `fe80` , the link-local prefix. The `NOFAILOVER` flag in the following `ifconfig` output indicates that the link-local address `fe80::a00:20ff:feb9:17fa/10` of the `hme0` interface is used as the test address.

```
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
                inet6 fe80::a00:20ff:feb9:17fa/10
```

For more information on link-local addresses, refer to Link-Local Unicast Address.

When an IPMP group has both IPv4 and IPv6 plumbed on all the group's interfaces, you do not need to configure separate IPv4 test addresses. The `in.mpathd` daemon can use the IPv6 link-local addresses as test addresses.

To create an IPv6 test address, refer to the task How to Configure an IPMP Group With Multiple Interfaces.

Preventing Applications From Using Test Addresses

After you have configured a test address, you need to ensure that this address is not used by applications. Otherwise, if the interface fails, the application is no longer reachable because test addresses do not fail over during the failover operation. To ensure that IP does not choose the test address for normal applications, mark the test address as `deprecated` .

IPv4 does not use a deprecated address as a source address for any communication, unless an application explicitly binds to the address. The `in.mpathd` daemon explicitly binds to such an address in order to send and receive probe traffic.

Because IPv6 link-local addresses are usually not present in a name service, DNS and NIS applications do not use link-local addresses for communication. Consequently, you must not mark IPv6 link-local addresses as `deprecated` .

IPv4 test addresses should not be placed in the DNS and NIS name service tables. In IPv6, link-local addresses are not normally placed in the name service tables.

IPMP Interface Configurations

An IPMP configuration typically consists of two or more physical interfaces on the same system that are attached to the same IP link. These physical interfaces might or might not be on the same NIC. The interfaces are configured as members of the same IPMP group. If the system has additional interfaces on a second IP link, you must configure these interfaces as another IPMP group.

A single interface can be configured in its own IPMP group. The single interface IPMP group has the same behavior as an IPMP group with multiple interfaces. However, failover and failback cannot occur for an IPMP group with only one interface.

You can also configure VLANs into an IPMP group by using the same steps to configure a group out of IP interfaces. For the procedures, see Configuring IPMP Groups. The same requirements that are listed in Basic Requirements of IPMP apply to configure VLANs into an IPMP group.

Caution –

The convention that is used to name VLANs might lead to errors when you configure VLANs as an IPMP group. For more details about VLAN names, see VLAN Tags and Physical Points of Attachment in *System Administration Guide: IP Services*. Consider the example of four VLANs, `bge1000`, `bge1001`, `bge2000`, and `bge2001`. IPMP implementation requires these VLANs to be grouped as follows: `bge1000` and `bge1001` belong to one group on the same VLAN 1, while `bge2000`, and `bge2001` belong to another group on the same VLAN 2. Because of VLAN names, errors such as mixing VLANs that belong to different links into an IPMP group can easily occur, for example, `bge1000` and `bge2000`.

Standby Interfaces in an IPMP Group

The **standby interface** in an IPMP group is not used for data traffic unless some other interface in the group fails. When a failure occurs, the data addresses on the failed interface migrate to the standby interface. Then, the standby interface is treated the same as other active interfaces until the failed interface is repaired. Some failovers might not choose a standby interface. Instead, these failovers might choose an active interface with fewer data addresses that are configured as UP than the standby interface.

You should configure only test addresses on a standby interface. IPMP does not permit you to add a data address to an interface that is configured through the `ifconfig` command as **`standby`**. Any attempt to create this type of configuration will fail. Similarly, if you configure as **`standby`** an interface that already has data addresses, these addresses automatically fail over to another interface in the IPMP group. Due to these restrictions, you must use the `ifconfig` command to mark any test addresses as **`deprecated`** and **`-failover`** prior to setting the interface as **`standby`**. To configure standby interfaces, refer to How to Configure a Standby Interface for an IPMP Group.

Common IPMP Interface Configurations

As mentioned in IPMP Addressing, interfaces in an IPMP group handle regular data traffic and probe traffic, depending on the interfaces' configuration. You use IPMP options of the `ifconfig` command to create the configuration.

An **active interface** is a physical interface that transmits both data traffic and probe traffic. You configure the interface as “active” by performing either the task How to Configure an IPMP Group With Multiple Interfaces or the task How to Configure a Single Interface IPMP Group.

The following are two common types of IPMP configurations:

Active-active configuration

A two interface IPMP group where both interfaces are “active,” that is they might be transmitting both probe and data traffic at all times.

Active-standby configuration

A two interface IPMP group where one interface is configured as “standby.”

Checking the Status of an Interface

You can check the status of an interface by issuing the `ifconfig` *interface* command. For general information on `ifconfig` status reporting, refer to How to Get Information About a Specific Interface.

For example, you can use the `ifconfig` command to obtain the status of a standby interface. When the standby interface is not hosting any data address, the interface has the `INACTIVE` flag for its status. You can observe this flag in the status lines for the interface in the `ifconfig` output.

IPMP Failure Detection and Recovery Features

The `in.mpathd` daemon handles the following types of failure detection:

- Link-based failure detection, if supported by the NIC driver
- Probe-based failure detection, when test addresses are configured
- Detection of interfaces that were missing at boot time

The in.mpathd(1M) man page completely describes how the `in.mpathd` daemon handles the detection of interface failures.

Link-Based Failure Detection

Link-based failure detection is always enabled, provided that the interface supports this type of failure detection. The following Sun network drivers are supported in the current release of Oracle Solaris:

- `hme`
- `eri`
- `ce`
- `ge`
- `bge`
- `qfe`
- `dmfe`
- `e1000g`
- `ixgb`
- `nge`
- `nxge`
- `rge`
- `xge`

To determine whether a third-party interface supports link-based failure detection, refer to the manufacturer's documentation.

These network interface drivers monitor the interface's link state and notify the networking subsystem when that link state changes. When notified of a change, the networking subsystem either sets or clears the `RUNNING` flag for that interface, as appropriate. When the daemon detects that the interface's `RUNNING` flag has been cleared, the daemon immediately fails the interface.

Probe-Based Failure Detection

The `in.mpathd` daemon performs probe-based failure detection on each interface in the IPMP group that has a test address. Probe-based failure detection involves the sending and receiving of ICMP probe messages that use test addresses. These messages go out over the interface to one or more target systems on the same IP link. For an introduction to test addresses, refer to Test Addresses. For information on configuring test addresses, refer to How to Configure an IPMP Group With Multiple Interfaces.

The `in.mpathd` daemon determines which target systems to probe dynamically. Routers that are connected to the IP link are automatically selected as targets for probing. If no routers exist on the link, `in.mpathd` sends probes to neighbor hosts on the link. A multicast packet that is sent to the all hosts multicast address, `224.0.0.1` in IPv4 and `ff02::1` in IPv6, determines which hosts to use as target systems. The first few hosts that respond to the echo packets are chosen as targets for probing. If `in.mpathd` cannot find routers or hosts that responded to the ICMP echo packets, `in.mpathd` cannot detect probe-based failures.

You can use host routes to explicitly configure a list of target systems to be used by `in.mpathd`. For instructions, refer to Configuring Target Systems.

To ensure that each interface in the IPMP group functions properly, `in.mpathd` probes all the targets separately through all the interfaces in the IPMP group. If no replies are made in response to five consecutive probes, `in.mpathd` considers the interface to have failed. The probing rate depends on the **failure detection time** (FDT). The default value for failure detection time is 10 seconds. However, you can tune the failure detection time in the `/etc/default/mpathd` file. For instructions, go to How to Configure the `/etc/default/mpathd` File.

For a repair detection time of 10 seconds, the probing rate is approximately one probe every two seconds. The minimum repair detection time is twice the failure detection time, 20 seconds by default, because replies to 10 consecutive probes must be received. The failure and repair detection times apply only to probe-based failure detection.

Note –

In an IPMP group that is composed of VLANs, link-based failure detection is implemented per physical-link and thus affects all VLANs on that link. Probe-based failure detection is performed per VLAN-link. For example, `bge0/bge1` and `bge1000/bge1001` are configured together in a group. If the cable for `bge0` is unplugged, then link-based failure detection will report both `bge0` and `bge1000` as having instantly failed. However, if all of the probe targets on `bge0` become unreachable, only `bge0` will be reported as failed because `bge1000` has its own probe targets on its own VLAN.

Group Failures

A **group failure** occurs when all interfaces in an IPMP group appear to fail at the same time. The `in.mpathd` daemon does not perform failovers for a group failure. Also, no failover occurs when all the target systems fail at the same time. In this instance, `in.mpathd` flushes all of its current target systems and discovers new target systems.

Detecting Physical Interface Repairs

For the `in.mpathd` daemon to consider an interface to be repaired, the `RUNNING` flag must be set for the interface. If probe-based failure detection is used, the `in.mpathd` daemon must receive responses to 10 consecutive probe packets from the interface before that interface is considered repaired. When an interface is considered repaired, any addresses that failed over to another interface then fail back to the repaired interface. If the interface was configured as “active” before it failed, after repair that interface can resume sending and receiving traffic.

What Happens During Interface Failover

The following two examples show a typical configuration and how that configuration automatically changes when an interface fails. When the `hme0` interface fails, notice that all data addresses move from `hme0` to `hme1` .

Example 30–1 Interface Configuration Before an Interface Failure

```
hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
     mtu 1500 index 2
     inet 192.168.85.19 netmask ffffff00 broadcast 192.168.85.255
     groupname test
hme0:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
     mtu 1500
     index 2 inet 192.168.85.21 netmask ffffff00 broadcast 192.168.85.255
hme1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
8     inet 192.168.85.20 netmask ffffff00 broadcast 192.168.85.255
     groupname test
hme1:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
     mtu 1500
     index 2 inet 192.168.85.22 netmask ffffff00 broadcast 192.168.85.255
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
     inet6 fe80::a00:20ff:feb9:19fa/10
     groupname test
hme1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
     inet6 fe80::a00:20ff:feb9:1bfc/10
     groupname test
```

Example 30–2 Interface Configuration After an Interface Failure

```
hme0: flags=19000842<BROADCAST,RUNNING,MULTICAST,IPv4,
      NOFAILOVER,FAILED> mtu 0 index 2
      inet 0.0.0.0 netmask 0
      groupname test
hme0:1: flags=19040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,
      NOFAILOVER,FAILED> mtu 1500 index 2
      inet 192.168.85.21 netmask ffffff00 broadcast 10.0.0.255
hme1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 192.168.85.20 netmask ffffff00 broadcast 192.168.85.255
      groupname test
hme1:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,
      NOFAILOVER> mtu 1500
      index 2 inet 192.168.85.22 netmask ffffff00 broadcast 10.0.0.255
hme1:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
      inet 192.168.85.19 netmask ffffff00 broadcast 192.168.18.255
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER,FAILED> mtu 1500 index 2
      inet6 fe80::a00:20ff:feb9:19fa/10
      groupname test
hme1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
      inet6 fe80::a00:20ff:feb9:1bfc/10
      groupname test
```

You can see that the `FAILED` flag is set on `hme0` to indicate that this interface has failed. You can also see that `hme1:2` has been created. `hme1:2` was originally `hme0` . The address `192.168.85.19` then becomes accessible through `hme1` .

Multicast memberships that are associated with `192.168.85.19` can still receive packets, but they now receive packets through `hme1` . When the failover of address `192.168.85.19` from `hme0` to `hme1` occurred, a dummy address `0.0.0.0` was created on `hme0` . The dummy address was created so that `hme0` can still be accessed. `hme0:1` cannot exist without `hme0` . The dummy address is removed when a subsequent failback takes place.

Similarly, failover of the IPv6 address from `hme0` to `hme1` occurred. In IPv6, multicast memberships are associated with interface indexes. Multicast memberships also fail over from `hme0` to `hme1` . All the addresses that `in.ndpd` configured also moved. This action is not shown in the examples.

The `in.mpathd` daemon continues to probe through the failed interface `hme0` . After the daemon receives 10 consecutive replies for a default repair detection time of 20 seconds, the daemon determines that the interface is repaired. Because the `RUNNING` flag is also set on `hme0` , the daemon invokes the failback. After failback, the original configuration is restored.

For a description of all error messages that are logged on the console during failures and repairs, see the `in.mpathd` (1M) man page.

IPMP and Dynamic Reconfiguration

The dynamic reconfiguration (DR) feature enables you to reconfigure system hardware, such as interfaces, while the system is running. This section explains how DR interoperates with IPMP.

On a system that supports DR of NICs, IPMP can be used to preserve connectivity and prevent disruption of existing connections. You can safely attach, detach, or reattach NIC's on a system that supports DR and uses IPMP. This is possible because IPMP is integrated into the Reconfiguration Coordination Manager (RCM) framework. **RCM** manages the dynamic reconfiguration of system components.

You typically use the `cfgadm` command to perform DR operations. However, some platforms provide other methods. Consult your platform's documentation for details. You can find specific documentation about DR from the following resources.

Table 30–1 Documentation Resources for Dynamic Reconfiguration

Description	**For Information**
Detailed information on the `cfgadm` command	cfgadm(1M) man page
Specific information about DR in the Sun Cluster environment	*Sun Cluster 3.1 System Administration Guide*
Specific information about DR in the Sun Fire environment	*Sun Fire 880 Dynamic Reconfiguration Guide*
Introductory information about DR and the `cfgadm` command	Chapter 6, *Dynamically Configuring Devices (Tasks),* in *System Administration Guide: Devices and File Systems*
Tasks for administering IPMP groups on a system that supports DR	Replacing a Failed Physical Interface on Systems That Support Dynamic Reconfiguration

Attaching NICs

You can add interfaces to an IPMP group at any time by using the `ifconfig` command, as explained in How to Configure an IPMP Group With Multiple Interfaces. Thus, any interfaces on system components that you attach after system boot can be plumbed and added to an existing IPMP group. Or, if appropriate, you can configure the newly added interfaces into their own IPMP group.

These interfaces and the data addresses that are configured on them are immediately available for use by the IPMP group. However, for the system to automatically configure and use the interfaces after a reboot, you must create an `/etc/hostname.` *interface* file for each new interface. For instructions, refer toHow to Configure a Physical Interface After System Installation.

If an `/etc/hostname.` *interface* file already exists when the interface is attached, then RCM automatically configures the interface according to the contents of this file. Thus, the interface receives the same configuration that it would have received after system boot.

Detaching NICs

All requests to detach system components that contain NICs are first checked to ensure that connectivity can be preserved. For instance, by default you cannot detach a NIC that is not in an IPMP group. You also cannot detach a NIC that contains the only functioning interfaces in an IPMP group. However, if you must remove the system component, you can override this behavior by using the **`-f`** option of `cfgadm` , as explained in the cfgadm(1M) man page.

If the checks are successful, the data addresses associated with the detached NIC fail over to a functioning NIC in the same group, as if the NIC being detached had failed. When the NIC is detached, all test addresses on the NIC's interfaces are unconfigured. Then, the NIC is unplumbed from the system. If any of these steps fail, or if the DR of other hardware on the same system component fails, then the previous configuration is restored to its original state. You should receive a status message regarding this event. Otherwise, the detach request completes successfully. You can remove the component from the system. No existing connections are disrupted.

Reattaching NICs

RCM records the configuration information associated with any NIC's that are detached from a running system. As a result, RCM treats the reattachment of a NIC that had been previously detached identically as it would to the attachment of a new NIC. That is, RCM only performs plumbing.

However, reattached NICs typically have an existing `/etc/hostname.` *interface* file. In this case, RCM automatically configures the interface according to the contents of the existing `/etc/hostname.` *interface* file. Additionally, RCM informs the `in.mpathd` daemon of each data address that was originally hosted on the reattached interface. Thus, once the reattached interface is functioning properly, all of its data addresses are failed back to the reattached interface as if it had been repaired.

If the NIC being reattached does not have an `/etc/hostname.` *interface* file, then no configuration information is available. RCM has no information regarding how to configure the interface. One consequence of this situation is that addresses that were previously failed over to another interface are not failed back.

NICs That Were Missing at System Boot

NICs that are not present at system boot represent a special instance of failure detection. At boot time, the startup scripts track any interfaces with `/etc/hostname.` *interface* files that cannot be plumbed. Any data addresses in such an interface's `/etc/hostname.` *interface* file are automatically hosted on an alternative interface in the IPMP group.

In such an event, you receive error messages similar to the following

```
moving addresses from failed IPv4 interfaces: hme0 (moved to hme1)
moving addresses from failed IPv6 interfaces: hme0 (moved to hme1)
```

If no alternative interface exists, you receive error messages similar to the following:

```
moving addresses from failed IPv4 interfaces: hme0 (couldn't move;
   no alternative interface)
 moving addresses from failed IPv6 interfaces: hme0 (couldn't move;
   no alternative interface)
```

Note –

In this instance of failure detection, only data addresses that are explicitly specified in the missing interface's `/etc/hostname.` *interface* file move to an alternative interface. Any addresses that are usually acquired through other means, such as through RARP or DHCP, are not acquired or moved.

If an interface with the same name as another interface that was missing at system boot is reattached using DR, RCM automatically plumbs the interface. Then, RCM configures the interface according to the contents of the interface's `/etc/hostname.` *interface* file. Finally, RCM fails back any data addresses, just as if the interface had been repaired. Thus, the final network configuration is identical to the configuration that would have been made if the system had been booted with the interface present.

Chapter 31 Administering IPMP (Tasks)

This chapter provides tasks for administering interface groups with IP network multipathing (IPMP). The following major topics are discussed:

For an overview of IPMP concepts, refer to Chapter 30, Introducing IPMP (Overview).

Configuring IPMP (Task Maps)

This section contains links to the tasks that are described in this chapter.

Configuring and Administering IPMP Groups (Task Map)

Task	Description	For Instructions
Plan for an IPMP group.	Lists all ancillary information and required tasks before you can configure an IPMP group.	How to Plan for an IPMP Group
Configure an IPMP interface group with multiple interfaces.	Configures multiple interfaces as members of an IPMP group.	How to Configure an IPMP Group With Multiple Interfaces

Task	Description	For Instructions
Configure an IPMP group where one of the interfaces is a standby interface.	Configures one of the interfaces in a multiple interface IPMP group as a standby interface.	How to Configure a Standby Interface for an IPMP Group
Configure an IPMP group that consists of a single interface.	Creates a single interface IPMP group.	How to Configure a Single Interface IPMP Group
Display the IPMP group to which a physical interface belongs.	Explains how to obtain the name of an interface's IPMP group from the output of the `ifconfig` command.	How to Display the IPMP Group Membership of an Interface
Add an interface to an IPMP group.	Configures a new interface as a member of an existing IPMP group.	How to Add an Interface to an IPMP Group
Remove an interface from an IPMP group.	Explains how to remove an interface from an IPMP group.	How to Remove an Interface From an IPMP Group
Move an interface from an existing IPMP group to a different group.	Moves interfaces among IPMP groups.	How to Move an Interface From One IPMP Group to Another Group
Change three default settings for the `in.mpathd` daemon.	Customizes failure detection time and other parameters of the `in.mpathd` daemon.	How to Configure the `/etc/default/mpathd` File

Administering IPMP on Interfaces That Support Dynamic Reconfiguration (Task Map)

Task	Description	For Instructions
Remove an interface that has failed.	Removes a failed interface on a system.	How to Remove a Physical Interface That Has Failed (DR-Detach)
Replace an interface that has failed.	Replaces a failed interface.	How to Replace a Physical Interface That Has Failed (DR-Attach)
Recover an interface that was not configured at boot time.	Recovers a failed interface.	How to Recover a Physical Interface That Was Not Present at System Boot

Configuring IPMP Groups

This section provides procedures for configuring IPMP groups. It also describes how to configure an interface as a standby.

Planning for an IPMP Group

Before you configure interfaces on a system as part of an IPMP group, you need to do some preconfiguration planning.

▼ How to Plan for an IPMP Group

The following procedure includes the planning tasks and information to be gathered prior to configuring the IPMP group. The tasks do not have to be performed in sequence.

1. Decide which interfaces on the system are to be part of the IPMP group.

   An IPMP group usually consists of at least two physical interfaces that are connected to the same IP link. However, you can configure a single interface IPMP group, if required. For an introduction to IPMP groups, refer to IPMP Interface Configurations. For example, you can configure the same Ethernet switch or the same IP subnet under the same IPMP group. You can configure any number of interfaces into the same IPMP group.

   You cannot use the `group` parameter of the `ifconfig` command with logical interfaces. For example, you can use the `group` parameter with `hme0`, but not with `hme0:1`.

2. Verify that each interface in the group has a unique MAC address.

   For instructions, refer to SPARC: How to Ensure That the MAC Address of an Interface Is Unique.

3. Choose a name for the IPMP group.

   Any non-null name is appropriate for the group. You might want to use a name that identifies the IP link to which the interfaces are attached.

4. Ensure that the same set of `STREAMS` modules is pushed and configured on all interfaces in the IPMP group.

   All interfaces in the same group must have the same `STREAMS` modules configured in the same order.

   a. Check the order of STREAMS modules on all interfaces in the prospective IPMP group.

      You can print out a list of `STREAMS` modules by using the `ifconfig` *`interface`* `modlist` command. For example, here is the `ifconfig` output for an `hme0` interface:

      ```
      # ifconfig hme0 modlist
             0 arp
             1 ip
             2 hme
      ```

      Interfaces normally exist as network drivers directly below the IP module, as shown in the output from `ifconfig hme0 modlist`. They should not require additional configuration.

      However, certain technologies, such as NCA or IP Filter, insert themselves as STREAMS modules between the IP module and the network driver. Problems can result in the way interfaces of the same IPMP group behave.

      If a STREAMS module is stateful, then unexpected behavior can occur on failover, even if you push the same module onto all of the interfaces in a group. However, you can use stateless STREAMS modules, provided that you push them in the same order on all interfaces in the IPMP group.

   b. Push the modules of an interface in the standard order for the IPMP group.

      ```
      ifconfig interface modinsert module-name
      ```

      ```
      ifconfig hme0 modinsert ip
      ```

5. Use the same IP addressing format on all interfaces of the IPMP group.

   If one interface is configured for IPv4, then all interfaces of the group must be configured for IPv4. Suppose you have an IPMP group that is composed of interfaces from several NICs. If you add IPv6 addressing to the interfaces of one NIC, then all interfaces in the IPMP group must be configured for IPv6 support.

6. Check that all interfaces in the IPMP group are connected to the same IP link.

7. Verify that the IPMP group does not contain interfaces with different network media types.

   The interfaces that are grouped together should be of the same interface type, as defined in `/usr/include/net/if_types.h`. For example, you cannot combine Ethernet and Token ring interfaces in an IPMP group. As another example, you cannot combine a Token bus interface with asynchronous transfer mode (ATM) interfaces in the same IPMP group.

8. For IPMP with ATM interfaces, configure the ATM interfaces in LAN emulation mode.

   IPMP is not supported for interfaces using Classical IP over ATM.

Configuring IPMP Groups

This section contains configuration tasks for a typical IPMP group with at least two physical interfaces.

- For an introduction to multiple interface IPMP groups, refer to IPMP Group.
- For planning tasks, refer to Planning for an IPMP Group.
- To configure an IPMP group with only one physical interface, refer to Configuring IPMP Groups With a Single Physical Interface.

How to Configure an IPMP Group With Multiple Interfaces

The following steps for configuring an IPMP group also apply when configuring VLANs into an IPMP group.

Before You Begin

You need to have already configured the IPv4 addresses, and, if appropriate, the IPv6 addresses of all interfaces in the prospective IPMP group.

Caution –

You must configure only one IPMP group for each subnet or L2 broadcast domain. For more information, see Basic Requirements of IPMP

1. On the system with the interfaces to be configured, assume the Primary Administrator role, or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Place each physical interface into an IPMP group.

   ```
   # ifconfig interface group group-name
   ```

   For example, to place `hme0` and `hme1` under group `testgroup1` , you would type the following commands:

   ```
   # ifconfig hme0 group testgroup1
   # ifconfig hme1 group testgroup1
   ```

   Avoid using spaces in group names. The `ifconfig` status display does not show spaces. Consequently, do not create two similar group names where the only difference is that one name also contains a space. If one of the group names contains a space, these group names look the same in the status display.

   In a dual-stack environment, placing the IPv4 instance of an interface under a particular group automatically places the IPv6 instance under the same group.

3. (Optional) Configure an IPv4 test address on one or more physical interfaces.

   You need to configure a test address only if you want to use probe-based failure detection on a particular interface. Test addresses are configured as logical interfaces of the physical interface that you specify to the `ifconfig` command.

   If one interface in the group is to become the standby interface, do not configure a test address for that interface at this time. You configure a test address for the standby interface as part of the task How to Configure a Standby Interface for an IPMP Group.

   Use the following syntax of the `ifconfig` command for configuring a test address:

   ```
   # ifconfig interface addif ip-address parameters -failover deprecated up
   ```

   For example, you would create the following test address for the primary network interface `hme0` :

   ```
   # ifconfig hme0 addif 192.168.85.21 netmask + broadcast + -failover deprecated up
   ```

   This command sets the following parameters for the primary network interface `hme0` :

   - Address set to `192.168.85.21`
   - Netmask and broadcast address set to the default value
   - **`-failover`** and `deprecated` options set

   Note –

   You must mark an IPv4 test address as `deprecated` to prevent applications from using the test address.

4. Check the IPv4 configuration for a specific interface.

   You can always view the current status of an interface by typing `ifconfig` *interface*. For more information on viewing an interface's status, refer to How to Get Information About a Specific Interface.

   You can get information about test address configuration for a physical interface by specifying the logical interface that is assigned to the test address.

   ```
   # ifconfig hme0:1
   ```

```
        hme0:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
    mtu 1500 index 2
    inet 192.168.85.21 netmask ffffff00 broadcast 192.168.85.255
```

5. (Optional) If applicable, configure an IPv6 test address.

   ```
   # ifconfig interface inet6 -failover
   ```

   Physical interfaces with IPv6 addresses are placed into the same IPMP group as the interfaces' IPv4 addresses. This happens when you configure the physical interface with IPv4 addresses into an IPMP group. If you first place physical interfaces with IPv6 addresses into an IPMP group, physical interfaces with IPv4 addresses are also implicitly placed in the same IPMP group.

   For example, to configure `hme0` with an IPv6 test address, you would type the following:

   ```
   # ifconfig hme0 inet6 -failover
   ```

   You do not need to mark an IPv6 test address as deprecated to prevent applications from using the test address.

6. Check the IPv6 configuration.

   ```
   # ifconfig hme0 inet6
           hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
                   inet6 fe80::a00:20ff:feb9:17fa/10
                   groupname test
   ```

   The IPv6 test address is the link-local address of the interface.

7. (Optional) Preserve the IPMP group configuration across reboots.

   - For IPv4, add the following line to the `/etc/hostname.` *interface* file:

     ```
     interface-address <parameters> group group-name up \
             addif logical-interface -failover deprecated <parameters> up
     ```

     In this instance, the test IPv4 address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1, 2, and, optionally 3.

   - For IPv6, add the following line to the `/etc/hostname6.` *interface* file:

     ```
     -failover group group-name up
     ```

     This test IPv6 address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1, 2, and, optionally, 5.

8. (Optional) Add more interfaces to the IPMP group by repeating steps 1 through 6.

   You can add new interfaces to an existing group on a live system. However, changes are lost across reboots.

Example 31–1 Configuring an IPMP Group With Two Interfaces

Suppose you want to do the following:

- Have the netmask and broadcast address set to the default value.
- Configure the interface with a test address `192.168.85.21` .

You would type the following command:

```
# ifconfig hme0 addif 192.168.85.21 netmask + broadcast + -failover deprecated up
```

You must mark an IPv4 test address as `deprecated` to prevent applications from using the test address. See How to Configure an IPMP Group With Multiple Interfaces.

To turn on the failover attribute of the address, you would use the **`failover`** option without the dash

All test IP addresses in an IPMP group must use the same network prefix. The test IP addresses must belong to a single IP subnet.

Example 31–2 Preserving an IPv4 IPMP Group Configuration Across Reboots

Suppose you want to create an IPMP group called `testgroup1` with the following configuration:

- Physical interface `hme0` with the data address `192.168.85.19`
- A logical interface with the test address `192.168.85.21`

  Note –

  In this example, physical interface and data address are paired together. Likewise for logical interface and test address. However, no inherent relationships exist between an interface “type” and the address type.

- `deprecated` and **`-failover`** options set
- Netmask and broadcast address set to the default value

You would add the following line to the `/etc/hostname.hme0` file:

```
192.168.85.19 netmask + broadcast + group testgroup1 up \
        addif 192.168.85.21 deprecated -failover netmask + broadcast + up
```

Similarly, to place the second interface `hme1` under the same group `testgroup1` and to configure a test address, you would add the following line:

```
192.168.85.20 netmask + broadcast + group testgroup1 up \
        addif 192.168.85.22 deprecated -failover netmask + broadcast + up
```

Example 31–3 Preserving an IPv6 IPMP Group Configuration Across Reboots

To create a test group for interface `hme0` with an IPv6 address, you would add the following line to the `/etc/hostname6.hme0` file:

```
-failover group testgroup1 up
```

Similarly, to place the second interface `hme1` in group `testgroup1` and to configure a test address, you would add the following line to the `/etc/hostname6.hme1` file:

```
-failover group testgroup1 up
```

Troubleshooting

During IPMP group configuration, `in.mpathd` outputs a number of messages to the system console or to the `syslog` file. These messages are informational in nature and indicate that the IPMP configuration functions correctly.

- This message indicates that interface `hme0` was added to IPMP group `testgroup1` . However, `hme0` does not have a test address configured. To enable probe-based failure detection, you need to assign a test address to the interface.

```
May 24 14:09:57 host1 in.mpathd[101180]:
No test address configured on interface hme0;
disabling probe-based failure detection on it.
testgroup1
```

- This message appears for all interfaces with only IPv4 addresses that are added to an IPMP group.

```
May 24 14:10:42 host4 in.mpathd[101180]:
NIC qfe0 of group testgroup1 is not
plumbed for IPv6 and may affect failover capability
```

- This message should appear when you have configured a test address for an interface.

```
Created new logical interface hme0:1
May 24 14:16:53 host1 in.mpathd[101180]:
Test address now configured on interface hme0;
```

```
enabling probe-based failure detection on it
```

See Also

If you want the IPMP group to have an active-standby configuration, go on to How to Configure a Standby Interface for an IPMP Group.

Configuring Target Systems

Probe-based failure detection involves the use of target systems, as explained in Probe-Based Failure Detection. For some IPMP groups, the default targets used by `in.mpathd` is sufficient. However, for some IPMP groups, you might want to configure specific targets for probe-based failure detection. You accomplish probe-based failure detection by setting up host routes in the routing table as probe targets. Any host routes that are configured in the routing table are listed before the default router. Therefore, IPMP uses the explicitly defined host routes for target selection. You can use either of two methods for directly specifying targets: manually setting host routes or creating a shell script that can become a startup script.

Consider the following criteria when evaluating which hosts on your network might make good targets.

- Make sure that the prospective targets are available and running. Make a list of their IP addresses.
- Ensure that the target interfaces are on the same network as the IPMP group that you are configuring.
- The netmask and broadcast address of the target systems must be the same as the addresses in the IPMP group.
- The target host must be able to answer ICMP requests from the interface that is using probe-based failure detection.

How to Manually Specify Target Systems for Probe-Based Failure Detection

1. Log in with your user account to the system where you are configuring probe-based failure detection.
2. Add a route to a particular host to be used as a target in probe-based failure detection.

   ```
   $  route add -host  destination-IP gateway-IP  -static
   ```

   Replace the values of *destination-IP* and *gateway-IP* with the IPv4 address of the host to be used as a target. For example, you would type the following to specify the target system `192.168.85.137`, which is on the same subnet as the interfaces in IPMP group `testgroup1`.

   ```
   $  route add -host 192.168.85.137 192.168.85.137 -static
   ```

3. Add routes to additional hosts on the network to be used as target systems.

How to Specify Target Systems in a Shell Script

1. On the system where you have configured an IPMP group, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Create a shell script that sets up static routes to your proposed targets.

   For example, you could create a shell script called `ipmp.targets` with the following contents:

   ```
   TARGETS="192.168.85.117 192.168.85.127 192.168.85.137"

   case "$1" in
           'start')
               /usr/bin/echo "Adding static routes for use as IPMP targets"
                   for target in $TARGETS; do
             /usr/sbin/route add -host $target $target
                   done
                     ;;
           'stop')
                 /usr/bin/echo "Removing static routes for use as IPMP targets"
                    for target in $TARGETS; do
                   /usr/sbin/route delete -host $target $target
                    done
                     ;;
     esac
   ```

3. Copy the shell script to the startup script directory.

   ```
   #  cp ipmp.targets /etc/init.d
   ```

4. Change the permissions on the new startup script.

   ```
   #  chmod 744 /etc/init.d/ipmp.targets
   ```

5. Change ownership of the new startup script.

   ```
   #  chown root:sys /etc/init.d/ipmp.targets
   ```

6. Create a link for the startup script in the `/etc/init.d` directory.

   ```
   # ln  /etc/init.d/ipmp.targets /etc/rc2.d/S70ipmp.targets
   ```

   The S70 prefix in the file name `S70ipmp.targets` orders the new script properly with respect to other startup scripts.

Configuring Standby Interfaces

Use this procedure if you want the IPMP group to have an active-standby configuration. For more information on this type of configuration, refer to IPMP Interface Configurations.

How to Configure a Standby Interface for an IPMP Group

Before You Begin

- You must have configured all interfaces as members of the IPMP group.
- You should not have configured a test address on the interface to become the standby interface.

For information on configuring an IPMP group and assigning test addresses, refer to How to Configure an IPMP Group With Multiple Interfaces.

1. On the system with the standby interfaces to be configured, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Configure an interface as a standby and assign the test address.

   ```
   #  ifconfig interface plumb \
   ip-address other-parameters deprecated -failover standby up
   ```

   A standby interface can have only one IP address, the test address. You must set the **`-failover`** option before you set the `standby up` option. For `<other-parameters>` , use the parameters that are required by your configuration, as described in the ifconfig(1M) man page.

   - For example, to create an IPv4 test address, you would type the following command:

     ```
     #  ifconfig hme1 plumb 192.168.85.22 netmask + broadcast + deprecated -failover standby up
     ```

     `hme1`

     Defines `hme1` as the physical interface to be configured as the standby interface.

     `192.168.85.22`

     Assigns this test address to the standby interface.

     `deprecated`

     Indicates that the test address is not used for outbound packets.

     `-failover`

     Indicates that the test address does not fail over if the interface fails.

     `standby`

     Marks the interface as a standby interface.

   - For example, to create an IPv6 test address, you would type the following command:

     ```
     #  ifconfig hme1 plumb -failover standby up
     ```

3. Check the results of the standby interface configuration.

   ```
   #  ifconfig hme1
   hme1: flags=69040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,
         STANDBY,INACTIVE mtu 1500
            index 4 inet 192.168.85.22 netmask ffffff00 broadcast 19.16.85.255
            groupname test
   ```

   The `INACTIVE` flag indicates that this interface is not used for any outbound packets. When a failover occurs on this standby interface, the `INACTIVE` flag is cleared.

   Note –

   You can always view the current status of an interface by typing the `ifconfig` *interface* command. For more information on viewing interface status, refer to How to Get Information About a Specific Interface.

4. (Optional) Preserve the IPv4 standby interface across reboots.

   Assign the standby interface to the same IPMP group, and configure a test address for the standby interface.

   For example, to configure `hme1` as the standby interface, you would add the following line to the `/etc/hostname.hme1` file:

   ```
   192.168.85.22 netmask + broadcast + deprecated group test -failover standby up
   ```

5. (Optional) Preserve the IPv6 standby interface across reboots.

   Assign the standby interface to the same IPMP group, and configure a test address for the standby interface.

   For example, to configure `hme1` as the standby interface, add the following line to the `/etc/hostname6.hme1` file:

   ```
   -failover group test standby up
   ```

Example 31–4 Configuring a Standby Interface for an IPMP Group

Suppose you want to create a test address with the following configuration:

- Physical interface `hme2` as a standby interface
- Test address of `192.168.85.22`
- `deprecated` and **`-failover`** options set
- Netmask and broadcast address set to the default value

You would type the following:

```
# ifconfig hme2 plumb 192.168.85.22 netmask + broadcast + \
deprecated -failover standby up
```

The interface is marked as a standby interface only after the address is marked as a `NOFAILOVER` address.

You would remove the standby status of an interface by typing the following:

```
# ifconfig interface -standby
```

Configuring IPMP Groups With a Single Physical Interface

When you have only one interface in an IPMP group, failover is not possible. However, you can enable failure detection on that interface by assigning the interface to an IPMP group. You do not have to configure a dedicated test IP address to establish failure detection for a single interface IPMP group. You can use a single IP address for sending data and detecting failure.

How to Configure a Single Interface IPMP Group

1. On the system with the prospective single interface IPMP group, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. For IPv4, create the single interface IPMP group.

   Use the following syntax to assign the single interface to an IPMP group.

   ```
   # ifconfig interface group group-name
   ```

   The following example assigns the interface `hme0` into the IPMP group `v4test`:

   ```
   # ifconfig hme0 group v4test
   ```

   After this step is performed, IPMP enables link-based failure detection on the interface.

   In addition, you can also use the `-failover` subcommand of the `ifconfig` command to enable probe-based failure detection. The following example enables probe-based failure detection on `hme0` by using the IP address currently assigned to `hme0`:

   ```
   # ifconfig hme0 -failover
   ```

   Note that unlike multiple-interface groups, the same IP address can act as both a data address and a test address. To enable applications to use the test address as a data address, test addresses must never be marked **`deprecated`** on single-interface IPMP groups.

3. For IPv6, create the single interface IPMP group.

   Use the following syntax to assign a single interface to an IPMP group:

   ```
   # ifconfig interface inet6 group group-name
   ```

   For example, to add the single interface `hme0` into the IPMP group `v6test`, type the following:

   ```
   # ifconfig hme0 inet6 group v6test
   ```

   After this step is performed, IPMP enables link-based failure detection on the interface.

   In addition, you can also use the `-failover` subcommand of the `ifconfig` command to enable probe-based failure detection. The following example enables probe-based failure detection on `hme0` by using the IP address currently assigned to `hme0`:

   ```
   # ifconfig hme0 inet6 -failover
   ```

   Note that unlike multiple-interface groups, the same IP address can act as both a data address and a test address. To enable applications to use the test address as a data address, test addresses must never be marked **`deprecated`** on single-interface IPMP groups.

   In a single physical interface configuration, you cannot verify whether the target system that is being probed has failed or whether the interface has failed. The target system can be probed through only one physical interface. If only one default router is on the subnet, turn off IPMP if a single physical interface is in the group. If a separate IPv4 and IPv6 default router exists, or multiple default routers exist, more than one target system needs to be probed. Hence, you can safely turn on IPMP.

Maintaining IPMP Groups

This section contains tasks for maintaining existing IPMP groups and the interfaces that compose those groups. The tasks presume that you have already configured an IPMP group, as explained in Configuring IPMP Groups.

▼ How to Display the IPMP Group Membership of an Interface

1. On the system with the IPMP group configuration, become superuser or assume an equivalent role.

   Roles contain authorizations and privileged commands. For more information about roles, see *Configuring RBAC (Task Map)* in *System Administration Guide: Security Services*.

2. Display information about the interface, including the group to which the interface belongs.

   ```
   # ifconfig  interface
   ```

3. If applicable, display IPv6 information for the interface.

   ```
   # ifconfig  interface  inet6
   ```

Example 31–5 Displaying Physical Interface Groups

To display the group name for `hme0`, you would type the following:

```
#  ifconfig hme0
        hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
      index 2 inet 192.168.85.19 netmask ffffff00 broadcast 192.168.85.255
      groupname testgroup1
```

To display the group name for only the IPv6 information, you would type the following:

```
#  ifconfig hme0 inet6
        hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
                inet6 fe80::a00:20ff:feb9:19fa/10
                groupname testgroup1
```

▼ How to Add an Interface to an IPMP Group

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Add the interface to the IPMP group.

   ```
   # ifconfig interface group group-name
   ```

   The interface specified in *interface* becomes a member of IPMP group *group-name*.

Example 31–6 Adding an Interface to an IPMP Group

To add `hme0` to the IPMP group `testgroup2`, you would type the following command:

```
#  ifconfig hme0 group testgroup2
  hme0: flags=9000843<UP ,BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER> mtu 1500 index 2
  inet 192.168.85.19 netmask ff000000 broadcast 10.255.255.255
  groupname testgroup2
  ether 8:0:20:c1:8b:c3
```

▼ How to Remove an Interface From an IPMP Group

When you execute the `ifconfig` command's `group` parameter with a null string, the interface is removed from its current IPMP group. Be careful when removing interfaces from a group. If some other interface in the IPMP group has failed, a failover could have happened earlier. For example, if `hme0` failed previously, all addresses are failed over to `hme1`, if `hme1` is part of the same group. The removal of `hme1` from the group causes the `in.mpathd` daemon to return all the failover addresses to some other interface in the group. If no other interfaces are functioning in the group, failover might not restore all the network accesses.

Similarly, when an interface in a group needs to be unplumbed, you should first remove the interface from the group. Then, ensure that the interface has all the original IP addresses configured. The `in.mpathd` daemon tries to restore the original configuration of an interface that is removed from the group. You need to ensure that the configuration is restored before unplumbing the interface. Refer to What Happens During Interface Failover to see how interfaces look before and after a failover.

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Remove the interface from the IPMP group.

   ```
   # ifconfig interface group ""
   ```

   The quotation marks indicate a null string.

Example 31–7 Removing an Interface From a Group

To remove hme0 from the IPMP group test , you would type the following command:

```
# ifconfig hme0 group ""
        # ifconfig hme0
        hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
    index 2 inet 192.168.85.19 netmask ffffff00 broadcast 192.168.85.255
        # ifconfig hme0 inet6
        hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:feb9:19fa/10
```

▼ How to Move an Interface From One IPMP Group to Another Group

You can place an interface in a new IPMP group when the interface belongs to an existing IPMP group. You do not need to remove the interface from the current IPMP group. When you place the interface in a new group, the interface is automatically removed from any existing IPMP group.

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Move the interface to a new IPMP group.

   ```
   # ifconfig interface group group-name
   ```

   Placing the interface in a new group automatically removes the interface from any existing group.

Example 31–8 Moving an Interface to a Different IPMP Group

To change the IPMP group of interface hme0 , you would type the following:

```
# ifconfig hme0 group cs-link
```

This command removes the hme0 interface from IPMP group test and then puts the interface in the group cs-link .

Replacing a Failed Physical Interface on Systems That Support Dynamic Reconfiguration

This section contains procedures that relate to administering systems that support dynamic reconfiguration (DR).

Note –

The tasks pertain only to IP layers that are configured by using the ifconfig command. Layers before or after the IP layer, such as ATM or other services, require specific manual steps if the layers are not automated. The steps in the next procedures are used to unconfigure interfaces during predetachment and configure interface after postattachment.

▼ How to Remove a Physical Interface That Has Failed (DR-Detach)

This procedure shows how to remove a physical interface on a system that supports DR. The procedure assumes that the following conditions already exist:

- Physical interfaces hme0 and hme1 are the example interfaces.
- Both interfaces are in the same IPMP group.
- hme0 has failed.
- Logical interface hme0:1 has the test address.
- You are replacing the failed interface with the same physical interface name, for example, hme0 with hme0 .

Note –

You can skip Step 2 if the test address is plumbed by using the `/etc/hostname.hme0` file.

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Display the test address configuration.

```
# ifconfig hme0: 1

hme0:1:
flags=9040842<BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3
inet 192.168.233.250 netmask ffffff00 broadcast 192.168.233.255
```

   You need this information to replumb the test address when replacing the physical interface.

3. Remove the physical interface.

   Refer to the following sources for a complete description of how to remove the physical interface:

   - cfgadm(1M) man page
   - *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*
   - *Sun Enterprise 10000 DR Configuration Guide*

▼ How to Replace a Physical Interface That Has Failed (DR-Attach)

This procedure shows how to replace a physical interface on a system that supports DR.

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Replace the physical interface.

   Refer to the instructions in the following sources:

   - cfgadm(1M) man page
   - *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*
   - *Sun Enterprise 10000 DR Configuration Guide*, or *Sun Fire 880 Dynamic Reconfiguration User's Guide*

Recovering a Physical Interface That Was Not Present at System Boot

Note –

The following procedure pertains only to IP layers that are configured by using the `ifconfig` command. Layers before or after the IP layer, such as ATM or other services, require specific manual steps if the layers are not automated. The specific steps in the next procedure are used to unconfigure interfaces during predetachment and to configure interfaces after postattachment.

Recovery after dynamic reconfiguration is automatic for an interface that is part of the I/O board on a Sun Fire™ platform. If the NIC is a Sun Crypto Accelerator I - cPCI board, the recovery is also automatic. Consequently, the following steps are not required for an interface that is coming back as part of a DR operation. For more information on the Sun Fire x800 and Sun Fire 15000 systems, see the cfgadm_sbd(1M) man page. The physical interface fails back to the configuration that is specified in the `/etc/hostname.` *interface* file. See Configuring IPMP Groups for details on how to configure interfaces to preserve the configuration across reboots.

Note –

On Sun Fire legacy (Exx00) systems, DR detachments are still subject to manual procedures. However, DR attachments are automated.

▼ How to Recover a Physical Interface That Was Not Present at System Boot

You must complete the following procedure before you recover a physical interface that was not present at system boot. The example in this procedure has the following configuration:

- Physical interfaces `hme0` and `hme1` are the interfaces.

- Both interfaces are in the same IPMP group.
- `hme0` was not installed at system boot.

Note –

The failback of IP addresses during the recovery of a failed physical interface takes up to three minutes. This time might vary, depending on network traffic. The time also depends on the stability of the incoming interface to fail back the failed-over interfaces by the `in.mpathd` daemon.

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Retrieve the failed network information from the failure error message of the console log.

   See the syslog(3C)man page. The error message might be similar to the following:

   ```
   moving addresses from failed IPv4 interfaces:
   hme1 (moved to hme0)
   ```

   This message indicates that the IPv4 addresses on the failed interface `hme1` have failed over to the `hme0` interface.

   Alternatively, you might receive the following similar message:

   ```
   moving addresses from failed IPv4 interfaces:
   hme1 (couldn't move, no alternative interface)
   ```

   This message indicates that no active interface could be found in the same group as failed interface `hme1` . Therefore, the IPv4 addresses on `hme1` could not fail over.

3. Attach the physical interface to the system.

   Refer to the following for instructions on how to replace the physical interface:

   - cfgadm(1M) man page
   - *Sun Enterprise 10000 DR Configuration Guide*
   - *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*

4. Refer to the message content from Step 2. If the addresses could not be moved, go to Step 6. If the addresses were moved, continue to Step 5.

5. Unplumb the logical interfaces that were configured as part of the failover process.

   a. Review the contents of the `/etc/hostname.`*`moved-from-interface`* file to determine what logical interfaces were configured as part of the failover process.

   b. Unplumb each failover IP address.

   ```
   # ifconfig moved-to-interface removeif moved-ip-address
   ```

   Note –

   Failover addresses are marked with the `failover` parameter, or are not marked with the `-failover` parameter. You do not need to unplumb IP addresses that are marked `-failover` .

   For example, assume that the contents of the `/etc/hostname.hme0` file contains the following lines:

   ```
   inet 10.0.0.4 -failover up group one
   addif 10.0.0.5 failover up
   addif 10.0.0.6 failover up
   ```

   To unplumb each failover IP address, you would type the following commands:

   ```
   # ifconfig hme0 removeif 10.0.0.5
   # ifconfig hme0 removeif 10.0.0.6
   ```

6. Reconfigure the IPv4 information for the replaced physical interface by typing the following command for each interface that was removed:

   ```
   # ifconfig removed-from-NIC <parameters>
   ```

For example, you would type the following commands:

```
# ifconfig hme1 inet plumb
# ifconfig hme1 inet 10.0.0.4 -failover up group one
# ifconfig hme1 addif 10.0.0.5 failover up
# ifconfig hme1 addif 10.0.0.6 failover up
```

Modifying IPMP Configurations

Use the IPMP configuration file `/etc/default/mpathd` to configure the following system-wide parameters for IPMP groups.

- `FAILURE_DETECTION_TIME`
- `TRACK_INTERFACES_ONLY_WITH_GROUPS`
- `FAILBACK`

▼ How to Configure the `/etc/default/mpathd` File

1. On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, *Working With the Solaris Management Console (Tasks),* in *System Administration Guide: Basic Administration*.

2. Edit the `/etc/default/mpathd` file.

   Change the default value of one or more of the three parameters.

   a. Type the new value for the `FAILURE_DETECTION_TIME` parameter.

   ```
   FAILURE_DETECTION_TIME=n
   ```

   where *n* is the amount of time in seconds for ICMP probes to detect whether an interface failure has occurred. The default is 10 seconds.

   b. Type the new value for the `FAILBACK` parameter.

   ```
   FAILBACK=[yes | no]
   ```

   - *yes*- The *yes* value is the default failback behavior of IPMP. When the repair of a failed interface is detected, network access fails back to the repaired interface, as described in IPMP Failure Detection and Recovery Features.
   - *no* - The *no* indicates that data traffic does not move back to a repaired interface. When a failed interfaces is detected as repaired, the *INACTIVE* flag is set for that interface. This flag indicates that the interface is currently not to be used for data traffic. The interface can still be used for probe traffic.

     For example, suppose an IPMP group consists of two interfaces, ce0 and ce1. Then assume that the value *FAILBACK=no* is set in `/etc/default/mpathd`. If ce0 fails, its traffic fails over to ce1, as is the expected behavior of IPMP. However, when IPMP detects that ce0 is repaired, traffic does not fail back from ce1, due to the *FAILBACK=no* parameter in `/etc/default/mpathd`. The ce0 interface retains its `INACTIVE` status and is not used for traffic unless the ce1 interface fails. If the ce1 interface fails, the addresses on ce1 are migrated back to ce0, whose INACTIVE flag is then cleared. This migration occurs provided that ce0 is the only INACTIVE interface in the group. If other INACTIVE interfaces exist in the group, the addresses might be migrated to an INACTIVE interface other than ce0.

   c. Type the new value for the `TRACK_INTERFACES_ONLY_WITH_GROUPS` parameter.

   ```
   TRACK_INTERFACES_ONLY_WITH_GROUPS=[yes | no]
   ```

   - *yes*- The *yes* value is the default behavior of IPMP. This parameter causes IPMP to ignore network interfaces that are not configured into an IPMP group.
   - *no* - The *no* value sets failure and repair detection for **all** network interfaces, regardless of whether they are configured into an IPMP group. However, when a failure or repair is detected on an interface that is not configured into an IPMP group, no failover or failback occurs. Therefore, the*no* value is only useful for reporting failures and does not directly improve network availability.

3. Restart the `in.mpathd` daemon.

```
# pkill -HUP in.mpathd
```

Previous: Part V Mobile IP   **Next**: Part VII IP Quality of Service (IPQoS)

© 2010, Oracle Corporation and/or its affiliates