Configuring an Oracle® Solaris 11.4 System as a Router or a Load Balancer

Part No: E60992
November 2020

ORACLE

Configuring an Oracle Solaris 11.4 System as a Router or a Load Balancer

Part No: E60992

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Référence: E60992

Restrictions de licence/Avis d'exclusion de responsabilité en cas de dommage indirect et/ou consécutif

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Exonération de garantie

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Avis sur la limitation des droits

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Avis sur les applications dangereuses

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Marques

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Inside sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Epyc, et le logo AMD sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Avis d'exclusion de responsabilité concernant les services, produits et contenu tiers

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Date de publication et mention de la version préliminaire de Disponibilité Générale ("Pre-GA")

Version préliminaire de Disponibilité Générale ("Pre-GA") : 15.01.2020

Avis sur la version préliminaire de Disponibilité Générale ("Pre-GA") de la documentation

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère public ou privé :

Cette documentation est fournie dans la Version préliminaire de Disponibilité Générale ("Pre-GA") et uniquement à des fins de démonstration et d'usage à titre préliminaire de la version finale. Celle-ci n'est pas toujours spécifique du matériel informatique sur lequel vous utilisez ce logiciel. Oracle Corporation et ses affiliés déclinent expressément toute responsabilité ou garantie expresse quant au contenu de cette documentation. Oracle Corporation et ses affiliés ne sauraient en aucun cas être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'utilisation de cette documentation.

Mention sur les informations confidentielles Oracle

INFORMATIONS CONFIDENTIELLES ORACLE. Destinées uniquement à un usage autorisé. Ne pas distribuer à des tiers.

Avis sur la reconnaissance du revenu

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère privé :

Les informations contenues dans ce document sont fournies à titre informatif uniquement et doivent être prises en compte en votre qualité de membre du customer advisory board ou conformément à votre contrat d'essai de Version préliminaire de Disponibilité Générale ("Pre-GA") uniquement. Ce document ne constitue en aucun cas un engagement à fournir des composants, du code ou des fonctionnalités et ne doit pas être retenu comme base d'une quelconque décision d'achat. Le développement, la commercialisation et la mise à disposition des fonctions ou fonctionnalités décrites restent à la seule discrétion d'Oracle.

Accessibilité de la documentation

Pour plus d'informations sur l'engagement d'Oracle pour l'accessibilité de la documentation, visitez le site Web Oracle Accessibility Program, à l'adresse : http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info ou le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Describes how to configure Oracle Solaris 11.4 as an IPv4 or an IPv6 router. Provides an overview and configuration instructions for Virtual Router Redundancy Protocol (VRRP) and integrated load balancer (ILB).
- **Audience** – System administrators.
- **Required knowledge** – Basic and some advanced network administration skills.

Product Documentation Library

Documentation and resources for this product and related products are available at http://www.oracle.com/pls/topic/lookup?ctx=E37838-01.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPTER 1

Introduction to Routers

This chapter describes how routers can be used in Oracle Solaris to connect computer networks. Routers handle routing activity by using different protocols, some of which are discssed here in detail.

The chapter covers the following topics:

- "Router Overview"
- "Routing Protocols"
- "Routing Tables and Routing Types"
- "Using Rights Profiles to Perform Network Configuration"

Router Overview

A router is a device that is used in a computer network to connect computers and transfer packets of data among themselves. A router can have two or more connections from different networks. The router reads the address information from the incoming data packets to determine their destination. Then, packets are forwarded to the next network by using the information in the router's routing table. This process is repeated until the data packets reach the destination node.

Note - IP addresses that are used in Oracle Solaris 11 documentation conform to RFC 5737, IPv4 Address Blocks Reserved for Documentation (`https://tools.ietf.org/html/rfc5737`) and RFC 3849, IPv6 Address Prefix Reserved for Documentation (`https://tools.ietf.org/html/rfc3849`). IPv4 addresses used in this documentation are blocks 192.0.2.0/24, 198.51.100.0/24, and 203.0.113.0/24. IPv6 addresses have prefix 2001:DB8::/32.

To show a subnet, the block is divided into multiple subnets by borrowing enough bits from the host to create the required subnet. For example, host address 192.0.2.0 might have subnets 192.0.2.32/27 and 192.0.2.64/27.

Routing Protocols

Routing protocols handle routing activity on systems and routers. Routers and other systems exchange routing information about known routes to remote networks. These protocols assist the system in determining where to forward packets. Some routing protocols, although not all, also maintain statistics that you can use to measure routing performance.

The following table describes the supported routing protocols in Oracle Solaris.

TABLE 1 Oracle Solaris Routing Protocols

Protocol	Associated Daemon	Description	For Instructions
RIP	`in.routed`	Interior Gateway Protocol (IGP) that routes IPv4 packets and maintains a routing table	"Configuring an IPv4 Router" on page 19
RDISC	`in.routed`	Enable systems to discover the presence of a router on the network	"Enabling Routing for Single-Interface Systems" on page 31
RIPng	`in.ripngd`	IGP that routes IPv6 packets and maintains a routing table	"How to Configure an IPv6-Enabled Router" on page 25
Neighbor Discovery Protocol (NDP)	`in.ndpd`	Advertises the presence of an IPv6 router and discovers the presence of IPv6 systems on a network	"How to Configure a System for IPv6" in *Configuring and Managing Network Components in Oracle Solaris 11.4*

For more information about routing tables and types in Oracle Solaris, see "Routing Tables and Routing Types" on page 16.

Routing Information Protocol

Routing Information Protocol (RIP) is a distance-vector routing protocol. RIP uses a hop counter as its routing metric. It is implemented by the routing daemon `in.routed`. The daemon automatically starts when the system is booted. When run on a router with the `-s` option specified, the `in.routed` daemon fills the kernel routing table with a route to every reachable network and advertises reachability through all network interfaces. When run on a system with the `-q` option specified, the `in.routed` daemon extracts routing information but does not advertise reachability.

On systems, routing information can be extracted in the following two ways:

- By *not* specifying the flag (capital `S` or space-saving mode). The `in.routed` daemon builds a full routing table exactly as it does on a router.

- By specifying the flag. The `in.routed` daemon creates a minimal kernel table containing a single default route for each available router.

ICMP Router Discovery Protocol

Systems use the Router Discovery (RDISC) protocol to obtain routing information from routers. When systems run RDISC, routers must also run another protocol, such as RIP, to exchange router information.

RDISC is implemented by the daemon `in.routed`, which must run on both routers and systems. On systems, `in.routed` uses RDISC to discover default routes from routers that advertise the address through RDISC. On routers, `in.routed` uses RDISC to advertise default routes to systems on directly-connected networks. See the `in.routed(8)` man page and the `gateways(5)` man page for more information.

Quagga Routing Protocol Suite

Quagga is a routing software suite that enables the implementation of RIP, RIPng, Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Border Gateway Protocol (BGP) protocols for UNIX platforms including Oracle Solaris.

RIPng offers an extension of RIP for support of IPv6, including various enhancements for IPv6. The functions of RIPng are similar to those of RIP.

OSPF is a router protocol which is used to distribute routing information within a larger autonomous system network. The latest version of OSPF, OSPFv3, adds support for IPv6.

IS-IS is a link state dynamic routing protocol which is used to distribute routing information within a large service provider network.

BGP uses a prefixed set of IP networks to make routing decisions based on the path and rules among large autonomous system networks.

The following table lists the Open Source Quagga routing protocols that are supported in Oracle Solaris.

TABLE 2 Quagga Routing Protocol Suite

Protocol	Associated Daemon	Description
RIP	`ripd`	IPv4 distance vectoring IGP that routes IPv4 packets and advertises its routing table to neighbors

Protocol	Associated Daemon	Description
RIPng	`ripngd`	IPv6 distance vectoring IGP that routes IPv6 packets and maintains a routing table
OSPF	`ospfd`	IPv4 link state IGP for packet routing and high availability networking
BGP	`bgpd`	IPv4 and IPv6 Exterior Gateway Protocol (EGP) for routing across administrative domains
IS-IS	`isisd`	IPv4 and IPv6 link state IGP for routing within an administrative domain or network

For more information about the Quagga protocols, go to the Quagga Routing Suite web site at `http://www.nongnu.org/quagga/index.html`.

Routing Tables and Routing Types

Both routers and hosts maintain a *routing table*. The routing table lists the IP addresses of networks that the system knows about, including the system's local, default network. The table also lists the IP address of a gateway system for each known network. A *gateway* is a system that can receive outgoing packets and forward them one hop beyond the local network. See the following example:

```
Routing Table: IPv4
Destination           Gateway           Flags  Ref    Use   Interface
-------------------- -------------------- ----- -----  ------ ---------
default              198.51.100.10         UG    1    532        net0
224.0.0.0            203.0.113.100          U    1      0        net1
203.0.113.0          203.0.113.100          U    1      0        net1
127.0.0.1            127.0.0.1             UH    1     57         lo0
```

You can configure static or dynamic types of routing on an Oracle Solaris system. Both types can exist on a single system. A system that implements *dynamic routing* relies on routing protocols, such as the Routing Information Protocol (RIP) for IPv4 networks and RIP next generation (RIPng) protocol for IPv6 networks, to route network traffic, as well as update routing information in the table. With *static routing*, information is maintained manually by using the `route` command. See the `route(8)` man page.

When you configure routing for the local network or an autonomous system (AS), consider which type of routing to support on particular routers and hosts. The following table shows the different types of routing and networking scenarios for which each routing type is best applied.

Routing Type	Best Uses
Static	Small networks and hosts that get their routes from a default router and default routers that only need to know about one or two routers on the next few hops.
Dynamic	Larger internetworks, including routers on local networks with many hosts and hosts on large autonomous systems. Dynamic routing is the best option for systems on most networks.
Combined static and dynamic	Routers that connect a statically routed network and a dynamically routed network and border routers that connect an interior autonomous system with external networks. Combining both static and dynamic routing on a system is a common practice.

The topology that is described in "IPv4 Autonomous System Topology" in *Planning for Network Deployment in Oracle Solaris 11.4* combines both static and dynamic routing.

Note - Two routes that are going to the same destination does not automatically cause a system to perform load balancing or failover. If you need these capabilities, use IPMP. For more information, see Chapter 2, "About IPMP Administration" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

Using Rights Profiles to Perform Network Configuration

Oracle Solaris implements role-based access control (RBAC) to control system access. To perform tasks associated with network configuration, you must be assigned at least the Network Management profile. This profile is a superset that consists of other network-related profiles such as in the following partial list:

- Name Service Management for configuring name services.
- Network Wifi Management for configuring WiFi.
- Elastic Virtual Switch Administration for configuring the elastic virtual switch.
- Network Observability for accessing observability devices.

To obtain a complete list of the profiles in the Network Management profile, type:

```
$ profiles -p "Network Management" info
```

An administrator that has the `solaris.delegate.*` authorization can assign the Network Management profile to users to enable them to administer the network.

For example, an administrator assigns the Network Management rights profile to user `jdoe`. Before `jdoe` executes a privileged network configuration command, `jdoe` must be in a profile shell. The shell can be created by issuing the `pfbash` command. Or, `jdoe` can combine `pfexec` with every privileged command that is issued, such as `pfexec dladm`.

As an alternative, instead of assigning the Network Management profile directly to individual users, a system administrator can create a role that would contain a combination of required profiles to perform a range of tasks.

Suppose that a role `netadmin` is created with the profiles for network configuration as well as zone creation and configuration. User `jdoe` can issue the `su` command to assume that role. All roles automatically get `pfbash` as the default shell.

For more information about rights profiles, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

♦ ♦ ♦ CHAPTER 2

Configuring a System as a Router

This chapter describes how to configure your Oracle Solaris system as either an IPv4 or IPv6 router. It contains the following topics:

- "Configuring an IPv4 Router"
- "Configuring an IPv6 Router"
- "Configuring Multihomed Hosts"

For information about configuring routing on an Oracle Solaris system, see "Enabling Routing for Single-Interface Systems" on page 31.

Note - To configure the system as a router and issue commands described in this chapter, you must have the appropriate rights profile. See "Using Rights Profiles to Perform Network Configuration" on page 17.

Configuring an IPv4 Router

You can use the following procedure to configure a system with only one physical interface (by default, a host) as a router. You might configure a single-interface system as a router if the system serves as one endpoint on a PPP link, as explained in "Planning a Dial-Up PPP Link" in *Managing Serial Networks Using UUCP and PPP in Oracle Solaris 11.4*.

Note - PPP was removed in the Oracle Solaris 11.4 SRU 24 release.

▼ How to Configure an IPv4 Router

The following procedure assumes that you are configuring interfaces for the router after installing the router.

Before You Begin

After the router is physically installed on the network, configure the router to operate in local files mode. This configuration ensures that routers boot even if the network configuration server is down.

Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Configure the IP interfaces for the NICs on the system.**

   ```
   $ ipadm create-ip IP-interface
   ```

2. **Configure the IP interface with a valid IP address by choosing one of the following commands:**

   - **To configure a static address, type the following command:**

     ```
     $ ipadm create-addr -a address [interface | addr-obj]
     ```

   - **To configure a nonstatic address, type the following command:**

     ```
     $ ipadm create-addr -T address-type [interface | addr-obj]
     ```

   For detailed instruction about how to configure IP interfaces, see Chapter 3, "Configuring and Administering IP Interfaces and Addresses in Oracle Solaris" in *Configuring and Managing Network Components in Oracle Solaris 11.4*.

   Make sure that each IP interface is configured with the IP address of the network for which the system must route packets. Therefore, if the system serves the `192.0.2.0` and `203.0.113.0` networks, then one NIC must be configured for each network.

Caution - Make sure you are thoroughly knowledgeable about DHCP administration before configuring an IPv4 router to use DHCP.

3. **Add the host name and IP address of each interface to the `/etc/inet/hosts` file.**

   For example, assume that the names you assigned for the two interfaces of the router are `krakatoa` and `krakatoa-1`, respectively. The entries in the `/etc/inet/hosts` file are as follows:

   ```
   192.0.2.1      krakatoa        #interface for network 192.0.2.0
   203.0.113.1         krakatoa-1      #interface for network 203.0.113.0
   ```

4. **To configure the router to run in local files mode, set the appropriate SMF name service switch property.**

   For example:

   ```
   $ svccfg -s name-service/switch setprop config/netmask = astring: "files"
   ```

```
$ svccfg -s name-service/switch:default refresh
```

5. **If the router is connected to any subnetted network, add the network number and the netmask to the `/etc/inet/netmasks` file.**

   For example, for IPv4 address notation, such as `192.0.2.0`, type the following:

   ```
   192.0.2.0    255.255.255.0
   ```

6. **Enable IPv4 packet forwarding on the router.**

   ```
   $ ipadm set-prop -p forwarding=on ipv4
   ```

7. **(Optional) Start a routing protocol.**

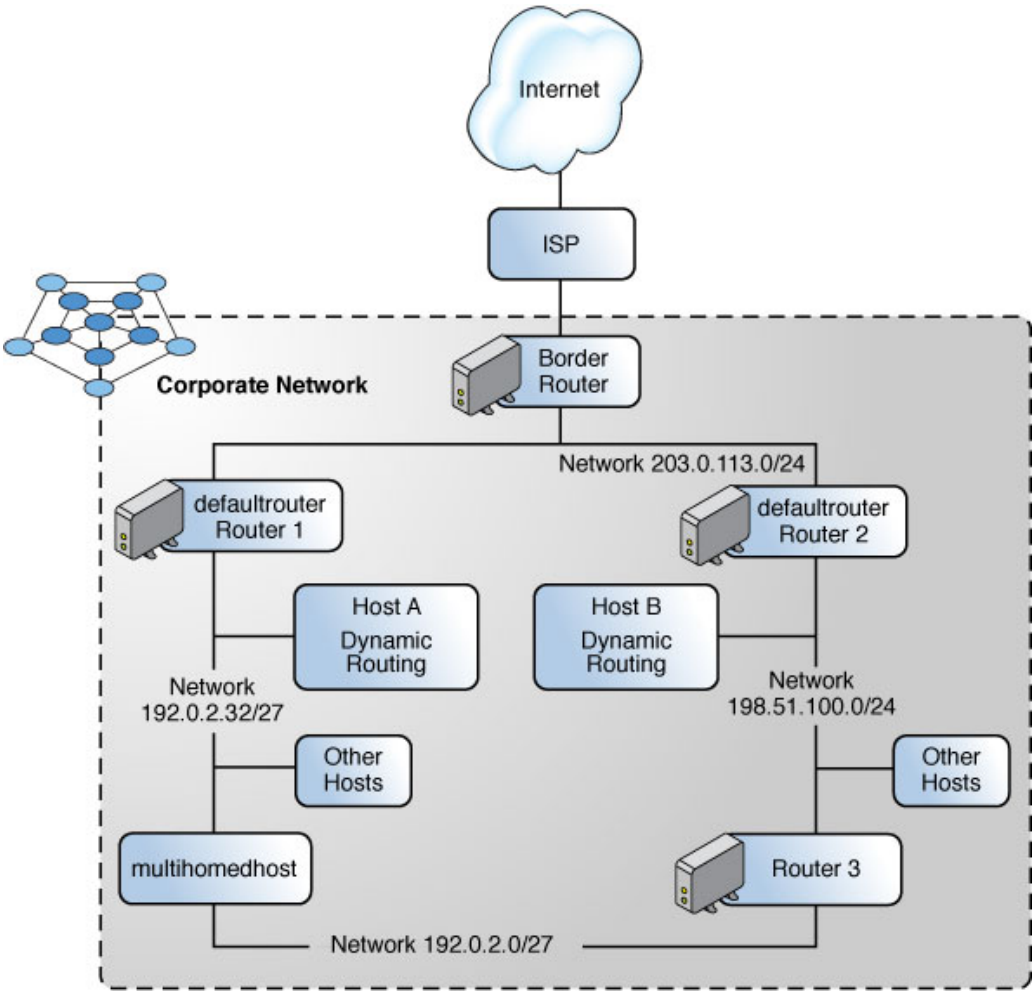
   ```
   $ svcadm enable route:default
   ```

   When you start a routing protocol, the routing daemon `/usr/sbin/in.routed` automatically updates the routing table, a process that is known as *dynamic routing*. For more information about the types of routing, see "Routing Tables and Routing Types" on page 16. For information about the `routeadm` command, see the `routeadm`(8) man page and for more information about the `ipadm` command, see the `ipadm`(8) man page.

   The Service Management Facility (SMF) Fault Management Resource Identifier (FMRI) associated with the `in.routed` daemon is `svc:/network/routing/route`.

Example 1 Configuring a System as a Router

This example is based on the following figure.



Router 2 contains two wired network connections, one connection to network `198.51.100.0` and one to network `203.0.113.0`. The example shows how to configure a system as a router (Router 2) of the `198.51.100.0` network.

The example begins with checking the status of the system's interfaces before proceeding with the configuration.

```
$ dladm show-link
LINK      CLASS     MTU     STATE   BRIDGE   OVER
net0      phys      1500    up      --       --
net1      phys      1500    up      --       --
net2      phys      1500    up      --       --
```

```
$ ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           198.51.100.10/24

$ ipadm create-ip net1
$ ipadm create-addr -a 203.0.113.10/24 net1
$ ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           198.51.100.10/24
net1/v4           static   ok           203.0.113.10/24

$ pfedit /etc/inet/hosts
192.0.2.1       localhost
198.51.100.10        router2        #interface for network 198.51.100
203.0.113.0          router2-out    #interface for network 203.0.113
$ pfedit /etc/inet/netmasks
198.51.100.0    255.255.255.0
203.0.113.0      255.255.255.0

$ ipadm set-prop -p forwarding=on ipv4
$ svcadm enable route:default
```

Next Steps To complete the default router configuration for the network,you must also do the following:

- Modify each system on the `198.51.100.0` network so that the system gets its routing information from the new default router. For more information, refer to "Creating Persistent (Static) Routes" in *Configuring an Oracle Solaris 11.4 System as a Router or a Load Balancer*.
- Define a static route to the border router in the routing table of Router 2. For more details, refer to "Routing Tables and Routing Types" on page 16. For more information about the `ipadm` command, see the `ipadm`(8) man page.

Configuring an IPv6 Router

This section describes the how to configure an IPv6 router.

`in.ripngd` Daemon, for IPv6 Routing

The `in.ripngd` daemon implements the Routing Information Protocol next-generation for IPv6 routers (RIPng). RIPng defines the IPv6 equivalent of RIP. When you configure an IPv6 router with the `routeadm` command and turn on IPv6 routing, the `in.ripngd` daemon implements RIPng on the router. For information about the supported options of RIPng, see `in.ripngd`(8).

Router Advertisement, Prefixes, and Messages

On multicast-capable links and point-to-point links, each router periodically sends to the multicast group a router advertisement packet that announces its availability. A system receives router advertisements from all routers, building a list of default routers. Routers generate router advertisements frequently enough so that systems learn of their presence within a few minutes. However, routers do not advertise frequently enough to rely on an absence of advertisements to detect router failure. A separate detection algorithm that determines neighbor unreachability provides failure detection.

Note - PPP was removed in the Oracle Solaris 11.4 SRU 24 release.

Router advertisements contain a list of subnet prefixes that is used to determine if a system is on the same link (on-link) as the router. The list of prefixes is also used for autonomous address configuration. Flags that are associated with the prefixes specify the intended uses of a particular prefix. Systems use the advertised on-link prefixes to build and maintain a list that is used to decide when a packet's destination is on-link or beyond a router. A destination can be on-link even though the destination is not covered by any advertised on-link prefix. In such instances, a router can send a redirect. The redirect informs the sender that the destination is a neighbor.

Router advertisements, and per-prefix flags, enable routers to inform systems how to perform stateless address autoconfiguration.

Router advertisement messages also contain Internet parameters, such as the hop limit, that systems should use in outgoing packets. Optionally, router advertisement messages also contain link parameters, such as the link MTU. This feature enables the centralized administration of critical parameters. The parameters can be set on routers and automatically propagated to all systems that are attached.

Nodes accomplish address resolution by sending to the multicast group a neighbor solicitation that asks the target node to return its link-layer address. Multicast neighbor solicitation messages are sent to the solicited-node multicast address of the target address. The target node

returns its link-layer address in a unicast neighbor advertisement message. A single request-response pair of packets is sufficient for both the initiator and the target nodes to resolve each other's link-layer addresses. The initiator includes its link-layer address in the neighbor solicitation.

▼ How to Configure an IPv6-Enabled Router

The following procedure assumes that you have already configured the system for IPv6. For the procedures, refer to Chapter 3, "Configuring and Administering IP Interfaces and Addresses in Oracle Solaris" in *Configuring and Managing Network Components in Oracle Solaris 11.4*.

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Configure IPv6 packet forwarding on all interfaces of the router.**

   ```
   $ ipadm set-prop -p forwarding=on ipv6
   ```

2. **Start the routing daemon.**

   The `in.ripngd` daemon handles IPv6 routing. Enable IPv6 routing by using the following command:

   ```
   $ svcadm enable ripng:default
   ```

3. **Create the `/etc/inet/ndpd.conf` file.**

   Specify the site prefix to be advertised by the router and other configuration information in the `/etc/inet/ndpd.conf` file. This file is read by the `in.ndpd` daemon, which implements the IPv6 Neighbor Discovery protocol.

   For a list of variables and allowable values, refer to the `ndpd.conf`(5) man page.

4. **Type the following text into the `/etc/inet/ndpd.conf` file:**

   ```
   ifdefault AdvSendAdvertisements true
   prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on
   ```

   This text tells the `in.ndpd` daemon to send out router advertisements over all interfaces of the router that are configured for IPv6.

5. **To configure the site prefix on the various interfaces of the router, add additional text to the `/etc/inet/ndpd.conf` file.**

   The text should be added in the following format:

`prefix` *global-routing-prefix*:*subnet ID*`/64` *interface*

In the following example, `/etc/inet/ndpd.conf` file configures the router to advertise the site prefix `2001:0db8:3c4d::/48` over the interfaces `net0` and `net1`.

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

if net0 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:15::0/64 net0

if net1 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:16::0/64 net1
```

6. Reboot the system.

The IPv6 router begins advertising on the local link any site prefix that is in the `ndpd.conf` file.

7. Display the interface configured for IPv6.

```
$ ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
net0/v4      static     ok      198.51.100.2/27
net1/v4      static     ok      198.51.100.35/27
net0/v6      addrconf   ok      fe80::203:baff:fe11:b115/10
lo0/v6       static     ok      ::1/128
net0/v6a     static     ok      2001:db8:3c4d:15:203:baff:fe11:b115/64
net1/v6      addrconf   ok      fe80::203:baff:fe11:b116/10
net1/v6a     static     ok      2001:db8:3c4d:16:203:baff:fe11:b116/64
```

In the output, each interface that was configured for IPv6 now has two addresses. The entry with the address object name such as *interface*`/v6` shows the link-local address for that interface. The entry with the address object name such as *interface*`/v6a` shows a global IPv6 address. In addition to the interface ID, this address includes the site prefix that you configured in the `/etc/ndpd.conf` file. Note that the designation `v6a` is a randomly defined string. You can define other strings to constitute the second part of the address object name provided that the *interface* reflects the interface over which you are creating the IPv6 addresses, for example, `net0/mystring`, `net0/ipv6addr`.

See Also

- To find out how to configure any tunnels from the routers that you have identified in your IPv6 network topology, refer to "Administering IP Tunnels" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.
- For information about configuring switches and hubs on your network, refer to the manufacturer's documentation.

Configuring Routing

This section discusses additional procedures to configure routing.

Creating Persistent (Static) Routes

In this Oracle Solaris release, the `route` command is the only way that you can manually configure routes, persistent or otherwise. For details, see the `route`(8) man page.

When adding routes persistently make sure that the routes do not already exist in the persistent configuration. If these routes already exist in the persistent configuration, the network routing tables could change without actually updating the persistent route. For example, typically in an Oracle Solaris setup, the system's default route is mapped to the system's primary interface. If you subsequently change the system's primary interface to another interface, then the system's default route should also be updated persistently. In this case, delete the current persistent route configuration prior to adding the new route. For more information, see "Troubleshooting Issues When Adding a Persistent Route" in *Troubleshooting Network Administration Issues in Oracle Solaris 11.4*.

The following list shows how you use the `route` command to manage routes:

- Use the `-p` option to add a persistent route:

  ```
  $ route -p add default ip-address
  ```

- Use the `-name` option to add a persistent route by specifying a name rather than destination and gateway:

  ```
  $ route -p add destination-address gateway-address -name name
  ```

- Use the `route -p show` command to display all of the persistent routes:

  ```
  $ route -p show
  ```

- To display the currently active routes on a system, use the `netstat` command as follows:

  ```
  $ netstat -rn
  ```

  See the `netstat`(8) and `route`(8) man pages.

▼ How to Add a Persistent Route by Specifying Destination and Gateway

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **View the current state of the routing table by using your regular user account.**

   ```
   $ netstat -rn
   ```

2. **(Optional) Flush the existing entries in the routing table.**

   ```
   $ route flush
   ```

3. **Add a persistent route.**

   ```
   $ route -p add -net network-address -gateway gateway-address
   ```

`-p`	Creates a static route that persists across system reboots. If you want the route to persist only for the current session, do not use the `-p` option.
`-net` *network-address*	Specifies that the route goes to the network with the address that is specified in *network-address*.
`-gateway` *gateway-address*	Indicates that the gateway system for the specified route has the IP address that is specified in *gateway-address*.

Example 2 Adding a Persistent Route by Specifying a Destination

The following example shows how to add a persistent (static) route to a router (Router 2). The static route is needed for the AS's border router, `203.0.113.150`. See the illustration used in Example 1, "Configuring a System as a Router," on page 21 of this particular setup.

You can view the routing table on Router 2 as follows:

```
$ netstat -rn
Routing Table: IPv4
Destination           Gateway         Flags   Ref     Use     Interface
-------------------- --------------- -------- ------ -------- -----------
default              198.51.100.10       UG     1     249         ce0
224.0.0.0            198.51.100.10        U     1       0         ce0
203.0.113.0          203.0.113.20         U     1      78        bge0
127.0.0.1            127.0.0.1           UH     1      57         lo0
```

```
Routing Table: IPv6
  Destination/Mask         Gateway         Flags Ref   Use    If
----------------------- ---------------- ------ ---- ------ -----
::1                           ::1            UH   2     0   lo0
```

The routing table indicates that there are two routes that Router 2 knows about. The default route uses Router 2's `198.51.100.10` interface as its gateway. The second route, `203.0.113.0`, was discovered by the `in.routed` daemon that is running on Router 2. The gateway for this route is Router 1 and it has the IP address `203.0.113.20`.

You can add a second route to network `203.0.113.0`, which has its gateway as the border router, as follows:

```
$ route -p add -net 203.0.113.0/24 -gateway 203.0.113.150
add net 203.0.113.0: gateway 203.0.113.150
```

The routing table now has a route for the border router, which has the IP address `203.0.113.150`.

```
$ netstat -rn
Routing Table: IPv4
Destination           Gateway         Flags   Ref     Use     Interface
-------------------- --------------- -------- ------ -------- -----------
default              198.51.100.10       UG     1     249         ce0
224.0.0.0            198.51.100.10        U     1       0         ce0
203.0.113.0          203.0.113.20         U     1      78        bge0
203.0.113.0          203.0.113.150        U     1     375        bge0
127.0.0.1            127.0.0.1           UH     1      57         lo0

Routing Table: IPv6
  Destination/Mask         Gateway         Flags Ref   Use    If
----------------------- ---------------- ------ ---- ------ -----
::1                           ::1            UH   2     0   lo0
```

▼ How to Specify a Name for a Persistent Route

When you add a persistent route, you can provide a name for it by using the `-name` option. You can specify any name other than `default`, which has a special meaning within the context of routing configuration. See the `route`(8) man page.

If you use the `-name` option, then you can refer to the route's name whenever you issue commands related to that route.

You cannot use the `-name` option to name an existing route or to change the name of a route. Instead, you must first delete the route and then add it again while specifying the `-name` option.

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Add a persistent route by specifying a name.**

   ```
   $ route -p add destination-address gateway-address -name name
   ```

   Note - The -name option is ignored for non-persistent routes.

2. **Display information about the route.**

   ```
   $ route get -name route-name
   ```

   You can also display information about the route as follows:

   ```
   $ route -p show
   ```

Example 3 Adding a Persistent Route by Specifying a Name

The following example shows how to add a persistent route, route1, by specifying a name.

```
$ route -p add 9.9.9.9 3.3.3.9 -name route1
persistent: route add 9.9.9.9 3.3.3.9 -name route1

$ route get -name route1
    route to : 9.9.9.9
        name : route1
 destination : 9.9.9.9
        mask : 255.255.255.25
     gateway : 3.3.3.9
   interface : net0
       flags : <UP,GATEWAY,HOST,DONE,STATIC>
recvpipe sendpipe ssthresh rtt,ms rttvar,ms hopcount mtu expire
0 0 0 0 0 0 1500 0
```

Example 4 Changing Information for a Persistent Route by Specifying the Route's Name

The following example shows how to modify information for a persistent route by specifying its name. In this example, the gateway information is changed. Note that the persistent route must be added by specifying the -name option initially.

```
$ route -p show
persistent: route add 9.9.9.9 3.3.3.9 -name route1
$ route change -name route1 9.9.8.8
change host -name route1 9.9.9.9: gateway 9.9.8.8
```

Example 5 Deleting a Persistent Route by Specifying the Route's Name

The following example shows how to delete a persistent route by specifying its name. Note that the persistent route must be added by specifying the `-name` option initially.

```
$ route -p delete -name route1
delete host -name route1 9.9.9.9: gateway 3.3.3.9: not in table
delete persistent host -name route1 9.9.9.9: gateway 3.3.3.9
```

If you do not specify the `-p` option with the `-name` option, the route is removed from the routing tables only, as shown in the following example:

```
$ route delete -name route1
delete host -name route1 9.9.9.9: gateway 3.3.3.9
```

Enabling Routing for Single-Interface Systems

In static routing, the host must rely upon the services of a default router for routing information. Thus, enabling dynamic routing that uses a routing protocol is the easiest way to manage routing on a system.

Sites with multiple routers and networks typically administer their network topology as a single routing domain or an *autonomous system* (AS). The procedures and examples in this section are based on the figure used in Example 1, “Configuring a System as a Router,” on page 21. In that figure, an AS is divided into three local networks: `203.0.113.0`, `198.51.100.0`, and `192.0.2.0`. The network has both routers and client systems. The types of routers that are on a network include the following: border routers, default routers, and packet-forwarding routers. The types of client systems on a network include both multihomed systems and single-interface systems. For more details about each of these components, see “IPv4 Autonomous System Topology” in *Planning for Network Deployment in Oracle Solaris 11.4*.

▼ How to Enable Dynamic Routing on a Single-Interface System

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See “Using Rights Profiles to Perform Network Configuration” on page 17.

1. **Configure one of the system's IP interfaces with an IP address for the network to which the system belongs.**

   For instructions, see “How to Configure an IPv4 Interface” in *Configuring and Managing Network Components in Oracle Solaris 11.4*.

2. **Delete all of the persistently defined routes from the system.**

You perform this step because the presence of any statically defined default routes prevents the enabling of dynamic routing during a system boot.

a. Determine all of the persistently defined default routes as follows:

```
$ route -p show
```

b. Delete each of the persistently defined routes.

```
$ route -p delete -net default -gateway ip-address
```

3. Ensure that packet forwarding is disabled.

```
$ ipadm set-prop -p forwarding=off ipv4
```

4. Enable routing on the system.

```
$ svcadm enable route:default
```

When you start a routing protocol, the routing daemon `/usr/sbin/in.routed` automatically updates the routing table. This process is known as *dynamic routing*.

Example 6 Running Dynamic Routing on a Single-Interface System

The following example shows how to configure dynamic routing for `hosta`, which is a single-interface system on the network `192.0.2.0`, as shown in the figure in Example 1, “Configuring a System as a Router,” on page 21. The system uses Router 1 as its default router. The example assumes that you have already configured the system's IP interface.

First, you would log into `hosta` with the appropriate rights. Then, you would remove all of the persistently defined routes from the system.

```
$ route -p show
persistent: route add default 198.51.100.10

$ route -p delete default 198.51.100.10
delete net default: gateway 198.51.100.10
delete persistent net default: gateway 198.51.100.10
```

Then you would disable packet forwarding as well as enable routing on the system.

```
$ ipadm set-prop -p forwarding=off ipv4
$ svcadm enable route:default
```

About IPv6 Routing

IPv6 routing is almost identical to IPv4 routing under CIDR. The only difference is that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms, such as Open Shortest Path First (OSPF), RIP, Inter-domain Routing Protocol (IDRP), and the Intermediate System to Intermediate System (IS-IS) protocol, can be used to route IPv6.

IPv6 also includes simple routing extensions that support the following powerful routing capabilities:

- Provider selection that is based on policy, performance, cost, and, so on
- Host mobility (route to current location)
- Auto-readdressing (route to new address)

You obtain these routing capabilities by creating sequences of IPv6 addresses that use the IPv6 routing option. An IPv6 source uses the routing option to list one or more intermediate nodes or topological group to be visited on the way to a packet's destination. This function is very similar to IPv4's loose source and record route option.

To make address sequences a general function, IPv6 hosts are required to reverse routes in a packet that a host receives. The packet must be successfully authenticated by using the IPv6 authentication header. The packet must contain address sequences to return the packet to its originator. This technique forces IPv6 host implementations to support the handling and reversal of source routes. The handling and reversal of source routes enables providers to work with hosts that implement various IPv6 capabilities, such as provider selection and extended addresses.

Configuring Multihomed Hosts

In Oracle Solaris, a system with more than one interface is considered a *multihomed host*. The interfaces of a multihomed host connect to different subnets, either on different physical networks or on the same physical network. For step-by-step instructions on creating a multihomed host, see "How to Create a Multihomed Host" on page 34.

On a system with multiple interfaces that connect to the same subnet, you must configure the interfaces into an IPMP group first. Otherwise, the system cannot be a multihomed host. For more information about IPMP, see Chapter 2, "About IPMP Administration" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

A multihomed host does not forward IP packets, but you can configure a multihomed host to run routing protocols. You typically configure the following types of systems as multihomed hosts:

- NFS servers, particularly those servers that function in a large deployment, can be attached to more than one network to share files among a large pool of users. These servers do not need to maintain routing tables.
- Database servers can have multiple network interfaces that provide resources to a large pool of users, just like NFS servers.
- Firewall gateways are systems that provide the connection between a company's network and public networks such as the Internet. Administrators set up firewalls as a security measure. When configured as a firewall, the host does not pass packets between the networks that are attached to the host's interfaces. However, the host can still provide standard TCP/IP services such as `ssh` to authorized users.

Note - When multihomed hosts have different types of firewalls on any of their interfaces, take care to avoid unintentional disruption of the host's packets. This problem arises particularly with stateful firewalls. One solution might be to configure stateless firewalls. For more information about firewalls, refer to "Firewall Systems" in *Securing Systems and Attached Devices in Oracle Solaris 11.4* or the documentation for your third-party firewall.

▼ How to Create a Multihomed Host

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Configure each additional network interface that was not configured as part of the installation process.**

   See "How to Configure an IPv4 Interface" in *Configuring and Managing Network Components in Oracle Solaris 11.4*.

2. **If multiple interfaces connect to the same subnet, configure those interfaces into an IPMP group.**

   See Chapter 3, "Administering IPMP" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

3. **Ensure that packet forwarding is disabled.**

   ```
   $ ipadm set-prop -p forwarding=off ipv4
   ```

4. **Enable dynamic routing for the multihomed host.**

```
$ svcadm enable route:default
```

Example 7 Configuring a Multihomed Host

The following example shows how to configure a multihomed host, as illustrated in the figure in "IPv4 Autonomous System Topology" in *Planning for Network Deployment in Oracle Solaris 11.4*. In this example, the system has the host name `hostc`. This host has two interfaces, which are connected to two different networks.

1. Display the status of the system's interfaces.

```
$ dladm show-link
LINK     CLASS     MTU     STATE   BRIDGE   OVER
net0     phys      1500    up      --       --
net1     phys      1500    up      --       --

$ ipadm show-addr
ADDROBJ         TYPE     STATE        ADDR
lo0/v4          static   ok           127.0.0.1/8
net0/v4         static   ok           192.0.2.3/27
```

   The `dladm show-link` command shows that `hostc` has two datalinks. However, only `net0` has been configured with an IP address. To configure `hostc` as a multihomed host, you must configure `net1` with an IP address on another subnet, for example, the `192.0.2.32/27` network.

2. Ensure that the underlying physical NIC of `net1` is physically connected to the network.

```
$ ipadm create-ip net1
$ ipadm create-addr -a 192.0.2.35/27 net1

$ ipadm show-addr
ADDROBJ         TYPE     STATE        ADDR
lo0/v4          static   ok           127.0.0.1/8
net0/v4         static   ok           192.0.2.3/27
net1/v4         static   ok           192.0.2.35/27
```

3. Add the `net1` interface to the `/etc/hosts` file.

```
$ vi /etc/inet/hosts
127.0.0.1           localhost
192.0.2.3/27        hostc    #primary network interface for host3
192.0.2.35/27       hostc-2  #second interface
```

4. Turn off packet forwarding if this service is running on the `hostc`.

```
$ ipadm set-prop -p forwarding=off ipv4
$ ipadm show-prop -p forwarding ipv4
PROTO PROPERTY              PERM CURRENT      PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding            rw   off          --           off          on,off
```

5. Enable dynamic routing for the multihomed host.

```
$ svcadm enable route:default
```

Implementing Symmetric Routing on Multihomed Hosts

By default, a multihomed host routes its network traffic based on the longest matching route to the traffic's destination in the routing table. When multiple routes of equal length to the destination exist, Oracle Solaris applies Equal-Cost Multi-Path (ECMP) algorithms to spread the traffic across those routes.

Spreading the traffic in this manner is not always ideal. For example, an IP packet might be sent through an interface on a multihomed host that is not on the same subnet as the IP source address in the packet. Furthermore, if the outgoing packet is in response to a certain incoming request, such as an ICMP echo request, the request and the response might not traverse the same interface. This type of traffic routing configuration is called *asymmetric routing*. If your Internet service provider (ISP) is implementing *ingress filtering*, as described in RFC 3704 (`https://www.rfc-editor.org/rfc/bcp/bcp84.txt`), an asymmetric routing configuration might cause an outgoing packet to be dropped by the ISP.

RFC 3704 intends to limit denial-of-service (DoS) attacks across the Internet. To comply with this intent, your network must be configured for symmetric routing. The IP `hostmodel` property enables you to meet this requirement. This property controls the behavior of IP packets that are received or transmitted through a multihomed host.

The `hostmodel` property can have one of three possible values:

`strong`	Corresponds to the strong end system (ES) model as defined in RFC 1122. This value implements symmetric routing.
`weak`	Corresponds to the weak ES model as defined in RFC 1122. With this value, a multihomed host uses asymmetric routing.
`src-priority`	Configures packet routing by using preferred routes. If multiple destination routes exist in the routing table, then the preferred routes are

those that use interfaces on which the IP source address of an outgoing packet is configured. If no such routes exist, then the outgoing packet will use the longest matching route to the packet's IP destination.

For example, you can implement symmetric routing of IP packets on a multihomed host as follows:

```
$ ipadm set-prop -p hostmodel=strong ipv4
$ ipadm set-prop -p hostmodel=strong ipv6
$ ipadm show-prop -p hostmodel ip
PROTO  PROPERTY   PERM  CURRENT  PERSISTENT  DEFAULT   POSSIBLE
ipv6   hostmodel  rw    strong   --          weak      strong,
                                                      src-priority,
                                                      weak
ipv4   hostmodel  rw    strong   --          weak      strong,
                                                      src-priority,
                                                      weak
```

♦ ♦ ♦ CHAPTER 3

Using Virtual Router Redundancy Protocol

Oracle Solaris provides an administrative tool that configures and manages the VRRP service. In addition to the existing standard Layer 2 VRRP, Oracle Solaris provides a proprietary Layer 3 VRRP to support VRRP over IPMP and InfiniBand interfaces and enhanced support for VRRP in zones.

This chapter describes both Layer 2 VRRP and proprietary Layer 3 VRRP in Oracle Solaris and covers the following topics:

Overview of VRRP

VRRP is an Internet standard protocol specified in RFC 5798 (`https://www.rfc-editor.org/rfc/rfc5798.txt`). It provides high availability of IP addresses, such as those that are used for routers or load balancers. Services that use VRRP are also referred to as VRRP routers even though the services provide functionality other than routing, such as load balancing. For information about how VRRP is used with load balancer to ensure high availability, see Chapter 7, "Configuring ILB for High Availability".

A VRRP router is a router that is running the VRRP. VRRP runs on each VRRP router and manages the state of the router. A system can have multiple routers on which VRRP is configured and each router belongs to a different virtual router. You can introduce virtual routers in a local area network (LAN) by using VRRP to provide failure recovery for a router.

The Layer 2 VRRP router uses the standard VRRP protocol and requires a unique virtual router MAC address. The virtual IP addresses always resolves to the same virtual MAC address. You

need to create a VRRP VNIC to get the unique virtual router MAC address. The proprietary Layer 3 VRRP feature in Oracle Solaris completely removes the need to configure unique VRRP virtual MAC addresses for VRRP routers, and thereby provides support for VRRP over IPMP and InfiniBand interfaces.

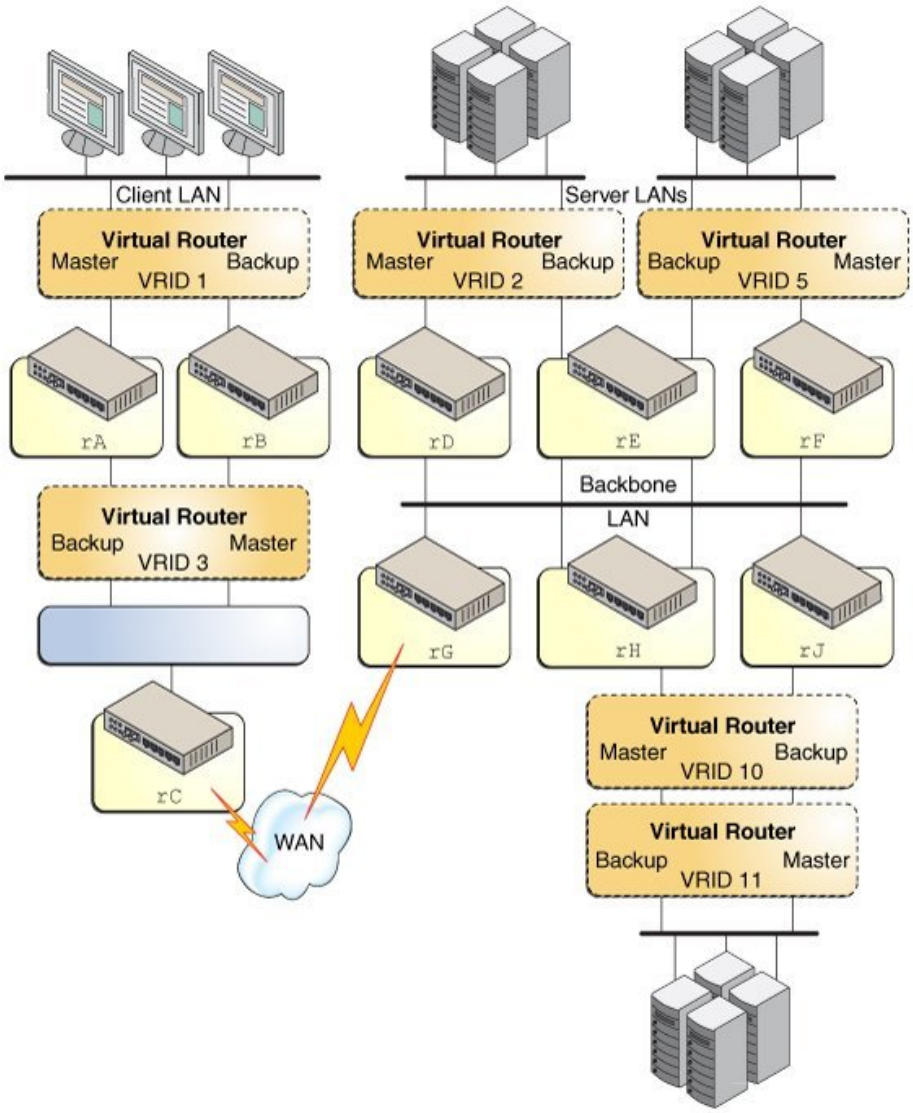
How VRRP Works

Note the following VRRP router terms:

- Router name – A system-wide unique identifier.
- Virtual Router ID (VRID) – A unique number used to identify a virtual router on a given network segment. VRIDs identify the virtual router within a LAN.
- Primary IP address – The source IP address of the VRRP advertisement.
- Virtual IP addresses (VRIP) – An IP address associated with a VRID from which other hosts can obtain network service. The VRIP is managed by the VRRP instances belonging to a VRID.
- Master router – A VRRP instance that performs the routing function for the virtual router at a given time. Only one *master router* is active at a time for a given VRID. The master router controls the IPv4 or IPv6 address or addresses that are associated with the virtual router. The virtual router forwards the packets that are sent to the IP address of the master router.
- Backup router – A VRRP instance for a VRID that is active but is not the master router is called a *backup router*. Any number of backup routers can exist for a VRID. A backup router assumes the role of a master router if the current master router fails.
- VRRP parameters – Includes priority, advertise interval, pre-empt mode, and accept mode.

The following VRRP load-sharing configuration figure shows that the multiple VRIDs can exist on a single router interface. The accompanying text explains the VRRP components that are used in the figure.

FIGURE 1 Load-Sharing Configuration of VRRP in a LAN



- Router `rA` is the master router for virtual router VRID 1 and the backup router for VRID 3. Router `rA` handles the routing of packets that are addressed to the virtual IP (VIP) address for VRID 1 and is ready to assume the routing role for VRID 3.
- Router `rB` is the master router for virtual router VRID 3 and the backup router for VRID 1. Router `rB` handles the routing of packets that are addressed to the VIP for VRID 3 and is ready to assume the routing role for VRID 1.
- Router `rC` does not have VRRP functions, but it uses the VRIP for VRID 3 to reach the client LAN subnet.
- Router `rD` is the master router for VRID 2. Router `rF` is the master router for VRID 5. Router `rE` is the backup router for both of these VRIDs. If `rD` or `rF` fails, `rE` becomes the master router for that VRID. Both `rD` and `rF` could fail at the same time. A VRRP router can be the master router for one or more VRIDs.
- Router `rG` is the wide area network (WAN) gateway for the backbone LAN. All of the routers attached to the backbone are sharing routing information with the routers on the WAN by using a dynamic routing protocol such as OSPF. VRRP is not involved in this aspect, although router `rC` advertises that the path to the client LAN subnet is through the VRIP of VRID 3.
- Router `rH` is the master router for VRID 10 and the backup router for VRID 11. Likewise, router `rJ` is the master router for VRID 11 and the backup router for VRID 10.

About the Layer 3 VRRP Feature

The proprietary Layer 3 Virtual Router Redundancy Protocol (L3 VRRP) feature in Oracle Solaris removes the need to configure unique VRRP virtual MAC addresses for VRRP routers. The L3 VRRP protocol does not conform to the standard VRRP specification. Instead of using a unique virtual MAC address among VRRP routers in the same virtual router, the L3 VRRP uses the gratuitous Address Resolution Protocol (ARP) messages and Neighbor Discovery Protocol (NDP) messages to refresh the mapping between the virtual IP addresses and the MAC address of the current master VRRP router. This implementation eliminates the need to create the VRRP VNIC. Thus, the L3 VRRP provides better support for VRRP over IPMP and InfiniBand interfaces as well as in zones.

Comparing Layer 2 and Layer 3 VRRP

The following table provides a comparison of Layer 2 and Layer 3 VRRP.

TABLE 3 Comparison of Layer 2 and Layer 3 VRRP

Feature	Layer 2 VRRP	Layer 3 VRRP
Creation of a VRRP VNIC	Required.	Not required because the virtual VRRP MAC address that is provided by the VRRP VNIC is not needed.
Support for IPMP	Not supported.	Supported. When a Layer 3 VRRP router is created over an IPMP group interface, each virtual IP address on the master router is associated with a MAC address of the active IPMP underlying interface according to the existing IPMP policy. If the failover occurs in the IPMP group, the L2 or L3 mappings are advertised by using the gratuitous ARP or NDP messages.
Zones support	There are issues running multiple VRRP routers that belong to the same virtual router in different zones. On a system with two or more VRRP routers that share the same VRRP virtual MAC address, the built-in virtual switch disrupts the normal flow of the VRRP advertisement packets to the VRRP router. For more information, see "Limitations of Layer 2 and Layer 3 VRRP" on page 44.	Supported.
InfiniBand support	Not supported.	Supported.
Unique virtual router MAC address	Requires a unique virtual router MAC address. The virtual IP addresses always resolve to the same virtual MAC address.	Not required. Uses the MAC address on which the VRRP router is created. The MAC address is different among all the VRRP routers that are in the same virtual router. The same MAC address is associated with the virtual IP addresses that are protected by this L3 VRRP router.
Configuring VRRP virtual IP addresses	Need to configure.	Need to configure.
Internet Control Message Protocol (ICMP) Redirects	Might be used when the L2 VRRP is running between group of routers. When an L2 VRRP router needs to use the ICMP redirects, it checks the destination MAC address (VRRP virtual MAC address) of the packets that need to be redirected. By using the destination MAC address, the L2 VRRP router determines the virtual router to which the packet was initially sent. Hence, the L2 VRRP router is able to select the source address and send the ICMP redirect message to the source node.	Need to disable ICMP redirects. When multiple VRRP routers are created over the same interface, they share the same MAC address. Therefore, the L3 VRRP cannot determine the destination MAC address.
Election of master router	The election of the master router is transparent to the system. When the master router changes, the switch that exists between the system and the router identifies	The election of the master router changes the Layer 2 mapping of the virtual IP addresses and the new mapping must be advertised by the gratuitous ARP or NDP messages.

Feature	Layer 2 VRRP	Layer 3 VRRP
	the new port to send the traffic by using its MAC learning capability.	
Failover time	Normal.	Might be longer because of the additional requirement of gratuitous ARP or the NDP messages when election of the master router changes.

Limitations of Layer 2 and Layer 3 VRRP

Both Layer 2 and Layer 3 VRRP have a common limitation that you must configure the Layer 2 and Layer 3 VRRP virtual IP addresses statically. You cannot auto-configure the VRRP virtual IP addresses by using either `in.ndpd` for IPv6 auto-configuration or `dhcpagent` for Dynamic Host Configuration Protocol (DHCP) configuration.

The Layer 2 VRRP feature has the following limitations:

- **Exclusive-IP Zone Support**

  When any VRRP router is created in an exclusive-IP zone, the VRRP service `svc:/network/vrrp/default` is enabled automatically. The VRRP service manages the VRRP router for that specific zone. However, support for an exclusive-IP zone is limited as follows:

  - Because a Virtual Network Interface Card (VNIC) cannot be created inside a non-global zone, you must create the VRRP VNIC in the global zone first. Then assign the VNIC to the non-global zone where the VRRP router resides. You can then create the VRRP router in the non-global zone by using the `vrrpadm` command.
  - On a single Oracle Solaris system, you cannot create two VRRP routers in different zones to participate with the same virtual router. Oracle Solaris does not allow you to create two VNICs with the same media access control (MAC) address.
- **Interoperations With Other Networking Features**
  - The L2 VRRP service cannot work on an IP network multipathing (IPMP) interface. VRRP requires specific VRRP MAC addresses but IPMP works completely in the IP layer. See Chapter 2, “About IPMP Administration” in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

    VRRP can be used on link aggregations in trunk or DLMP aggregation modes. See Chapter 2, “Configuring High Availability by Using Link Aggregations” in *Managing Network Datalinks in Oracle Solaris 11.4*.
  - The L2 VRRP service cannot work on an IP over Infiniband (IPoIB) interface.
- **Ethernet Over InfiniBand Support**

L2 VRRP does not support the Ethernet over InfiniBand (EoIB) interface. Because every L2 VRRP router is associated with a unique virtual MAC address, the VRRP routers participating with the same virtual router need to use the same virtual MAC address simultaneously, which is not supported by the EoIB interface. L3 VRRP overcomes this limitation as it uses a different MAC address among all the VRRP routers that exist on the same virtual router.

The Layer 3 VRRP feature has the following limitations:

- Using gratuitous ARP or NDP messages might result in a longer failover time during the election of the master router.

  L3 VRRP uses gratuitous ARP or NDP messages to advertise the new L2 or L3 mapping when the election of the master router changes. This additional requirement of using gratuitous ARP or NDP messages might result in a longer failover time. In some cases, if all the advertised gratuitous ARP or NDP messages are lost, it might take more time for a system to receive the refreshed ARP or NDP entry. Therefore, sending of packets to the new master router might be delayed.
- Unable to determine the destination MAC address when using ICMP redirects because the same destination MAC address is shared by multiple routers.

  You can use ICMP redirects when you are using VRRP among a group of routers in a network topology that is not symmetric. The IPv4 or IPv6 source address of an ICMPv4 redirect or ICMPv6 redirect must be the address used by the end system when making the next-hop routing decision.

  When an L3 VRRP router needs to use ICMP redirects, the L3 VRRP router checks the destination MAC address (VRRP virtual MAC address) of the packets that need to be redirected. Because the same destination MAC address is shared by multiple routers created over the same interface, the L3 VRRP router cannot determine the destination MAC address. Therefore, it might be useful to disable ICMP redirects when you use L3 VRRP routers. You can disable ICMP redirects by using the `send-redirects` public IPv4 and IPv6 protocol properties as follows:

  ```
  $ ipadm set-prop -m ipv4 -p send-redirects=off
  ```
- VRRP virtual IP addresses cannot be configured automatically either by `in.ndpd` or DHCP.

♦ ♦ ♦ CHAPTER 4

Configuring and Administering Virtual Router Redundancy Protocol

This chapter describes the tasks for configuring Layer 2 and Layer 3 VRRP. It covers the following topics:

- "Planning a VRRP Configuration"
- "Installing VRRP"
- "Configuring VRRP"
- "Use Case: Configuring a Layer 2 VRRP Router"
- "Use Case: Configuring a Layer 3 VRRP Router on an IPMP Interface"

Note - To configure VRRP and issue commands described in this chapter, you must have the appropriate rights profile. See "Using Rights Profiles to Perform Network Configuration" on page 17.

Planning a VRRP Configuration

Planning a Layer 2 or Layer 3 VRRP configuration involves the following steps:

1. Determining whether to configure an L2 VRRP or an L3 VRRP router.
2. (For L2 VRRP router only) Creating a VRRP VNIC.

   You can automatically create a VRRP VNIC by using the `-f` option of the `vrrpadm` command while you create the L2 VRRP router.
3. Creating a VRRP router.
4. Configuring the virtual IP address for the VRRP router.

Installing VRRP

You must install VRRP to use VRRP on your system.

▼ How to Install VRRP

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Verify whether the VRRP package is installed.**

   ```
   $ pkg info vrrp
   ```

2. **Install VRRP package if it is not installed.**

   ```
   $ pkg install vrrp
   ```

Configuring VRRP

Use the `vrrpadm` command to configure a VRRP router. The command creates persistent configurations. For reference, see the `vrrpadm`(8) man page.

Caution - When you use VRRP with the network firewall bundled with Oracle Solaris, make sure that the incoming or outgoing VRRP packets are allowed by firewall. The packets that are sent to the VRRP multicast address, `224.0.0.18/32`, must be allowed to enter or leave the system. Use the `pfctl -sr` command to see the rules that are currently loaded to the firewall. See also "Troubleshooting Issues With VRRP and the Oracle Solaris Bundled Packet Filter" in *Troubleshooting Network Administration Issues in Oracle Solaris 11.4*.

Creating a VRRP VNIC for Layer 2 VRRP

VNICs are virtual network interfaces configured on top of a system's physical network adapter and are essential components of network virtualization. A physical interface can have more

than one VNIC. For more information about VNICs, see *Managing Network Virtualization and Network Resources in Oracle Solaris 11.4*.

Each Layer 2 VRRP router requires a special VRRP VNIC which you create with the following command syntax.

```
$ dladm create-vnic [-t] [-R root-dir] -m vrrp -V VRID \
   -A {inet|inet6} [-v VLAN-ID] [-p prop=value[,...]] -l link VNIC
```

This command creates a VNIC with a virtual router MAC address that is defined by the VRRP specification. For the VNIC address type (`-m`), specify `vrrp`. The address family (`-a`) is either `inet` or `inet6`, which refers to either IPv4 or IPv6 addresses. For example:

```
$ dladm create-vnic -m vrrp -V 21 -A inet6 -l net0 vnic0
```

For more information, see the `dladm`(8) man page.

Note - You can create both the VRRP router and a VRRP VNIC at the same time by using the `vrrpadm` command. See the next section, *Creating a VRRP Router*.

Creating a VRRP Router

To create a VRRP router, use the following syntax:

```
$ vrrpadm create-router [-t] [-T {l2 | l3}] [-f] -V VRID -I ifname \
   -A [inet | inet6] [-a assoc-IPaddress] [-P primary-IPaddress] \
   [-p priority] [-i adv-interval] [-o flags] router-name
```

`-t` — Creates a temporary VRRP router that is removed at the next reboot.

`-T` *type* — Specifies the type of VRRP router which can be either `l2` (default) or `l3`.

`-f` — (L2 VRRP only) Creates a VRRP VNIC with an L2 VRRP router. A VRRP VNIC is created only if it does not already exist. The VNIC name uses the format `vrrp-`*VRID*`_`*ifname*`_[v4|v6]`.

`-V` *VRID* — The virtual router identifier that defines the VLAN when associated with the address family.

`-I` *ifname* — The interface on which the VRRP router is configured. For a Layer 2 VRRP, the interface can be a physical link, a VLAN, or an aggregation. For a Layer 3 VRRP, the interface can also include an IPMP interface,

	a DHCP managed interface, and an InfiniBand interface. This link determines the LAN in which this VRRP router is running.
`-A` *address-family*	The address family can be either `inet` or `inet6`, for IPv4 or IPv6 addresses, respectively.
`-a` *assoc-IPaddress*	Comma-separated list of IP addresses which can be specified as*IP-address*[/*prefix-length*], *hostname*[/*prefix-length*], or `linklocal`. `linklocal` applies only to IPv6 VRRP routers. The IPv6 address is configured based on the VRID of the associated virtual router. You can combine options `-a` and `-f` so that the VNIC is created and plumbed automatically.
`-P` *primary-IPaddress*	VRRP primary IP address that is used to send the VRRP advertisement.
`-p` *priority*	Priority of the specified VRRP router used for master router selection. The default value is `255`. The router with the highest priority value is selected as the master router.
`-i` *adv-interval*	Advertisement interval in milliseconds. The default value is `1000`.
`-o` *flags*	The preempt and accept flags of the VRRP router. The preempt flag is either `preempt` or `un_preempt`. The accept flag is either `accept` or `no_accept`. By default, the preempt and accept modes are set to `preempt` and `accept` respectively.
router-name	Unique identifier of this VRRP router. The permitted characters in a router name are alphanumeric (`a-z`, `A-Z`, `0-9`), and underscore (_). The maximum length of a router name is 31 characters.

EXAMPLE 8 Creating a Layer 2 VRRP Router

The following example shows how to create a router over a datalink `net0`.

```
$ dladm create-vnic -m vrrp -V 12 -A inet -l net0 vnic1
$ vrrpadm create-router -V 12 -A inet -p 100 -I net0 l2router1
$ vrrpadm show-router l2router1
NAME       VRID  TYPE   IFNAME AF   PRIO ADV_INTV MODE  STATE  VNIC
l2router1  12    L2     net0   IPv4 100  1000     e-pa- BACK   vnic1
```

An L2 VRRP router `l2router1` is created over the datalink `net0` with an IPv4 address family and VRID `12`. For information about the `vrrpadm show-router` command, see "Displaying Layer 2 and Layer 3 VRRP Router Configurations" on page 54.

EXAMPLE 9 Creating a Layer 3 VRRP Router

The following example shows how to create an L3 VRRP router over an IPMP interface named `ipmp0`.

```
$ vrrpadm create-router -V 6 -I ipmp0 -A inet -T l3 l3router1
$ vrrpadm show-router
NAME      VRID TYPE IFNAME AF   PRIO ADV_INTV MODE  STATE VNIC
l3router1 6    L3   ipmp0  IPv4 255  1000     eopa- INIT  --
```

An L3 VRRP router `l3router1` is created over the IPMP interface `ipmp0` with an IPv4 address family and VRID 6. For information about the `vrrpadm show-router` command, see "Displaying Layer 2 and Layer 3 VRRP Router Configurations" on page 54.

EXAMPLE 10 Creating a Temporary Layer 3 VRRP Router

The following example shows how to create a temporary L3 VRRP router.

```
$ vrrpadm create-router -t -V 8 -I net0 -A inet -T l3 l3router1
$ vrrpadm show-router
NAME      VRID TYPE IFNAME AF   PRIO ADV_INTV MODE  STATE VNIC
l3router1 8    L3   net0  IPv4 255  1000     eopa- INIT  --
```

Configuring the Virtual IP Address for Layer 2 and Layer 3 VRRP Routers

To configure the IP address for an L2 VRRP router, you must configure the virtual IP address of type `vrrp` over the VRRP VNIC that is associated with it.

To configure the virtual IP address for an L3 VRRP router, you must use an IP address of type `vrrp` on the same IP interface over which the L3 VRRP router is configured.

Note - To configure an IPv6 address, you must have created the VRRP VNIC or the L3 VRRP router by specifying the address family of the router as `inet6`.

To configure a virtual IP address for a VRRP router, use the following syntax:

```
$ ipadm create-addr [-t] -T vrrp [-a local=addr[/prefix-length]] \
   [-n router-name].... addr-obj | interface
```

`-t`	Specifies that the configured address is temporary and that the changes apply only to the active configuration.
`-T vrrp`	Specifies that the configured address is of the type `vrrp`.
`-n` *router-name*	The `-n` *router-name* option is optional for an L2 VRRP router because the VRRP router name can be derived from the VRRP VNIC interface on which the IP addresses are configured.

For more information, see the `ipadm`(8) man page.

Note - You can also configure virtual IP addresses by using the `-a` option with the `vrrpadm` command. For more information, see "Creating a VRRP Router" on page 49.

EXAMPLE 11 Configuring Virtual IP Address for an L2 VRRP Router

You can use the `vrrp` type IP address to configure the virtual IP addresses for an L2 VRRP router. The following example shows how to create the virtual IP address for `l2router1`.

```
$ ipadm create-ip vrrp_vnic1
$ ipadm create-addr -T vrrp -n l2router1 -a 192.0.2.8/27 vrrp_vnic1/vaddr1
```

The following example shows how to create an IPv6 link-local `vrrp` IP address for `V6vrrp_vnic1/vaddr1`.

```
$ ipadm create-ip V6vrrp_vnic1
$ ipadm create-addr -T vrrp V6vrrp_vnic1/vaddr1
```

To configure the IPv6 link-local `vrrp` type IP address for an VRRP router, you do not need to specify the local address. An IPv6 link-local `vrrp` type IP address is created based on the VRID of the associated VRRP router.

EXAMPLE 12 Configuring the Virtual IP Address for an L3 VRRP Router

The following example shows how to configure the virtual IP address for `l3router1`.

```
$ ipadm create-ip ipmp0
$ ipadm create-addr -T vrrp -n l3router1 -a 198.51.100.8/27 ipmp0/vaddr1
```

The following example shows how to configure an IPv6 link-local `vrrp` type IP address for the L3 VRRP router `l3V6router1`.

```
$ ipadm create-ip ipmp1
$ ipadm create-addr -T vrrp -n l3V6router1 ipmp1/vaddr0
```

Enabling and Disabling VRRP Routers

A VRRP router is enabled by default when you first create it. You can disable a VRRP router or all the VRRP routers on the system or zone at the same time by using the `vrrpadm disable-router` command. You can then re-enable a VRRP router or all the disabled VRRP routers on the system or zone at the same time by using the `vrrpadm enable-router` command.

The interface over which the VRRP router is created (specified with the `-I` option when the router is created with `vrrpadm create-router`) must exist when the router is enabled. Otherwise, the enable operation fails. For an L2 VRRP router, if the router's VRRP VNIC does not exist, the router is not effective. The syntax is as follows:

```
# vrrpadm enable-router [t] [-a] [router-name]
```

`-t` Specifies that enabling of the VRRP router is temporary and the change lasts only till the next reboot.

`-a` Specifies that all the disabled routers in the system or zone have to be re-enabled. If the `-t` option is specified, all the currently active VRRP routers are enabled, or else all the active and persistent VRRP routers are enabled. You must not specify *router-name* with this option.

router-name Specifies the name of the router to be re-enabled.

At times, you might need to temporarily disable a VRRP router to make configuration changes and then re-enable the router. The syntax for disabling a router is as follows:

```
$ vrrpadm disable-router [t] [-a] [router-name]
```

`-t` Specifies that disabling of the VRRP router is temporary and the change lasts only till the next reboot.

`-a` Specifies that all the routers in the system or zone have to be disabled. If the `-t` option is specified, all the currently active VRRP routers are disabled, or else all the active and persistent VRRP routers are disabled. You must not specify *router-name* with this option.

router-name Specifies the name of the router to be disabled.

Modifying a VRRP Router

The `vrrpadm modify-router` command changes the configuration of a specified VRRP router. You can modify the priority, the advertisement interval, the pre-empt mode, and the accept mode of the router. You can modify the router either temporarily to change only the active system configuration by specifying the `-t` option or permanently to change the persistent system configuration. The syntax is as follows:

```
$ vrrpadm modify-router [t] [-p priority] [-i adv-interval] [-o flags] router-name
```

where the `-t` option specifies that the modification is temporary and lasts only till the next reboot.

Displaying Layer 2 and Layer 3 VRRP Router Configurations

The `vrrpadm show-router` command shows the configuration and status of a specified VRRP router. For more information, see the `vrrpadm`(8) man page. The syntax is as follows:

```
$ vrrpadm show-router [-P | -x] [-p] [-S] [-o field[,...]] [router-name]
```

where the `-S` option is used to display the persistent configuration information of the given VRRP router. If the `-S` option is not specified, `vrrpadm show-router` displays the currently active VRRP router configuration.

EXAMPLE 13 Displaying a Layer 2 VRRP Router Configuration

The following examples show the `vrrpadm show-router` command output.

```
$ vrrpadm show-router vrrp1
NAME VRID TYPE  IFNAME  AF   PRIO ADV_INTV MODE  STATE  VNIC
vrrp1 1   L2    net1    IPv4 100  1000     e-pa- BACK   vnic1
```

`NAME` Name of the VRRP router.

`VRID` VRID of the VRRP router.

TYPE	The type of VRRP router, which is either L2 or L3.
IFNAME	The interface on which the VRRP router is configured. For an L2 VRRP router, the interface can be a physical Ethernet interface, a VLAN, or an aggregation.
AF	The address family of the VRRP router. It can be either IPv4 or IPv6.
PRIO	The priority of the VRRP router, which is used for master router selection.
ADV_INTV	The advertisement interval displayed in milliseconds.
MODE	A set of flags that are associated with the VRRP router and include the following possible values: ■ e – Specifies that the router is enabled. ■ p – Specifies that the mode is preempt. ■ a – Specifies that the mode is accept. ■ o – Specifies that the router is the virtual address owner.
STATE	The current state of the VRRP router. The possible values are: INIT (initialize), BACK (backup), and MAST (master).

In this example, information about the specified VRRP router vrrp1 is displayed.

```
$ vrrpadm show-router -x vrrp1
NAME  STATE PRV_STAT STAT_LAST VNIC  PRIMARY_IP     VIRTUAL_IPS
vrrp1 BACK  MAST     1m17s     vnic1 203.0.113.100  203.0.113.1
```

PRV_STAT	The previous state of the VRRP router.
STAT_LAST	Time since the last state transition.
PRIMARY_IP	The primary IP address selected by the VRRP router.
VIRTUAL_IPS	The virtual IP addresses configured on the VRRP router.

In this example, additional information about the router, such as the primary IP address selected by the VRRP router, virtual IP address configured on the VRRP router, and the previous state of the VRRP router is displayed.

```
$ vrrpadm show-router -P vrrp1
NAME  PEER           P_PRIO  P_INTV  P_ADV_LAST  M_DOWN_INTV
vrrp1 203.0.113.123  120     1000    0.313s      3609
```

PEER	The primary IP address of the peer VRRP router.
P_PRIO	The priority of the peer VRRP router, which is part of the advertisement received from the peer.
P_INTV	The advertisement interval (in milliseconds), which is part of the advertisements received from the peer.
P_ADV_LAST	Time since the last received advertisement from the peer.
M_DOWN_INTV	Time interval (in milliseconds) after which the master router is declared down.

The -P option is used only when the VRRP router is in the backup state.

EXAMPLE 14 Displaying the L3 VRRP Router on a System

```
$ vrrpadm show-router
NAME   VRID  TYPE  IFNAME  AF    PRIO  ADV_INTV  MODE   STATE   VNIC
l3vr1  12    L3    net1    IPv6  255   1000      eopa-  INIT    -
```

In this example, the L3 VRRP router l3vr1 is configured over the interface net1.

Displaying IP Addresses That Are Associated With VRRP Routers

You can display the IP address associated with a VRRP router by using the ipadm show-addr command. The ROUTER field in the output of the ipadm show-addr command displays the name of the VRRP router that is associated with a specific vrrp type IP address.

For the vrrp type IP address of an L2 VRRP, the name of the VRRP router is derived from the VRRP VNIC over which the IP address is configured. If you issue the ipadm show-addr command before you create the L2 router for a VRRP VNIC, the ROUTER field displays ?. For the vrrp type IP address of an L3 VRRP, the ROUTER field always displays the specified router name. For other types of IP addresses, the ROUTER field is not applicable and -- is displayed.

EXAMPLE 15 Displaying IP Addresses That Are Associated With VRRP Routers

```
$ ipadm show-addr -o addrobj,type,vrrp-router,addr
```

```
ADDROBJ              TYPE     VRRP-ROUTER   ADDR
lo0/v4               static   --               127.0.0.1/8
net1/p1              static   --               192.0.2.10/27
net1/v1              vrrp     l3router1        192.0.2.38/27
vrrp_vnic1/vaddr1    vrrp     l2router1        192.0.2.66/27
lo0/v6               static   --               ::1/128
```

In this example, `l3router1` is associated with the `vrrp` type IP address `192.0.2.38/27` and `l2router1` is associated with the `vrrp` type IP address `192.0.2.66/27`.

The output shows the following information:

`ADDROBJ` The name of the address object.

`TYPE` The type of the address object, which can be one of the following:

- `from-gz`
- `static`
- `dhcp`
- `addrconf`
- `vrrp`

`VRRP-ROUTER` The name of the VRRP router.

`ADDR` The numeric IPv4 or IPv6 address.

Deleting a VRRP Router

The `vrrpadm delete-router` command deletes a specified VRRP router. The syntax is as follows:

```
$ vrrpadm delete-router [-t] router-name
```

where the `-t` option specifies that the deletion is temporary. The temporary deletion lasts only until the next reboot.

Note - The VRRP VNIC, the `vrrp` type IP address, and the primary IP address that are created by using the `-f`, `-a`, `-P` options of the `vrrpadm create-router` command respectively are not deleted as a result of the `vrrpadm delete-router` command. You must explicitly delete them by using the corresponding `ipadm` and `dladm` commands.

Controlling Gratuitous ARP and NDP Messages

When a backup router becomes a master VRRP router, VRRP sets a flag on all the virtual IP addresses associated with the master router and therefore the virtual IP addresses are protected. If there are no conflicts for the virtual IP addresses, several gratuitous ARP and neighbor advertisement messages are sent to advertise the new mapping between the virtual IP address and the MAC address of the new master router.

To control the number of messages sent and the interval between the advertisement of messages, you can use the following IP protocol properties:

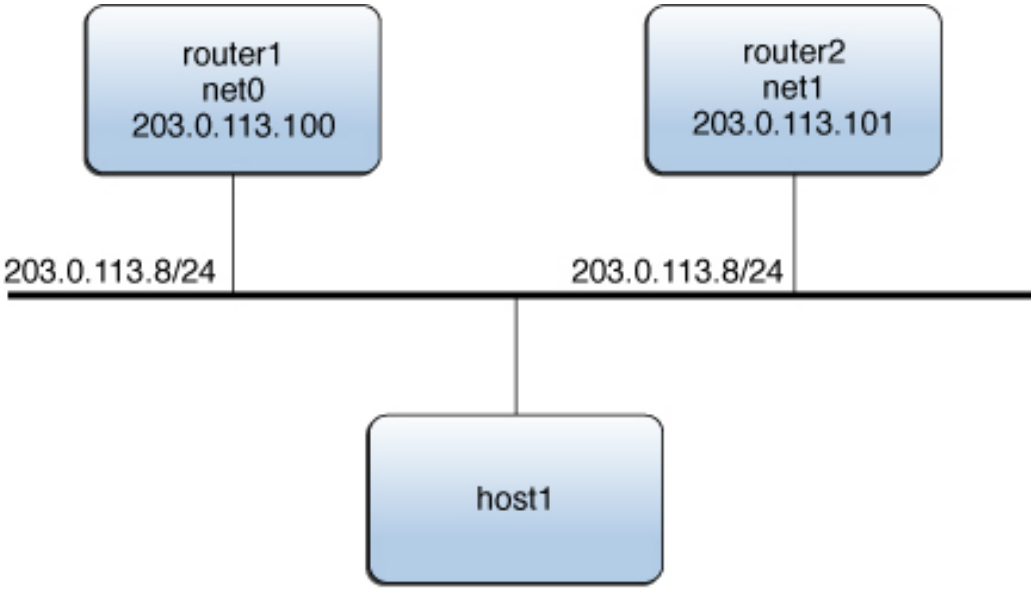
- `arp-publish-count`
- `arp-publish-interval`
- `ndp-unsolicit-count`
- `ndp-unsolicit-interval`

For more information about the IP protocol properties, see "IP Tunable Parameters Related to Duplicate Address Detection" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual*.

Use Case: Configuring a Layer 2 VRRP Router

The following figure shows a typical VRRP configuration.

FIGURE 2 Layer 2 VRRP Router



In this example, the IP address `203.0.113.8` is configured as the default gateway for `host1`. This IP address is the virtual IP address that is protected by the virtual router that is configured on two routers: `router1` and `router2`. At any given time, only one of the two routers serves as the master router, which assumes the responsibilities of the virtual router and forwards packets that come from `host1`.

Assume that the VRID of the virtual router is `12`. The following examples show the commands that are used to configure the example VRRP configuration on `router1` and `router2`. `router1` is the owner of the virtual IP address `203.0.113.8` and its priority is the default value (`255`). `router2` is the standby router whose priority is `100`.

On `router1`:

```
$ pkg install vrrp

$ dladm create-vnic -m vrrp -V 12 -A inet -l net0 vnic0
$ vrrpadm create-router -V 12 -A inet -I net0 vrrp1

$ ipadm create-ip vnic0
```

```
$ ipadm create-addr -T vrrp -a 203.0.113.8/24 vnic0/router1

$ ipadm create-ip net0
$ ipadm create-addr -a 203.0.113.100/24 net0/router1

$ vrrpadm show-router -x vrrp1
NAME  STATE   PRV_STAT STAT_LAST  VNIC   PRIMARY_IP     VIRTUAL_IPS
vrrp1 MASTER  INIT     14.444s    vnic0  203.0.113.100  203.0.113.8
```

On `router2`:

```
$ dladm create-vnic -m vrrp -V 12 -A inet -l net1 vnic1

$ vrrpadm create-router -V 12 -A inet -I net1 -p 100 vrrp2

$ ipadm create-ip vnic1
$ ipadm create-addr -T vrrp -a 203.0.113.8/24 vnic1/router2

$ ipadm create-ip net1
$ ipadm create-addr -a 203.0.113.101/24 net1/router2

$ vrrpadm show-router -x vrrp2
NAME  STATE    PRV_STAT  STAT_LAST  VNIC   PRIMARY_IP     VIRTUAL_IPS
vrrp2 BACKUP   INIT      2m32s      vnic1  203.0.113.101  203.0.113.8
```

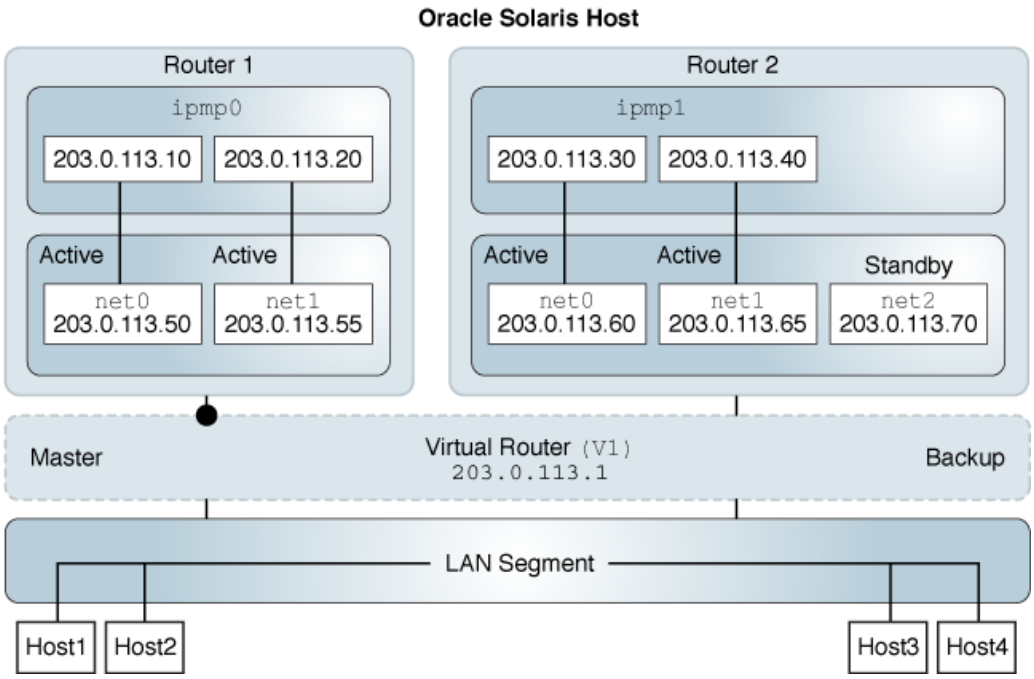
Using the configuration of `router1` as an example, you would configure at least one IP address over `net0`. This IP address of `router1` is the primary IP address, which is used to send the VRRP advertisement packets.

```
$ vrrpadm show-router -x vrrp1
NAME  STATE   PRV_STAT STAT_LAST  VNIC  PRIMARY_IP     VIRTUAL_IPS
vrrp1 MASTER  INIT     14.444s    vnic1 203.0.113.100  203.0.113.8
```

Use Case: Configuring a Layer 3 VRRP Router on an IPMP Interface

The following example shows the configuration of L3 VRRP router on IPMP interface. The configuration is based on the following scenario:

FIGURE 3 Layer 3 VRRP Router on an IPMP Interface



- There is a virtual router `V1`.
- In the virtual router V1, Router1 is the master router and Router2 is the backup.
- In Router1, the underlying interfaces `net0` and `net1` are configured into an IPMP group and all the underlying interfaces are assigned the test addresses.
- Master router is configured on the active-active IPMP interface, which is Router1.
- In Router2, the underlying interfaces `net0`, `net1`, and `net2` are configured into an IPMP group. The interface `net2` is configured as a standby interface.
- Backup router is configured on the active-standby IPMP interface.

On Router1:

```
$ pkg install vrrp

Router1$ ipadm create-ipmp ipmp0
Router1$ ipadm create-ip net0
```

```
Router1$ ipadm create-ip net1
Router1$ ipadm add-ipmp -i net0 -i net1 ipmp0

Router1$ ipadm create-addr -a 203.0.113.10/24 ipmp0
ipadm: ipmp0/v4
Router1$ ipadm create-addr -a 203.0.113.20/24 ipmp0
ipadm: ipmp0/v4a

Router1$ ipadm create-addr -a 203.0.113.50/24 net0
ipadm: net0/v4
Router1$ ipadm create-addr -a 203.0.113.55/24 net1
ipadm: net1/v4

Router1$ vrrpadm create-router -T L3 -V 1 -A inet -I ipmp0 -P 203.0.113.10 \
-a 203.0.113.1 -p 150 vrrp1

Router1$ vrrpadm show-router -x vrrp1
NAME    VRID TYPE IFNAME  AF    PRIO  ADV_INTV MODE  STATE   VNIC
vrrp1   1    L3   ipmp0   IPv4  150   1000     e-pa- MASTER  --
```

On Router2:

```
Router2$ ipadm create-ipmp ipmp1

Router2$ ipadm create-ip net0
Router2$ ipadm create-ip net1
Router2$ ipadm create-ip net2
Router2$ ipadm add-ipmp -i net0 -i net1 -i net2 ipmp1

Router2$ ipadm create-addr -a 203.0.113.30/24 ipmp1
ipadm: ipmp1/v4
Router2$ ipadm create-addr -a 203.0.113.40/24 ipmp1
ipadm: ipmp1/v4a

Router2$ ipadm create-addr -a 203.0.113.60/24 net0
ipadm: net0/v4
Router2$ ipadm create-addr -a 203.0.113.65/24 net1
ipadm: net1/v4
Router2$ ipadm create-addr -a 203.0.113.70/24 net2
ipadm: net2/v4

Router2$ ipadm set-ifprop -p standby=on net2

Router2$ vrrpadm create-router -T L3 -V 1 -A inet -I ipmp1 -P 203.0.113.30 \
-a 203.0.113.1 -p 100 vrrp2

Router2$ vrrpadm show-router -x
NAME    VRID TYPE IFNAME  AF    PRIO  ADV_INTV MODE  STATE   VNIC
vrrp2   1    L3   ipmp1   IPv4  100   1000     e-pa- BACKUP  --
```

The following commands show you how specifying a higher priority number can promote a router to become the master router. The priority of `vrrp2` is raised to 200, which is over `vrrp1`'s priority of 150.

```
Router2$ vrrpadm disable-router vrrp2
Router2$ vrrpadm modify-router -p 200 vrrp2
Router2$ vrrpadm enable-router vrrp2

Router2$ vrrpadm show-router -x vrrp1
NAME  VRID TYPE IFNAME  AF    PRIO  ADV_INTV MODE  STATE   VNIC
vrrp1 1    L3   ipmp0   IPv4  150   1000     e-pa- BACKUP  --

Router2$ vrrpadm show-router -x vrrp2
NAME  VRID TYPE IFNAME  AF    PRIO  ADV_INTV MODE  STATE   VNIC
vrrp2 1    L3   ipmp1   IPv4  200   1000     e-pa- MASTER  --
```

♦ ♦ ♦ CHAPTER 5

Overview of an Integrated Load Balancer

This chapter describes Integrated Load Balancer (ILB) components and the modes in which the ILB operates. It contains the following topics:

- "About the Integrated Load Balancer"
- "How ILB Works"
- "ILB Operation Modes"

For more information about ILB, see "About the Integrated Load Balancer" on page 65.

About the Integrated Load Balancer

Load balancers distribute network traffic across a number of servers. The distribution of a network's workload optimizes resource sharing and increases throughput and availability.

In Oracle Solaris, ILB provides Layer 3 and Layer 4 load-balancing capabilities. ILB operates at the network (IP) and transport (TCP/UDP) layers. ILB can be used to improve reliability and scalability, and to minimize the response time of network services.

Features of ILB

The key features of ILB include the following:

- Supports stateless Direct Server Return (DSR) and Network Address Translation (NAT) modes of operation for IPv4 and IPv6.
- Assists traffic and load distribution and server selection by using a set of algorithms for the two modes of operation.
- Enables ILB administration through the `ilbadm` command.

- Provides server monitoring capabilities through health checks.

The following table lists and describes the features of ILB that are available for different modes of operation.

TABLE 4 ILB Features

Features	Description	Mode of Operation
Client systems can ping virtual IP (VIP) addresses	ILB responds to ICMP echo requests from client systems to VIP addresses.	DSR and NAT
Add or remove servers from group without interrupting service	ILB dynamically adds or removes servers.	NAT
Configure session persistence ("stickiness")	Through ILB rules, applications can send the connections or packets from a client system to the same back-end server.	DSR and NAT
Perform connection draining	Servers can be shut down without disrupting active connections or sessions.	NAT
Load balance TCP and UDP ports	ILB balances the load on all ports on a given IP address across different sets of servers without requiring explicit rules for each port.	DSR and NAT
Specify independent ports for virtual services	ILB enables you to specify different destination ports across multiple servers in the same server group.	NAT
Load balance a simple port range	ILB balances loads on a range of ports on the VIP to a given server group.	DSR and NAT
Port range shifting and collapsing	Port range shifting and collapsing depend on the port range of a server in a load-balancing rule.	NAT

ILB Components

ILB is managed by the Service Management Facility (SMF) service `svc:/network/loadbalancer/ilb:default`. ILB has three major components:

- `ilbadm` command-line interface (CLI) – Enables you to configure load-balancing rules, perform optional health checks, and view statistics.
- `libilb` configuration library – Contains functionalities that `ilbadm` and third-party applications can use for ILB administration.
- `ilbd` daemon – Performs the following tasks:
  - Manages persistent configuration across reboots and package updates
  - Provides serial access to the ILB kernel module by processing the configuration information and sending it to the ILB kernel module for execution
  - Performs health checks and sends the results to the ILB kernel module so that the load distribution is properly adjusted

How ILB Works

ILB intercepts incoming requests from client systems, decides which back-end server should handle the requests based on load-balancing rules, and then forwards the requests to the selected server. ILB can also be used as a router for the back-end server. ILB performs optional health checks and provides the data for the load-balancing algorithms to verify whether the selected server can handle the incoming requests.

ILB Operation Modes

ILB supports stateless DSR and NAT modes of operation for IPv4 and IPv6 in single-legged and dual-legged topologies.

Direct Server Return Mode

In DSR mode, ILB balances the incoming requests to the back-end servers but allows the return traffic to the clients can bypass ILB. However, if ILB also serves as a router for a back-end server, then return traffic is routed through the system that is running ILB. ILB's currently implements a stateless DSR where TCP connection tracking is unavailable. Thus ILB does not save any state information of the processed packets except for basic statistics. ILB performance is comparable to the normal IP forwarding performance. The DSR mode is best suited for connectionless protocols.

Advantages:

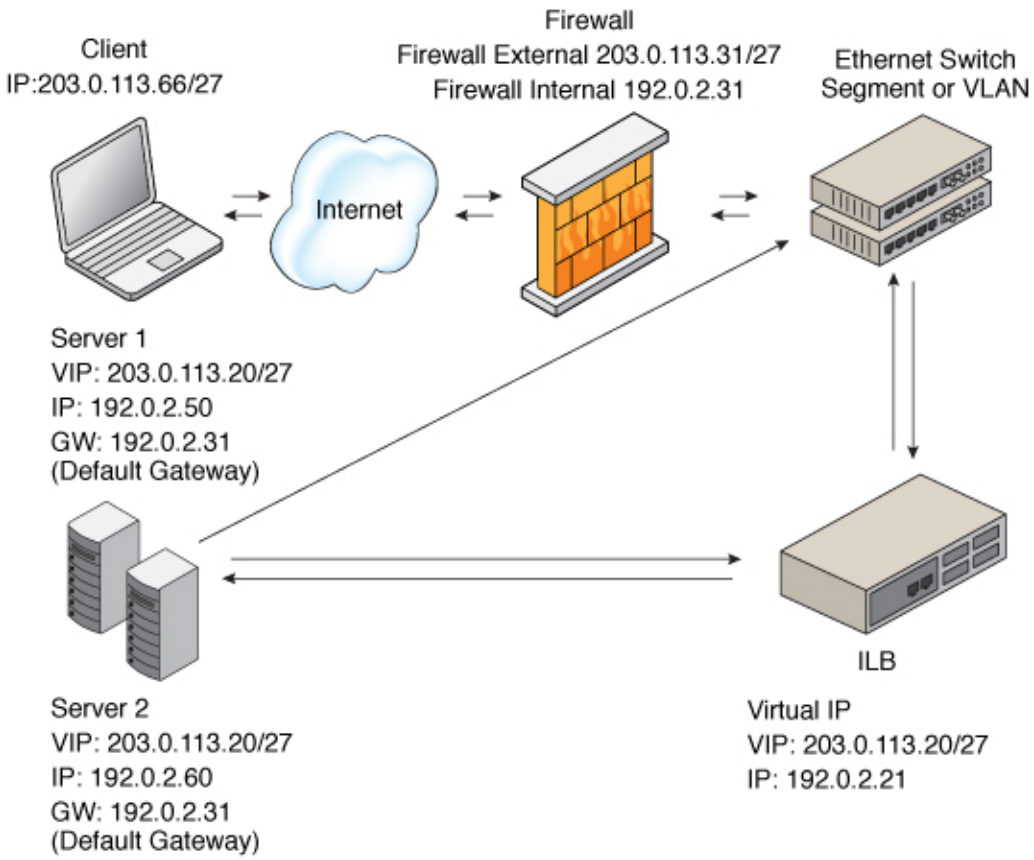
- DSR provides better performance than NAT because only the destination MAC address of packets is changed and servers respond directly to clients.
- Server and client transactions are fully transparent. The servers see a connection directly from the client IP address and reply to the client through the default gateway.

Disadvantages:

- The back-end server must respond to both its own IP address (for health checks) and the virtual IP address (for load-balanced traffic).
- Being stateless, adding or removing servers causes connection disruption.

The following figure shows the implementation of ILB in the DSR mode.

FIGURE 4 Direct Server Return Topology



In this figure, both back-end servers are in the same subnet (`192.0.2.0/27`) as the ILB box. The servers are also connected to the router so that they can reply directly to clients after receiving a request forwarded by the ILB box.

Network Address Translator Mode

ILB uses NAT in stand-alone mode strictly for load-balancing functionality. In this mode, ILB rewrites the header information and handles the incoming as well as the outgoing traffic. ILB

operates in both the half-NAT and full-NAT modes. However, full-NAT also rewrites the source IP address, making it appear to the server that all connections are originating from the load balancer. NAT does provide TCP connection tracking (meaning that it is stateful). The NAT mode provides additional security and is best suited for Hypertext Transfer Protocol (HTTP) or Secure Sockets Layer (SSL) traffic.

Advantages:

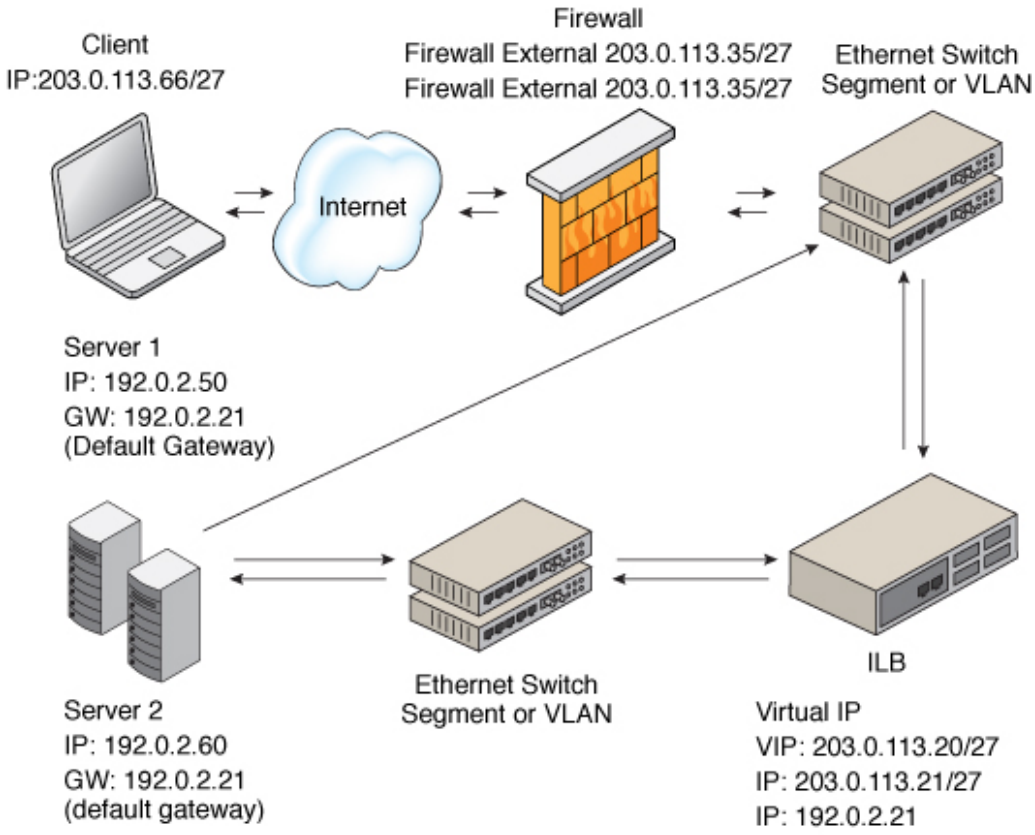
- Works with all back-end servers by changing the default gateway to point to the load balancer
- Adding or removing servers without connection disruption is possible because the load balancer maintains the connection state

Disadvantages:

- Slower performance than DSR because processing involves manipulation of the IP header and servers send responses to the load balancer
- All the back-end servers must use the load balancer as a default gateway

The general implementation of the NAT mode is as shown in the following figure.

FIGURE 5 Network Address Translation Topology



In this case, all requests to the VIP go through the ILB and are forwarded to the back-end servers. All the replies from the back-end servers pass through the ILB for NAT.

Half-NAT Load-Balancing Mode

In the half-NAT mode of the ILB operation, ILB rewrites only the destination IP address in the header of the packets. If you are using the half-NAT implementation, you cannot connect to a VIP address of the service from the same subnet on which the server resides. The following

table shows the IP addresses of the packets flowing between the client and ILB, and between ILB and back-end servers.

TABLE 5 Request Flow and Response Flow for the Half-NAT Implementation When the Server and Client Are on Different Networks

Request Flow Sequence	Source IP Address	Destination IP Address
1. Client –> ILB	Client	VIP of ILB
2. ILB –> Server	Client	Server
Response Flow Sequence	**Source IP Address**	**Destination IP Address**
3. Server –> ILB	Server	Client
4. ILB –> Client	VIP of ILB	Client

If you connect the client system to the same network as that of the servers, the intended server responds directly to the client, and the fourth step in the table does not occur. Therefore, the source IP address for the server response to the client is invalid. When the client sends a connection request to the load balancer, the response occurs from the intended server. From this point onwards, the IP stack of the client correctly drops all the responses. For this scenario, the request flow and response flow proceed as shown in the following table.

TABLE 6 Request Flow and Response Flow for the Half-NAT Implementation When the Server and Client Are on the Same Network

Request Flow Sequence	Source IP Address	Destination IP Address
1. Client –> ILB	Client	VIP of ILB
2. ILB –> Server	Client	Server
Response Flow Sequence	**Source IP Address**	**Destination IP Address**
3. Server –> Client	Server	Client

Full-NAT Load-Balancing Mode

In the full-NAT implementation of the ILB operation, the source and destination IP addresses are rewritten to ensure that the traffic goes through the load balancer in both directions. The full-NAT mode makes it possible to connect to the VIP from the same subnet that the servers are on.

The following table depicts the IP addresses of the packets flowing between a client and ILB, and between ILB and a back-end server by using the full-NAT mode. No special default route using the ILB box is required in the servers. Note that the full-NAT mode requires the administrator to set aside one IP address or a range of IP addresses to be used by ILB as source

addresses to communicate with the back-end servers. Assume that the addresses used belong to subnet C. In this scenario, ILB behaves as a proxy.

TABLE 7 Request Flow and Response Flow for the Full-NAT Implementation

Request Flow Sequence	**Source IP Address**	**Destination IP Address**
1. Client –> ILB	Client	VIP of ILB
2. ILB –> Server	Interface address of the load balancer (subnet C)	Server
Response Flow Sequence	**Source IP Address**	**Destination IP Address**
3. Server –> ILB	Server	Interface address of the ILB (subnet C)
4. ILB –> Client	VIP of ILB	Client

♦ ♦ ♦ CHAPTER 6

Configuring and Managing the Integrated Load Balancer

This chapter describes configuration and other administrative tasks related to ILB. It covers the following topics:

- "Preparing to Use ILB"
- "How to Deploy ILB"
- "Managing an ILB"
- "Use Case: Configuring an Integrated Load Balancer"
- "Displaying ILB Statistics"
- "Importing and Exporting Configurations"

Note - To configure the integrated load balancer and issue commands described in this chapter, you must have the appropriate rights profile. See "Using Rights Profiles to Perform Network Configuration" on page 17.

Preparing to Use ILB

ILB has kernel and userland installations. ILB kernel installation is performed automatically as a part of the Oracle Solaris installation. However, you must perform additional steps to make ILB functional.

▼ How to Deploy ILB

Before You Begin Ensure that your role has the appropriate rights to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Install the ILB userland package.**

   ```
   $ pkg install ilb
   ```

2. **Enable IP forwarding services.**

   ```
   $ ipadm set-prop -p forwarding=on ipv4
   ```

   Or:

   ```
   $ ipadm set-prop -p forwarding=on ipv6
   ```

3. **Enable the ILB service.**

   ```
   $ svcadm enable ilb
   ```

Disabling ILB

The previous procedure is reversed if you no longer want to use ILB. After the service is disabled, you can disable switch off packet forwarding if this functionality is not used in other network operations.

Managing an ILB

ILB management consists of defining the server groups, monitoring the health checks of ILB, and creating ILB rules.

Defining Server Groups and Back-End Servers

Back-end servers that are added to the group automatically obtain server IDs that are unique within the group. For reference, see the `ilbadm(8)` man page.

Creating an ILB Server Group

To create an ILB server group, first identify the servers that are to be included in the group. Servers can be specified by their host name or IP addresses and optional ports. Then issue the following command:

```
$ ilbadm create-servergroup -s servers=server1,server2,server3  servergroup
```

Unique server IDs prefixed with a leading underscore (_) are generated for each added server.

Note - A server that belongs to multiple server groups would have multiple IDs.

Adding Back-End Servers to an ILB Sever Group

To add a back-end server to a server group, issue the following command:

```
$ ilbadm add-server -s server=server1[,server2...] servergroup
```

Server specifications must include a host name or IP address and can optionally include a port or a range of ports. Server entries with the same IP address are disallowed within a server group.

Note - IPv6 addresses must be enclosed in square brackets.

EXAMPLE 16 Creating an ILB Server Group and Adding Back-End Servers

The following example shows how to simultaneously create a server group and its three back-end servers.

```
$ ilbadm create-servergroup -s \
   servers=192.0.2.11,192.0.2.12,192.0.2.13 webgroup
$ ilbadm show-servergroup
SGNAME         SERVERID            MINPORT MAXPORT IP_ADDRESS
webgroup       _webgroup.0         --      --      192.0.2.11
webgroup       _webgroup.1         --      --      192.0.2.12
webgroup       _webgroup.2         --      --      192.0.2.13
```

The following example shows how to create a server group and separately add three back-end servers.

```
$ ilbadm create-servergroup webgroup1
$ ilbadm add-server -s server=[2001:0db8:7::feed:6]:8080,\
   [2001:0db8:7::feed:7]:8080,[2001:0db8:7::feed:8]:8080 webgroup1
```

Enabling or Disabling a Back-End Server in an ILB Server Group

First identify the IP address, host name, or server ID of the back-end server you want to re-enable or disable. You must associate the server group with a rule before the servers in the server group can be enabled or disabled. In disabled servers, packet forwarding is halted.

A server can have multiple server IDs if it belongs to multiple server groups. You must specify a server ID to re-enable or disable the server for the specific rules that are associated with the server ID.

Use the following command syntax:

```
$ ilbadm disable-server|enable-server server1
```

To display the state of the server, type the following command:

```
$ ilbadm show-server [[-p] -o field[,field...]] [rulename]
```

Note - A server's enabled or disabled state is displayed only when the server group to which it belongs is associated with a rule.

Deleting a Back-End Server From an ILB Server Group

To remove a back-end server from one or more server groups, first Identify the server's ID:

```
ilbadm show-servergroup -o all
```

The server ID is a unique name for the IP address that is assigned to a system when the server is added to a server group.

Then, delete the server.

```
$ ilbadm remove-server -s server=server-ID server-group
```

This example removes the server `_sg1.2` from server group `sg1`.

```
$ ilbadm remove-server -s server=_sg1.2 sg1
```

If the server is being used by a NAT or half-NAT rule, disable the server first before removing it. See “Enabling or Disabling a Back-End Server in an ILB Server Group” on page 76. A

server being disabled enters the connection-draining state. Periodically check the NAT table by using the `ilbadm show-nat` command to see whether the server still has connections. After all the connections are drained, the server is no longer included in the `show-nat` command output. You can then remove the server.

If the `conn-drain` timeout value is set, the connection-draining state will be completed upon conclusion of the timeout period. The default value of `conn-drain` timeout is `0`, which means that the connection-draining waits until a connection is gracefully shut down.

Deleting ILB Server Groups

This section describes how to delete an ILB server group. You cannot delete a server group that is used by any active rule.

First, display all the available information about server groups.

```
$ ilbadm show-servergroup -o all
sgname      serverID       minport     maxport     IP_address
specgroup   _specgroup.0   7001        7001        192.0.2.18
specgroup   _specgroup.1   7001        7001        192.0.2.19
test123     _test123.0     7002        7002        192.0.2.18
test123     _test123.1     7002        7002        192.0.2.19
```

Then, delete the group. For example, based on the previous output, you would type:

```
$ ilbadm delete-servergroup test123
```

If the server group is in use by an active rule, the deletion fails.

Monitoring Health Checks in ILB

ILB provides the following optional types of server health checks:

- Built-in ping probes
- Built-in TCP probes
- Built-in UDP probes
- User-supplied custom tests that can run as health checks

By default, ILB does not perform any health checks. You can specify health checks for each server group when you create a load-balancing rule. Only one health check for every load-

balancing rule is allowed. Health checks on the server group that is associated with the enabled virtual service start automatically and are repeated periodically. The checks stop if the virtual service is disabled. The previous health check states are not preserved when the virtual service is re-enabled.

When you specify a TCP, UDP, or custom test probe for running a health check, ILB sends a ping probe, by default, to determine whether the server is reachable before it sends the specified TCP, UDP, or custom test probe to the server. If the ping probe fails, the corresponding server is disabled with the health check status `unreachable`. If the ping probe succeeds but the TCP, UDP, or custom test probe fails, the server is disabled with the health check status `dead`.

You can disable the default ping probe except for the UDP probe. The ping probe is always the default probe for UDP health checks.

Creating a Health Check

In the following example, two health check objects, `hc1` and `hc-myscript`, are created. The first health check uses the built-in TCP probe. The second health check uses a custom test, `/var/tmp/my-script`.

```
$ ilbadm create-healthcheck -h hc-timeout=3,\
   hc-count=2,hc-interval=8,hc-test=tcp hc1
$ ilbadm create-healthcheck -h hc-timeout=3,\
   hc-count=2,hc-interval=8,hc-test=/var/tmp/my-script hc-myscript
```

`hc-timeout` — Time limit beyond which a health check that does not complete is considered to have failed.

`hc-count` — Number of attempts to run the `hc-test` health check.

`hc-interval` — Time between consecutive health checks. To avoid sending probes to all servers at the same time, the actual interval is randomized between `0.5 * hc-interval` and `1.5 * hc-interval`.

`hc-test` — Type of health check. You can specify the built-in health checks, such as `tcp`, `udp`, and `ping` or an external health check, which has to be specified with the full path name.

Note - The port specification for `hc-test` is specified with the `hc-port` keyword in the `create-rule` subcommand. For more information, see the `ilbadm`(8) man page.

A user-supplied custom test can be a binary or a script.

- The test can reside anywhere on the system. You must specify the absolute path when you create the health check.

  When you specify the test, for example, `/var/tmp/my-script`, as part of the health check specification, the `ilbd` daemon forks a process and runs the test as follows:

  ```
  /var/tmp/my-script $1 $2 $3 $4 $5
  ```

`$1`	VIP (literal IPv4 or IPv6 address)
`$2`	Server IP (literal IPv4 or IPv6 address)
`$3`	Protocol (UDP, TCP as a string)
`$4`	Numeric port range (the user-specified value for `hc-port`)
`$5`	Maximum time (in seconds) that the test must wait before returning a failure. If the test runs beyond the specified time, it might be stopped, and the test is considered failed. This value is user-defined and specified in `hc-timeout`.

- The user-supplied test, does not have to use all the arguments, but it *must* return one of the following:
  - Round-trip time (RTT) in microseconds
  - 0 if the test does not calculate RTT
  - -1 for failure

By default, the health check test runs with the following privileges: `PRIV_PROC_FORK`, `RIV_PROC_EXEC`, and `RIV_NET_ICMPACCESS`.

If a broader privilege set is required, you must implement `setuid` in the test. For more details on the privileges, refer to the `privileges(7)` man page.

Listing Health Checks

To obtain detailed information about configured health checks, issue the following command:

```
$ ilbadm show-healthcheck
HCNAME      TIMEOUT COUNT   INTERVAL DEF_PING TEST
hc1         3       2       8        Y        tcp
hc2         3       2       8        N        /var/usr-script
```

Displaying Health Check Results

The `ilbadm list-hc-result` command shows results of all the health checks unless you specify a specific rule to be displayed.

The following example displays the health check results associated with a rule called `rule1`.

```
$ ilbadm show-hc-result rule1
RULENAME   HCNAME     SERVERID   STATUS   FAIL LAST      NEXT      RTT
rule1      hc1        _sg1:0     dead     10   11:01:19  11:01:27  941
rule1      hc1        _sg1:1     alive    0    11:01:20  11:01:34  1111
```

Note - The `show-hc-result` command displays the health check result only when the rules have associated health checks.

The `LAST` column of the output shows the time a health check ran, while the `NEXT` column shows the time the next health check will run.

Deleting a Health Check

The following example deletes a health check called `hc1`.

```
$ ilbadm delete-healthcheck hc1
```

Configuring ILB Rules

This section describes how you can use the `ilbadm` command to create, delete, and list the load-balancing rules.

ILB Algorithms

ILB algorithms control traffic distribution and provide various characteristics for load distribution and server selection.

ILB provides the following algorithms for both DSR and NAT modes of operation:

- Round-robin – The load balancer assigns the requests to a server group on a rotating basis. After a server is assigned a request, the server is moved to the end of the list.

- *src-IP* hash – The load balancer selects a server based on the hash value of the source IP address of the incoming request.
- *src-IP, port* hash – The load balancer selects a server based on the hash value of the source IP address and the source port of the incoming request.
- *src-IP, VIP* hash – The load balancer selects a server based on the hash value of the source IP address and the destination IP address of the incoming request.

Creating an ILB Rule

In ILB, a virtual service is represented by a load-balancing rule and is defined by the following parameters:

- Virtual IP address
- Transport protocol: TCP or UDP
- Port number (or a port range)
- Load-balancing algorithm
- Load-balancing mode (DSR, full-NAT, or half-NAT)
- Server group consisting of a set of back-end servers
- Optional server health checks that can be run for each server in the server group
- Optional port to use for health checks

Note - You can specify health checks on a particular port or on any port that the `ilbd` daemon randomly selects from the port range for the server.

- Rule name to represent a virtual service

Before you can create a rule, you must do the following:

- Create a server group that includes the appropriate back-end servers. See "Defining Server Groups and Back-End Servers" on page 74.
- Create a health check to associate the server health check with the rule. See "Creating a Health Check" on page 78.
- Identify the VIP, port, and optional protocol that are to be associated with the rule.
- Identify the operation you want to use (DSR, half-NAT, or full-NAT).
- Identify the load-balancing algorithm to be used. See "ILB Algorithms" on page 80.

To create an ILB rule, issue the `ilbadm create-rule` command together with specific parameter definitions from the previous list. For reference, see the `ilbadm`(8) man page.

```
$ ilbadm create-rule -e -i vip=IPaddr,port=port,protocol=protocol \
  -m lbalg=lb-algorithm,type=topology-type,proxy-src=IPaddr1-IPaddr2,\
```

pmask=*value* **-h hc-name=***hc1***-o servergroup=***sg rule1*

Note - The `-e` option enables the rule that is being created. Otherwise, rules you create are disabled by default.

EXAMPLE 17 Creating a Full-NAT Rule With Health Check Session Persistence

This example creates a health check called `hc1` and a server group called `sg1`. The server group consists of two servers, each with a range of ports. The last command creates and enables a rule called `rule1` and associates the rule to the server group and the health check. This rule implements the full-NAT mode of operation. Note that the creation of the server group and health check must precede the creation of the rule.

```
$ ilbadm create-healthcheck -h hc-test=tcp,hc-timeout=2,\
   hc-count=3,hc-interval=10 hc1
$ ilbadm create-servergroup -s server=192.0.2.10:6000-6009,192.0.2.11:7000-7009 sg1
$ ilbadm create-rule -e -p -i vip=203.0.113.10,port=5000-5009,\
   protocol=tcp -m lbalg=rr,type=NAT,proxy-src=192.0.2.34-192.0.2.44,pmask=27 \
   -h hc-name=hc1 -o servergroup=sg1 rule1
```

When you create persistent mapping, subsequent requests for connections, packets, or both, to a virtual service with a matching source IP address of the client are forwarded to the same back-end server. The prefix length in Classless Inter-Domain Routing (CIDR) notation is a value between `0-32` for IPv4 and `0-128` for IPv6.

When creating a half-NAT or a full-NAT rule, specify the value for the `connection-drain` timeout. The default value of `conn-drain` timeout is `0`, which means that connection draining keeps waiting until a connection is gracefully shut down.

Proxy source IP address is needed only for full NAT configuration. In full NAT mode of operation, ILB rewrites both the source and destination IP addresses of the packets coming from a client. The destination IP address is changed to one of the back-end servers' IP address. The source address is changed to be one of the proxy source addresses given in the `ilbadm` command line.

Proxy source address is needed because only a maximum of 65535 connections exist between one source address and one back-end server that is using one service port at any point of time. This limit becomes a bottleneck in load balancing. The list of proxy source addresses enables ILB to overcome this bottleneck because ILB has a number of source addresses to use.

Using proxy source address also avoids the problem of address conflict between ILB and the system, whether virtual or not, where ILB is running. Some network configurations require that the source address used by NAT be completely different from the address used by the system, whether virtual or not, where ILB is running.

Listing ILB Rules

To list the configuration details of a rule, issue the following command. If no rule name is specified, information is provided for all rules.

```
$ ilbadm show-rule
RULENAME        STATUS   LBALG           TYPE    PROTOCOL VIP           PORT
rule-http       E        hash-ip-port    NAT     TCP      203.0.113.1      80
rule-dns        D        hash-ip         NAT     UDP      203.0.113.1      53
rule-abc        D        roundrobin      NAT     TCP      2001:db8::1   1024
rule-xyz        E        ip-vip          NAT     TCP      2001:db8::1   2048-2050
```

Deleting an ILB Rule

You use the `ilbadm delete-rule` command to delete a rule. Add the `-a` option to delete all rules.

```
$ ilbadm delete-rule rule1
```

Use Case: Configuring an Integrated Load Balancer

This sample case describes how to set up ILB to use a half-NAT topology to load balance traffic among two servers. It assumes the following:

- Two servers have the IP addresses `192.0.2.50` and `192.0.2.60`, respectively.
- The name of the server group is `srvgrp1`.
- The back-end servers are set up to use ILB as the default router in this scenario.

In the example, after the applications are started on the two servers, a simple TCP level health check `hc-srvgrp1` is created that probes for the availability of the server application. If the server is not reachable after 3 tries, the server is marked as `dead`.

```
$ ilbadm create-sg -s servers=192.0.2.50,192.0.2.60 srvgrp1
```

FirstServer`$ route add -p default 192.0.2.33`
SecondServer`$ route add -p default 192.0.2.33`

At this stage, you would start the applications on both servers.

The next two commands create the health check and the load balancing rule.

```
$ ilbadm create-hc -h hc-test=tcp,hc-timeout=3, \
   hc-count=3,hc-interval=60 hc-srvgrp1

$ ilbadm create-rule -e -p -i vip=203.0.113.20,port=5000 -m \
   lbalg=rr,type=half-nat,pmask=27 \
   -h hc-name=hc-srvgrp1 -o servergroup=srvgrp1 rule1_rr
```

For additional examples of how to configure and deploy ILB, refer also to the following sources:

- Deploying the Oracle Solaris Integrated Load Balancer in 60 Minutes (https://www.oracle.com/technical-resources/articles/solaris11/deploy-solaris-integrated-load-balancer.html)
- How to Set Up a Load-Balanced Application Across Two Oracle Solaris Zones (https://www.oracle.com/technical-resources/articles/solaris/loadbalancedapp.html)

Displaying ILB Statistics

This section describes how to use the `ilbadm` command to obtain information such as the printing statistics for a server or statistics for a rule. You can also display NAT table information and the session persistence mapping table.

Displaying ILB Statistics

Use the `ilbadm show-statistics` command to view load distribution details as shown in the following example.

```
$ ilbadm show-statistics
PKT_P   BYTES_P   PKT_U   BYTES_U   PKT_D   BYTES_D
9       636        0        0        0        0
```

The output displays the number of packets processed, unprocessed, or dropped, and their respective sizes in bytes.

Displaying the NAT Connection Table

The following example displays five entries from the NAT connection table.

```
$ ilbadm show-nat 5
UDP: 192.0.2.0 > IP2 >>> IP3 . IP4
UDP: 198.51.100.0 > IP2 >>> IP3 . IP4
UDP: 203.0.113.0. > IP2 >>> IP3 . IP4
```

`UDP`	Transport protocol used in this entry
`IP2`	VIP and port
`IP3`	In half-NAT mode, the client's IP address and port. In full-NAT mode, the client's IP address and port.
`IP4`	Back-end server's IP address and port.

When you display the NAT connection table, the relative positions of elements in consecutive runs of the command are not significant. For example, if you issue the command `ilbadm show-nat 10` twice, you might not see the same 10 items each time, especially on a busy system. If a count value is not specified, the entire NAT connection table is displayed.

Displaying the Session Persistence Mapping Table

To display the session persistence mapping table, use the `ilbadm show-persist` command.

For example, to display five entries from the session persistence mapping table, you would type the following:

```
$ ilbadm show-persist 5
rule: 192.0.2.0 → IP2
rule: 198.51.100.0 → IP2
rule: 203.0.113..0 → IP2
```

rule	Rule that the persistence entry is tied to.
`IP2`	Back-end server's IP address.

Importing and Exporting Configurations

The `export` and `import` subcommands are used to move a configuration from one system to another. For example, to set up a back up of ILB using VRRP to have a active-passive

configuration, you can just export the configuration to a file and the import it in the back up system.

Unless specifically instructed otherwise, the `ilbadm import` command deletes the existing configuration before importing. Omission of a file name instructs the command to read from `stdin` or write to `stdout`.

The following example exports the current configuration into the file `/var/tmp/ilb_config` in a format suitable for importing by using the `import` command.

```
$ ilbadm export-config /var/tmp/ilb_config
```

To import an ILB configuration stored in that same file, you would type:

```
$ ilbadm import-config /var/tmp/ilb_config
```

The import operation overwrites the existing configuration.

♦ ♦ ♦ CHAPTER 7

Configuring ILB for High Availability

This chapter describes the high availability (HA) configuration of ILB by using VRRP. The configuration is illustrated in three scenarios as follows:

- "Configuring ILB for High Availability By Using the DSR Topology"
- "Configuring ILB for High Availability by Using the Half-NAT Topology"

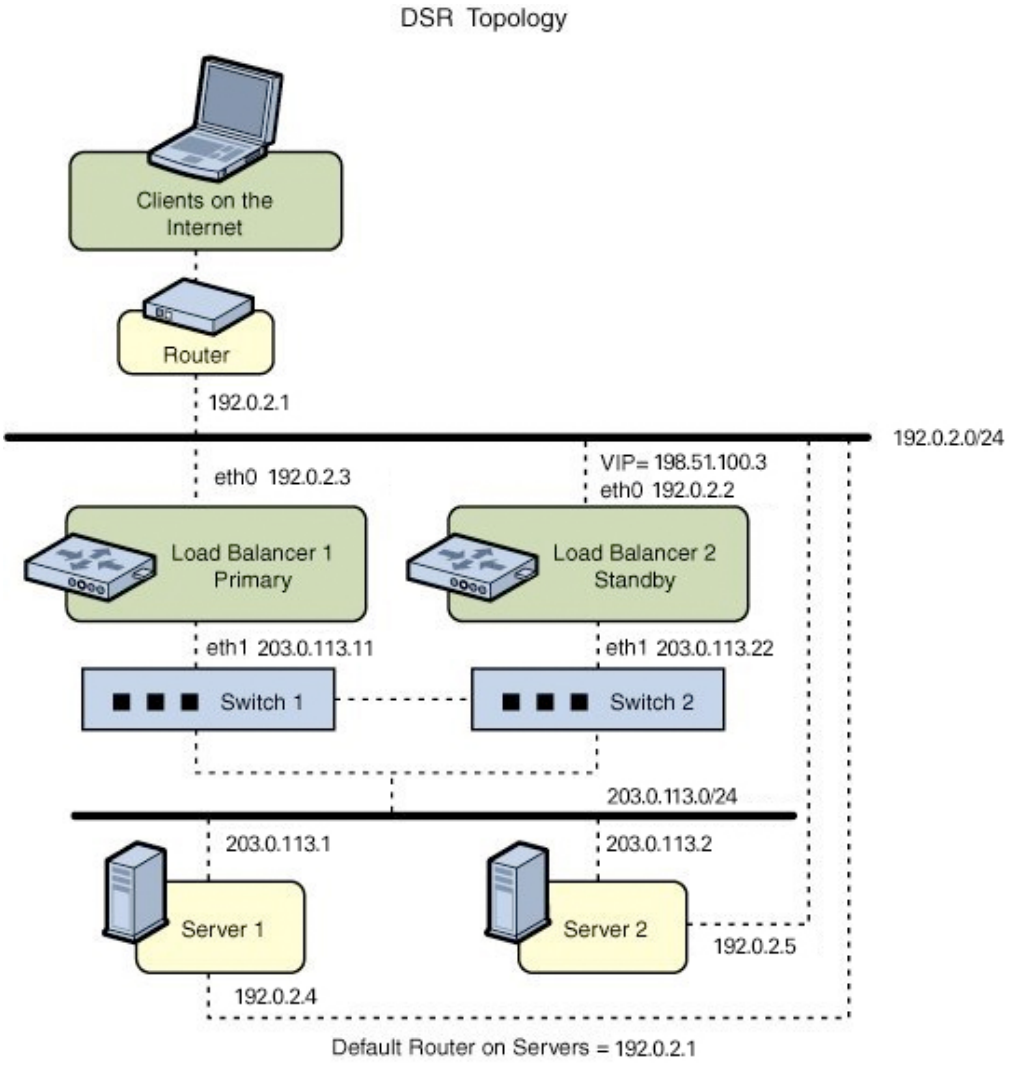
Note - To configure high availability for the integrated load balancer and to issue commands described in this chapter, you must have the appropriate rights profile. See "Using Rights Profiles to Perform Network Configuration" on page 17.

Configuring ILB for High Availability By Using the DSR Topology

This case uses two load balancers to be primary and standby load balancers. The virtual IP address of an ILB rule acts as the virtual router IP address.

The following figure shows the DSR topology of this scenario.

FIGURE 6 ILB for HA Configuration by Using DSR Topology



All VIPs on Load Balancers are configured on interfaces facing subnet 192.0.2.0/24

How to Configure ILB for High Availability by Using the DSR Topology

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Configure both the primary and standby (passive) load balancers to have the same set up.**

```
$ ilbadm create-servergroup -s server=203.0.113.1,203.0.113.2 sg1
$ ilbadm create-rule -i vip=198.51.100.3,port=9001 \
   -m lbalg=hash-ip-port,type=DSR -o servergroup=sg1 rule1
```

2. **Configure Load Balancer 1 to serve as the primary load balancer.**

```
LB1$ dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1$ vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1$ ipadm create-ip vnic1
LB1$ ipadm create-addr -d -a 198.51.100.3/24 vnic1
```

   The priority of the `vrrp1` router is set to be `255`. The priority value makes the router the master router and hence the active load balancer.

3. **Configure Load Balancer 2 to serve as the standby load balancer.**

```
LB2$ dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2$ vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2$ ipadm create-ip vnic1
LB2$ ipadm create-addr -d -a 198.51.100.3/24 vnicl
```

   The preceding configuration provides protection against the following failure scenarios:

   - If Load Balancer 1 fails, Load Balancer 2 becomes the primary load balancer. Load balancer 2 then takes over address resolution for the VIP `198.51.100.3` and handles all the packets from clients with the destination IP address `198.51.100.3`.

     When Load Balancer 1 recovers, Load Balancer 2 returns to standby mode.
   - If one or both of Load Balancer 1's interfaces fail, Load Balancer 2 takes over as the primary load balancer. Load Balancer 2 then takes over address resolution for VIP `198.51.100.3` and handles all the packets from clients with the destination IP address `198.51.100.3`.

     When both of Load Balancer 1's interfaces are healthy, Load Balancer 2 returns to standby mode.

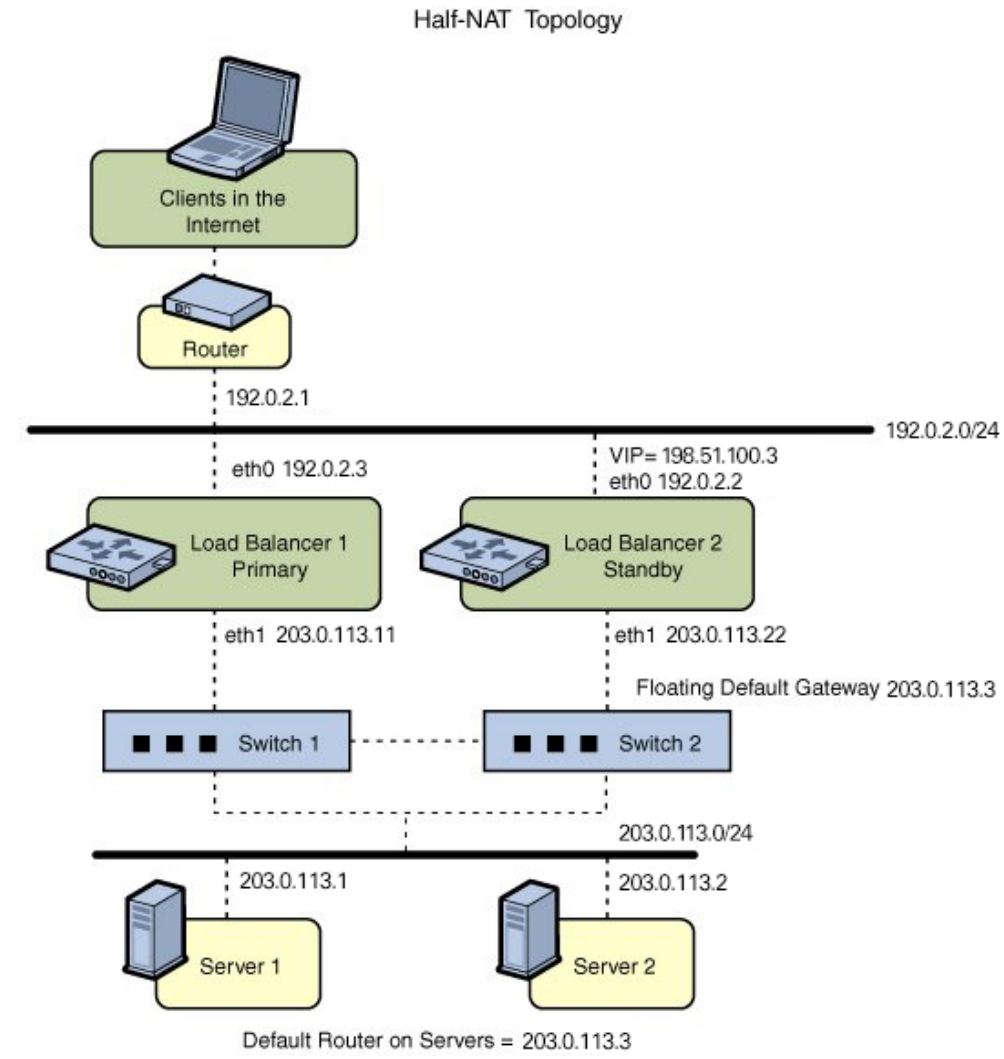
Configuring ILB for High Availability by Using the Half-NAT Topology

This section describes how to set up highly available ILB connections in a half-NAT topology. The scenario also uses two load balancers as primary and standby load balancers, respectively.

Note - The current implementation of ILB does not synchronize primary and standby load balancers. When the primary load balancer fails and the standby load balancer takes over, the existing connections fail. However, HA without synchronization is still valuable under circumstances when the primary load balancer fails.

The following figure shows the configuration.

FIGURE 7 ILB for HA Configuration By Using Half-NAT Topology



All VIPs on Load Balancers are configured on interfaces facing subnet 192.0.2.0/24.

How to Configure ILB for High-Availability by Using the Half-NAT Topology

Before You Begin Ensure that your role has the appropriate rights profile to perform this procedure. See "Using Rights Profiles to Perform Network Configuration" on page 17.

1. **Configure both the primary and standby load balancers.**

```
$ ilbadm create servergroup -s server=203.0.113.1,203.0.113.2 sg1
$ ilbadm create-healthcheck -h hc-timeout=4,hc-count=3,hc-interval=6,hc-test=tcp hc1
$ ilbadm create-rule -ep -i vip=198.51.100.3,port=9001-9006,protocol=udp \
-m lbalg=roundrobin,type=HALF-NAT,pmask=24 -h hc-name=hc1,hc-port=9006 \

-t conn-drain=70,nat-timeout=70,persist-timeout=70 -o servergroup=sg1 rule1
```

2. **Configure Load Balancer 1 to serve as the primary load balancer.**

```
LB1$ dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1$ ipadm create-ip vnic1
LB1$ ipadm create-addr -d -a 198.51.100.3/24 vnic1
LB1$ vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1$ dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB1$ ipadm create-ip vnic2
LB1$ ipadm create-addr -d -a 203.0.113.3/24 vnic2
LB1$ vrrpadm create-router -V 2 -A inet -l eth1 -p 255 vrrp2
```

3. **Configure Load Balancer 2 to serve as the standby load balancer.**

```
LB2$ dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2$ ipadm create-ip vnic1
LB2$ ipadm create-addr -d -a 198.51.100.3/24 vnic1
LB2$ vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2$ dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB2$ ipadm create-ip vnic2
LB2$ ipadm create-addr -d -a 203.0.113.3/27 vnic2
LB2$ vrrpadm create-router -V 2 -A inet -l eth1 -p 100 vrrp2
```

4. **Add the IP address for the floating default gateway to both servers.**

```
$ route add default 203.0.113.3
```

This configuration provides protection against the following failure scenarios:

- If Load Balancer 1 fails, Load Balancer 2 becomes the primary load balancer. Load balancer 2 then takes over address resolution for the VIP `198.51.100.3` and handles all the packets from clients with the destination IP address `198.51.100.3`. Load balancer 2 also handles all the packets that are sent to the floating gateway address `203.0.113.3`.

When Load Balancer 1 recovers, Load Balancer 2 returns to the standby mode.

- If one or both of Load Balancer 1's interfaces fail, Load Balancer 2 takes over as primary load balancer. Load Balancer 2 then takes over address resolution for VIP `198.51.100.3` and handles all packets from clients with the destination IP address `198.51.100.3`. Load balancer 2 also handles all the packets that are sent to the floating gateway address `203.0.113.3`.

  When both of Load Balancer 1's interfaces are healthy, Load Balancer 2 returns to standby mode.

Index

S

T

V