



HP-UX Containers (SRP) A.03.01 Administrator's Guide

HP-UX 11iv3



Table of contents

Preface.....	5
Intended audience.....	5
Typographic conventions	5
Related information	6
Publishing history	6
HP encourages your comments	7
Part I: Getting started	8
1 Introduction.....	9
1.1 HP-UX Containers overview.....	9
1.2 Resource management.....	9
1.3 Network management	9
1.4 Isolation and namespace	10
1.5 Container types	10
1.6 Global view	11
1.7 Compatibility with other virtualization continuum products	12
2 Installing HP-UX Containers.....	13
2.1 Upgrading to HP-UX Containers A.03.01	13
2.2 Resource considerations for containers	13
Part II: Managing containers	14
3 HP-UX Containers components	15
3.1 The Container Manager.....	15
3.2 The <code>srp_sys</code> command.....	15
3.3 The <code>srp</code> command	15
3.4 The <code>srp_init</code> daemon.....	15
3.5 The <code>srp_su</code> command.....	16
3.6 The <code>srp_ps</code> command.....	16
3.7 The <code>srp_check</code> command.....	16
3.8 The <code>srp_sync</code> command.....	16
4 Using Container Manager	17
4.1 Accessing Container Manager from SMH	17
4.2 Container Manager home page	18
4.3 Accessing Container Manager help	19
4.4 Setting the HP-UX Containers environment.....	19
4.5 Container Manager – creating a container	20
4.6 Container Manager – viewing or modifying a container.....	22
4.7 Container Manager – starting a container.....	25

4.8 Container Manager – stopping a container.....	27
5 Configuring HP-UX Containers using the <code>srp_sys</code> command.....	29
5.1 Configuring the system for HP-UX Containers.....	29
5.2 Displaying subsystem properties	34
5.3 Disabling HP-UX Containers subsystems	35
6 Managing containers using the <code>srp</code> command	36
6.1 Templates, services, and variables	36
6.2 Creating a container	37
6.3 Starting and stopping a container	39
6.4 Displaying the status of a container.....	40
6.5 Updating container configuration.....	41
6.6 Non-interactive <code>srp</code> command invocation	43
6.7 Displaying help text for input parameters	43
6.8 Listing container configuration information	44
6.9 Copying containers by exporting and importing	44
6.9.1 Using the <code>srp -export</code> command.....	46
6.9.2 Using the <code>srp -import</code> command.....	47
6.9.3 Best practices for exporting and importing a container.....	48
7 Using the <code>srp_init</code> command.....	50
7.1 Changing the run level of a container.....	50
7.1.1 Log files	51
8 Using the <code>srp_su</code> command	52
8.1 Allowing additional users to use the <code>srp_su</code> command	52
9 Using the <code>srp_ps</code> command	53
10 Establishing a user session in a container	54
11 Container startup and shutdown	55
12 Networking with containers	57
12.1 Configuring the first network interface	57
12.2 Configuring an additional network interface.....	58
12.3 Configuring additional routing entries.....	60
12.4 Network configuration for the global view	61
12.5 Address collisions with <code>INADDR_ANY</code> and <code>IN6ADDR_ANY</code> sockets in the global view.....	62
12.6 Cross-container network traffic and Force-to-Wire	62
12.7 IP Routers and strong end system (ES) model	63
12.7.1 Application gateway servers.....	63
13 Using Serviceguard with containers	64
13.1 Choosing a model.....	64
13.2 Creating a container to use with Serviceguard.....	64
13.3 Adapting Serviceguard scripts for the classic model	65
13.4 Creating Serviceguard scripts for the container package model.....	66
Part III: Container type specific management	68
14 Container types	69
14.1 System containers	69
14.2 Workload containers.....	69
14.3 Comparison of system and workload types	69
14.4 Choosing a container type	72
14.4.1 When to use a system container.....	72
14.4.2 When to use a workload container.....	72
15 System container	73
15.1 Managing file system	73
15.1.1 Choosing a file system subtype	73
15.1.2 Mounting file system	74
15.2 Users, groups and authentication	76

15.2.1	Displaying system container user, group, and process names from the global view	76
15.3	Security features	77
15.3.1	Extended security attributes	77
15.3.2	Audit	77
15.4	Managing devices	79
15.5	Software management	79
15.5.1	Managing software	80
15.5.2	Remote network registered depots	81
15.5.3	Allowed products	82
15.5.4	Updating the global view and individual system containers	82
15.5.5	Recovering unsynchronized containers	83
15.5.6	Advanced installation: varying software versions	84
15.5.7	Installation methods other than SD	84
15.6	Deploying applications	85
15.7	Limitations and disallowed operations	86
15.7.1	Disallowed commands	88
15.8	System templates	88
16	Workload Container	92
16.1	File system	92
16.2	Users, groups and authorization	92
16.3	Security features	93
16.4	Devices	93
16.5	Installing software	93
16.6	Deploying applications	94
16.6.1	Single instance applications	94
16.6.2	Multi-instance applications	94
16.6.3	Deploying applications with the application templates	94
16.6.4	Ensuring access to application files located outside the container home directory	94
16.6.5	Best practices for application deployment with workload containers	94
16.7	Limitation and disallowed operations	95
16.8	Workload templates	95
16.8.1	Workload template	96
16.8.2	SSHD template	98
16.8.3	Apache template	100
16.8.4	Tomcat template	102
16.8.5	Oracle template	105
16.8.6	Custom template	106
17	Compatibility with other HP-UX products	108
17.1	Bastille revert feature	108
17.2	PRM HP-UX Containers commands	108
17.3	Configuration Synchronization Manager (CMGR) utility and libraries	108
18	Verifying and troubleshooting containers	109
18.1	Quick verification procedure	109
18.2	Troubleshooting scenarios	109
18.3	Advanced verification procedures	111
18.3.1	Verifying Security Containment compartment data	111
18.3.2	Verifying RBAC data	112
18.3.3	Verifying PRM data	112
18.3.4	Verifying network data	113
18.3.5	Verifying IPFilter data	113
18.3.6	Verifying IPsec data	114
18.4	Advanced Troubleshooting Procedures	115
18.4.1	Using the Security Containment compartment discover feature (workload containers only)	115
18.4.2	Removing or disabling IPFilter	116
18.4.3	Removing or disabling IPsec	116
18.5	Reporting Problems	116
	Glossary	118

Appendix A: Container default route script for Serviceguard.....	120
Appendix B: Direct customization of container properties	123
B.1 Execute customer defined operations via provision scripts	123
B.2 Customize security containment definition (not supported for system containers).....	123
B.2.1 Securing containers with compartment rule Include files (Not supported for system containers) .	123
B.2.2 Manually editing configuration data	124
Appendix C: Template services – detailed description	127
C.1 The cmpt service.....	127
C.1.1 Configuration location	127
C.2 The admin service.....	127
C.2.1 Configuration location	127
C.3 The prm service	128
C.3.1 Configuration data.....	128
C.4 The network service	128
C.4.1 Configuration location	128
C.5 The init Service.....	129
C.5.1 Configuration location	129
C.6 The login service (workload containers).....	130
C.6.1 Configuration location	130
C.7 The ipfilter service.....	130
C.7.1 Configuration location	130
C.8 The ipsec service	131
C.8.1 Configuration location	131

Preface

This document describes how to install, configure, and troubleshoot HP-UX Containers (formerly Secure Resource Partitions (SRP)).

Intended audience

This document is intended for system and network administrators responsible for installing, configuring, and managing HP-UX Containers – formerly Secure Resource Partitions (SRP). Administrators are expected to have knowledge of operating system and networking concepts, commands, and configuration. Familiarity with the HP-UX Security Containment, HP Process Resource Manager (PRM), HP-UX IPFilter, and HP-UX IPsec products is useful. This document is not a tutorial.

Typographic conventions

This document uses the following typographical conventions:

<code>%</code> , <code>\$</code> , or <code>#</code>	A percent sign represents the C shell system prompt. A dollar sign represents the system prompt for the Bourne, Korn, and POSIX shells. A number sign represents the superuser prompt.
<code>audit(5)</code>	A manpage. The manpage name is <i>audit</i> , and it is located in Section 5.
Command	A command name or qualified command phrase.
Computer output	Text displayed by the computer.
Ctrl+x	A key sequence. A sequence such as Ctrl+x indicates that you must hold down the key labeled Ctrl while you press another key or mouse button.
ENVIRONMENT VARIABLE	The name of an environment variable; for example, <code>PATH</code> .
ERROR NAME	The name of an error, usually returned in the <code>errno</code> variable.
Key	The name of a keyboard key. Return and Enter both refer to the same key.
<i>Term</i>	The defined use of an important word or phrase.
User input	Commands and other text that you type.
<i>Variable</i>	The name of a placeholder in a command, function, or other syntax display that you replace with an actual value.
[]	The contents are optional in syntax. If the contents are a list separated by <code> </code> , you can choose one of the items.
{ }	The contents are required in syntax. If the contents are a list separated by <code> </code> , you must choose one of the items.
...	The preceding element can be repeated an arbitrary number of times.
:	Indicates the continuation of a code example.
	Separates items in a list of choices.
WARNING	A warning calls attention to important information that if not understood or followed results in personal injury or nonrecoverable system problems.
CAUTION	A caution calls attention to important information that if not understood or followed results in data loss, data corruption, or damage to hardware or software.
IMPORTANT	An important provides essential information to explain a concept or to complete a task.
NOTE	A note contains additional information to emphasize or supplement important

points of the main text.

Related information

For more information about the products and subsystems used with HP-UX Containers, see the following documentation:

Document	Location
<i>HP 9000 Containers Administrator's Guide</i>	http://www.hp.com/go/hp9000-containers
HP-UX Security Containment and Role-Based Access Control (RBAC), documented in the <i>HP-UX System Administrator's Guide: Security Management: HP-UX 11i v3</i> .	http://www.hp.com/go/hpux-core-docs Select the HP-UX 11i v3 product.
HP-UX IPFilter	http://www.hp.com/go/hpux-security-docs Select the HP-UX IPFilter Software product.
HP-UX IPSec	http://www.hp.com/go/hpux-security-docs Select the HP-UX IPSec Software product.
<i>HP PRM User's Guide</i>	http://www.hp.com/go/hpux-core-docs Select the HP-UX 11i Process Resource Management (PRM) Software product.

HP-UX Containers include the following manpages:

Manpage	Description
<code>cmgr(1M)</code>	Describes the Configuration Synchronization Manager.
<code>container(5)</code>	Defines HP-UX Containers.
<code>container_system(5)</code>	Describes an HP-UX system container.
<code>container_workload(5)</code>	Describes an HP-UX workload container.
<code>srp(1M)</code>	Manages the configuration of HP-UX Containers.
<code>srp_check(1M)</code>	Identifies and lists the products that are out of sync with the global.
<code>srp_init(1M)</code>	Container specific process control initialization.
<code>srp_ps(1M)</code>	Reports process status for a container in an HP-UX Containers environment.
<code>srp_su(1M)</code>	Executes the <code>su</code> command in the specified HP-UX Containers.
<code>spr_sync(1M)</code>	Displays products and patches in a system container that are out of sync with the global system.
<code>srp_sys(1M)</code>	Manages system level configuration settings for HP-UX Containers.

Publishing history

The document printing date and part number indicate the document's current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The document part number will change when extensive changes are

made. Document updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

This document is located at:

www.hp.com/go/virtualization-manuals

Select the **HP-UX Containers (SRP) Software** product.

Manufacturing Part Number	Supported Operating Systems	Supported Versions	Publication Date
5900-1837	HP-UX 11i v3	Version 3.1	July 2011
5900-1316	HP-UX 11i v3	Version 3.0	April 2011
5900-0911	HP-UX 11i v3	Version 2.2	August 2010
5992-5172	HP-UX 11i v3	Version 2.01	December 2009
5992-4679	HP-UX 11i v3	Version 2.0	October 2008

HP encourages your comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to:

<http://www.hp.com/bizsupport/feedback/ww/webfeedback.html>

Include the document title, manufacturing part number, and any comment, error found, or suggestion for improvement you have concerning this document.

Part I: Getting started

1 Introduction

HP-UX Containers allows you to deploy multiple isolated container-based environments on a single server platform. This allows the enterprise to host multiple workloads on a single operating system instance, thereby better utilizing server resources (CPU, memory, network access) and data center resources (power, cooling, footprint), and reduce the overall number of operating system instances to manage. HP-UX Containers share a single operating system kernel, global server administrative domain, and are configurable to either isolate or share system hardware and software resources.

1.1 HP-UX Containers overview

HP-UX Containers are nested within the HP-UX operating system. Processes within a container have limitations on access to CPU, memory, networking, storage, file system, and other processes on the system. Containers can be created, deleted, and cloned or migrated to another system. Containers can be individually started (booted), stopped (shutdown), and restarted any time after system startup has completed.

1.2 Resource management

HP-UX Containers supports the ability to allocate CPU and memory usage per container. By default, each container on the system is assigned a Process Resource Manager (PRM) group. Each PRM group can be assigned CPU and memory allocations. PRM provides two allocation models:

- **Share based:** Restrictions (excluding maximum utilization caps) are not applied until the managed resource is fully utilized, at which the operating system scheduler or memory manager applies an algorithm to allocate resources proportional to each PRM group's share size. This model ensures that individual containers can utilize available resources without frequent tuning of allocations.
- **Dedicated:** The specified PRM group is allocated a fixed quantity of the resource for its own exclusive use. This model guarantees immediate and complete access to the resource at the expense of the ability to allow other PRM groups access to the currently unused resource. Dedicated CPU allocation can be used to limit the software license requirements for some software products.

You can apply a combination of resource allocation models on a single server. You can also choose to disable the use of PRM, either to allow the use of a different resource allocation utility such as Workload Manager (WLM) or Global Workload Manager (gWLM), or to disable per container resource management for your server.

1.3 Network management

Each container is allocated one or more logical network IP address interfaces. By default, a container will only be allowed access to its assigned interface. Multiple containers can utilize a single physical network interface. HP-UX Containers supports the usage of IPv4 and IPv6 addresses, IPSec secured transport, and IPFilter host firewall protection. HP-UX Containers supports the usage of *Force-to-Wire* network interface option to ensure that network traffic between specified containers on the same server will always traverse a physical network. For more information on network administration with HP-UX Containers, refer to *12 Networking with containers*.

1.4 Isolation and namespace

Container isolation is provided by applying a combination of access restrictions and private namespace features.

Access restrictions

HP-UX Containers uses HP-UX Security Containment Compartments features to limit access to file system paths, network interfaces, kernel functionality, inter-process communication (IPC) and process view/signaling for processes executing within containers. Each container is configured with its own compartment definition. Access restrictions enforced by compartment definitions take precedence over standard HP-UX access control mechanisms on all users (including `root`). The global view inherits the security properties of the INIT security containment compartment which has no access restrictions. Some container types allow for the customization of compartment access restrictions. See *Part III: Container type specific* for more information.

Private namespaces

A namespace is a scope for unique usage of resource names or other identifiers. When HP-UX Containers is disabled, HP-UX subsystems provide only a single namespace per server. When HP-UX Containers is enabled, many HP-UX subsystems support the option to assign a separate namespace per container. For example, a private hostname namespace allows a container to apply its own value for hostname. For each subsystem, a container will either be utilizing a private namespace only visible from within the container, or a shared namespace visible to the global view and all other containers in the shared namespace. Note that visibility to the shared namespace does not provide access, which is separately granted through the compartment definition plus standard HP-UX access control mechanisms.

HP-UX Containers provides the ability to assign private namespaces per container for the following subsystems:

- File system (via `chroot`)
- Host, node, and domain name
- Loopback IP address port space
- IPC (such as semaphores, message queues, and shared memory)

With the exception of file system namespace in which the per-container private namespaces are nested within the shared namespace, private namespaces are isolated from other containers. Private namespace support varies by container type. See *Part III: Container type specific* for details on namespace support by container type.

1.5 Container types

HP-UX Containers support multiple container types. When creating a new container on the system, you must select the container type, which is a permanent attribute that cannot be changed for the life of the container.

All container types support CPU and memory resource allocations per container, dedicated IP (logical) network interfaces, process and file access isolation, per container initialization and shutdown, cloning and migration to and from other HP-UX servers.

The following container types are managed by the HP-UX Container utilities:

- **System:** Provides many of the user space capabilities of a virtual machine guest without the associated management and performance overhead. Each container has a private `chroot`.

based file system view, hostname, IPC namespace, and service daemons. Common system administration activities such as user management are performed within each system container.

- **Workload:** Provides lightweight workload hosting environment. All workload containers share the global file system view, hostname, IPC namespace, and service daemons. System administration activities are shared with the global view.
- **HP 9000:** Provides a binary emulation environment for HP-UX PA-RISC workloads. The HP 9000 Containers product is separately installed from the HP-UX Containers (SRP) product. Once the HP-UX Containers (SRP) and HP 9000 Containers product are both installed, you can use the `srp(1M)` command to create and manage HP 9000 Containers. Refer to the HP 9000 Containers documentation (<http://www.hp.com/go/hp9000-containers>) for product version requirements, and details on how to configure and manage an HP 9000 container.

For more information on choosing and managing container types, see *14 Container types*.

1.6 Global view

When you enable HP-UX Containers on a system, all processes not executing within a container execute in the **global view**. Sessions logging in via the system console, or connecting via telnet or SSH to server IP addresses not assigned to any container, will execute in the global view.

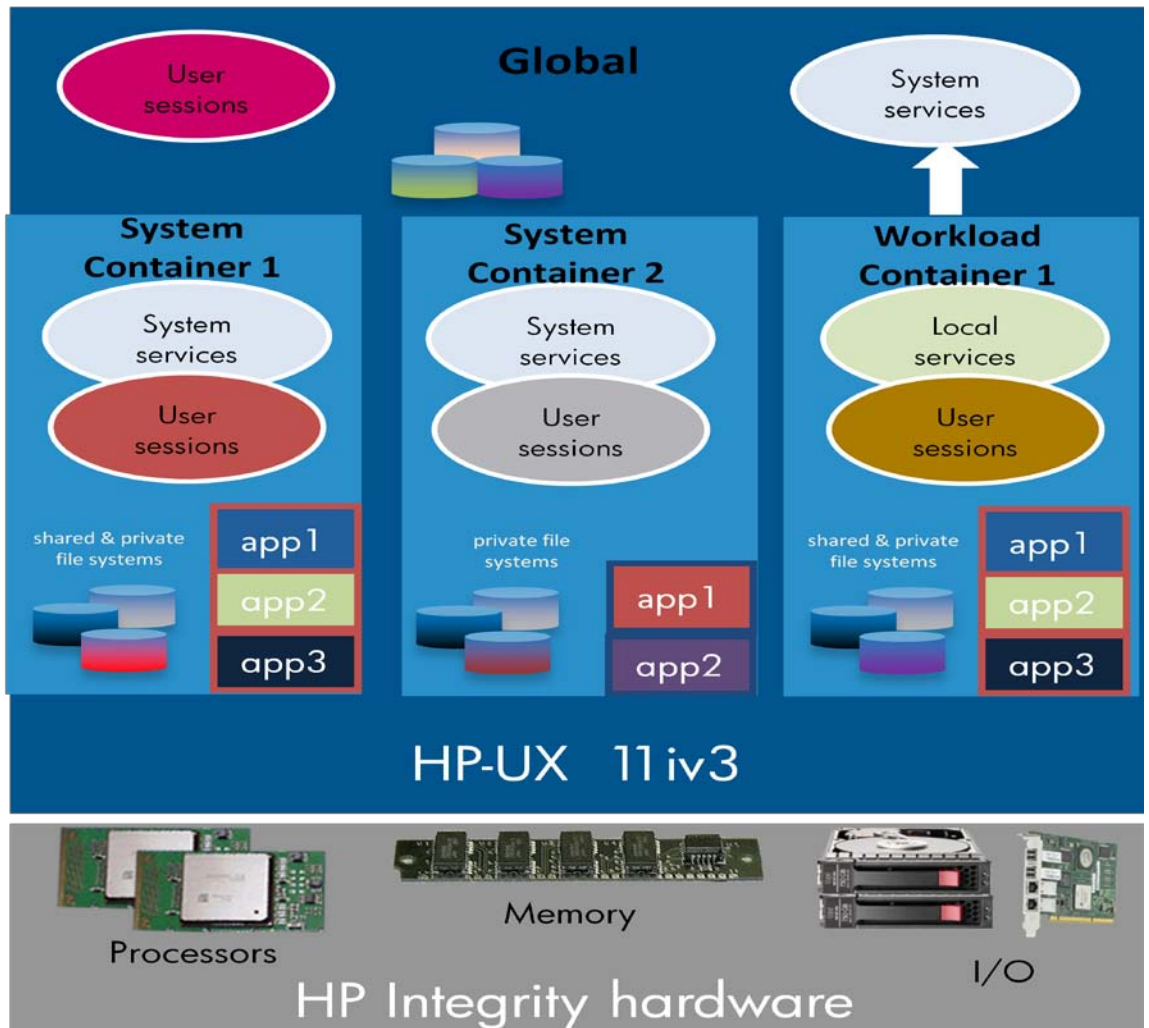
The global view has no access restrictions and therefore can view and manage processes in the global view and all containers. System administration activity that you must perform in the global view includes installing Software Distributor (SD) packaged software, device management, network interface management, setting kernel tunables, and executing system management utilities such as `smh(1M)`, and `srp(1M)`. You can perform file backup and recovery in either the global view or from within the individual containers.

As the global view has unrestricted access to system resources, HP recommends that you use the global view for system management activities. Hosting of general purpose application workload should occur within a container. HP also recommends that users provided with the root account to a system container not be provided any account access to the global view.

You can determine if your session is executing within a container or the global view by using the `getprocxsec -c` command. The `getprocxsec` command will return `init` as the compartment name if your session is executing within the global view, or the container name if your session is executing within a container.

Figure 1.1 shows an HP-UX Integrity server with HP-UX Containers installed, including the global view, and three containers: two system containers and one workload container. Each container has a dedicated IP interface, isolated container home directory (`/var/hpsrp/container_name`), container dedicated processor set (`pset`), and separate instances of service daemons running. Each system container utilizes a private set of service daemons, while all workload containers share most service daemons with the server.

Figure 1.1 HP-UX Containers on HP-UX Integrity



1.7 Compatibility with other virtualization continuum products

HP-UX Containers is a component of the Virtualization Continuum for HP-UX and is compatible with HP-UX nPartitions, HP-UX vPars, and Integrity Virtual Machine (VM) solutions. You can create a container in any HP-UX OS image; the OS image can exist in an nPartition, vPars, Integrity VM, or directly on non-partitioned server hardware.

2 Installing HP-UX Containers

You can acquire and install HP-UX Containers free of charge from HP Software Depot:

<http://www.software.hp.com>

For system and environment requirements, see the *HP-UX Containers (SRP) A.03.01 Release Notes* located at:

www.hp.com/go/virtualization-manuals

Select the **HP-UX Containers (SRP) Software** product.

2.1 Upgrading to HP-UX Containers A.03.01

If your system has an operating system older than HP-UX 11i, March 2011 Release, then you must first update your system to March 2011 (or later) before installing HP-UX Containers A.03.01.

HP-UX Containers A.03.01 supports upgrade from HP-UX SRP A.02.02 only. If you have workload SRPs configured using HP-UX SRP A.02.02 and you need to continue using them, you can migrate the workload SRPs to workload containers after the upgrade.

For more information about updating your system from HP-UX SRP A.02.02 with existing containers configured, please refer to the *HP-UX Containers (SRP) A.03.01 Release Notes*.

2.2 Resource considerations for containers

The system container with shared file system requires a minimum of 4.5 GB disk space and the system container with private file system subtype requires minimum of 9.0 GB disk space. The workload container requires 700 KB disk space (see *1.5 Container types* and *14 Container types* for a description of container types). Other than additional disk space, containers do not have any hardware or additional capacity requirements.

Memory, CPU, `nprocs`, `nfiles`, and `nthreads` have the same settings as those used for hosting the same set of applications without containers. The kernel tunable adjustment may be required when the system hosts a larger set of workloads.

There is a negligible additional overhead in terms of processes and virtual memory consumed for a started system container that has no additional applications deployed. Containers that are not started or active do not require resources with the exception of disk space.

NOTE: Processor cores assigned to a container using PRM as PSET are completely dedicated and will not be available for other use, even if the container is in the `stopped` state.

Part II: Managing containers

3 HP-UX Containers components

HP-UX Containers includes the following components to help manage the containers:

- The Container Manager
- The `srp_sys` command
- The `srp` command
- The `srp_init` daemon
- The `srp_su` command
- The `srp_ps` command
- The `srp_check` command
- The `srp_sync` command

3.1 The Container Manager

The Container Manager is integrated with System Management Homepage (SMH) and provides a graphical user interface (GUI) to configure and manage containers. See [4 Using Container Manager](#) for more information.

3.2 The `srp_sys` command

The `/opt/hpsrp/bin/srp_sys` command manages system-wide configuration properties for HP-UX Containers. You must run the `srp_sys -setup` command (or `srp_sys -enable` for the non-interactive mode) to configure the system for HP-UX Containers prior to configuring individual containers on the system. You can also use `srp_sys` to view and modify system-wide configuration settings for HP-UX Containers. See [5 Configuring HP-UX Containers using the `srp_sys` command](#) for more information.

3.3 The `srp` command

The `/opt/hpsrp/bin/srp` command is an interactive program that prompts you for information and creates a container. The `srp` command supports options to perform the following tasks:

- Create a container
- List the status of a container
- List the container configuration contents
- Replace configuration information for an existing container
- Delete all or part of the configuration for an existing container
- Start up or shut down a container
- Export (copy) a container by creating a container exchange package
- Import (create) a container on your system by using a container exchange package
- Display help information, including information about input parameter

See [6 Managing containers using the `srp` command](#) for more information.

3.4 The `srp_init` daemon

The `/sbin/srp_init` daemon provides the container with a bootstrap service to launch other processes and services inside the container when it is started. Similar to the `init(1M)` command, the `srp_init` command runs as a daemon and processes run level requests to invoke commands configured in the container's `inittab` file. See [7 Using the `srp_init` command](#) for more information.

3.5 The `srp_su` command

The `/opt/hpsrp/bin/srp_su` command allows a user in the global view to launch a shell (via `su(1)` command) in the specified target container. This can be used by system administrators for the purpose of login or command execution within a container. See *8 Using the `srp_su` command* for more information.

3.6 The `srp_ps` command

The `/opt/hpsrp/bin/srp_ps` command reports process status for a specific container on your system. See *9 Using the `srp_ps` command* for more information.

3.7 The `srp_check` command

The `srp_check` command identifies and lists the products that are out of sync with the global. Refer to `srp_check(1M)` for more information.

3.8 The `srp_sync` command

The `srp_sync` command displays products and patches in a system container that are out of sync with the global system. Refer to `srp_sync(1M)` for more information.

4 Using Container Manager

The Container Manager is integrated in the System Management Homepage (SMH) and provides a graphical user interface (GUI) to configure and manage system and workload containers. With Container Manager, you can perform the following tasks:

- Enable or disable system wide configuration to configure containers
- Monitor a container status and activity on your system
- Create a new container
- Start or stop a container
- Export and import a container
- Modify a container
- Delete a container
- Configure multiple IP addresses for a container

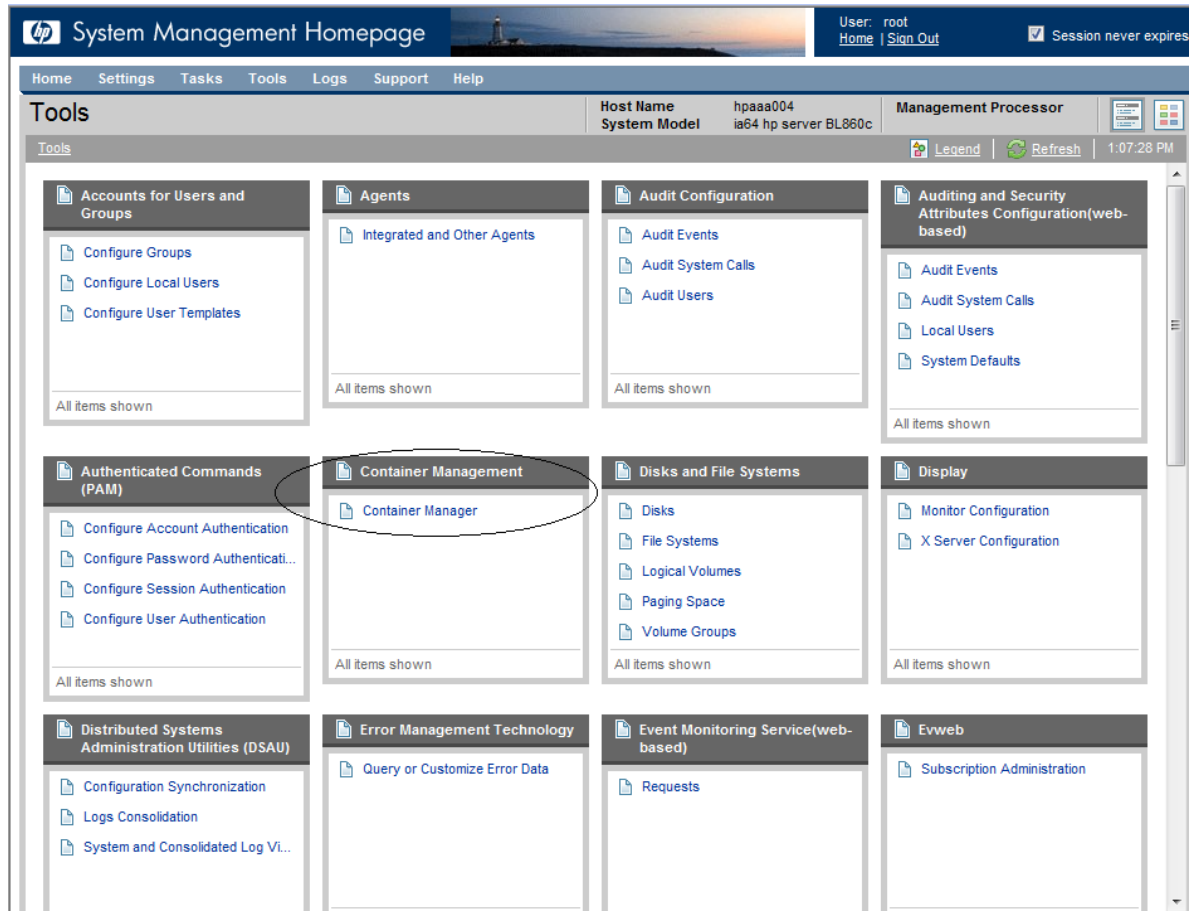
This section provides an overview to introduce the basic operations of the Container Manager. More information is included with the Help subsystem included with the Container Manager.

4.1 Accessing Container Manager from SMH

To access Container Manager from SMH, follow these steps:

1. Login to SMH at using the SMH administrator or root credential:
<http://hostname:2301/>
2. On the SMH Homepage, click **Tools**.
NOTE: SMH GUI sessions stop after the session timeout period elapses without any user activity. By default, the session timeout period is 15 minutes. To prevent the session from timing out, select the **Session never expires** check box in the upper right corner of the page.
3. On the SMH Tools page, select **Container Manager** from the **Container Management** menu box, as shown in Figure 4.1.

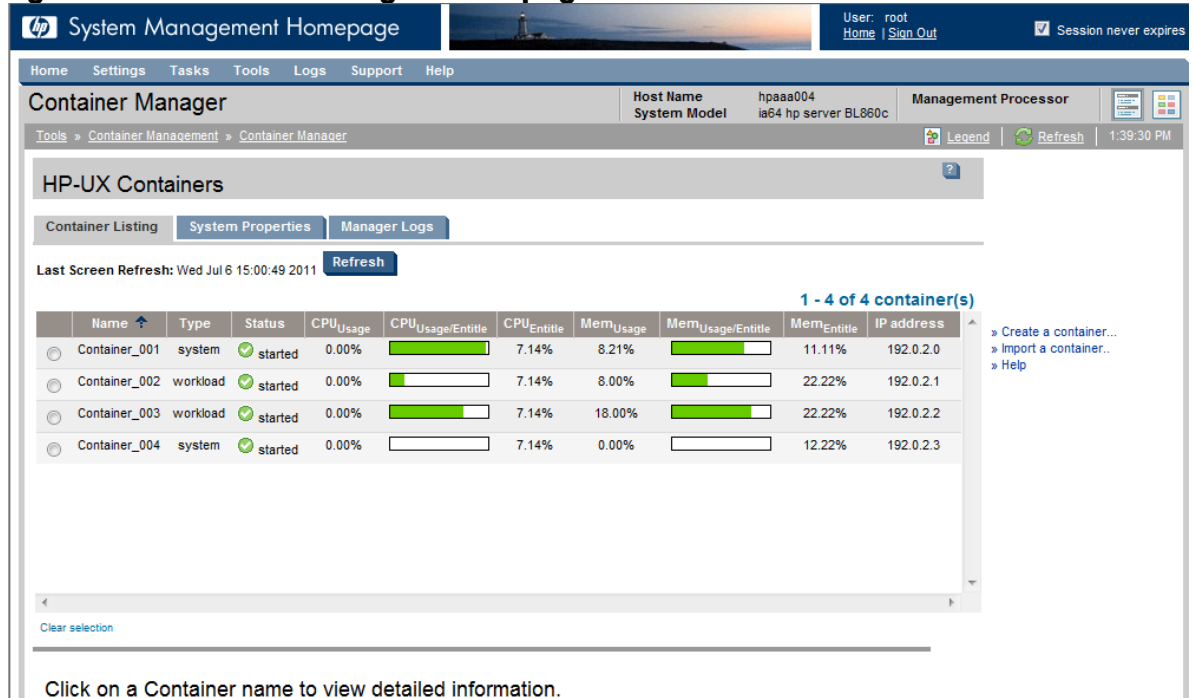
Figure 4.1 SMH: Selecting Container Manager



4.2 Container Manager home page

The Container Manager home page provides a view of all containers on the system including current state and resource utilization for each container. Figure 4.2 shows the Container Manager home page.

Figure 4.2 Container Manager home page



NOTE: From the Container Manager help files, select the **Displaying Container List and Status** help menu item for more information on the Container Manager home page.

4.3 Accessing Container Manager help

To access help information for any Container Manager page, click the question mark icon located in the upper right corner of the Container Manager homepage.

4.4 Setting the HP-UX Containers environment

Before you can create any container, you must setup your HP-UX Containers environment. The system properties verify that the subsystem level configuration meets the requirements to configure a container on the host. To use the HP-UX Containers technology, you must enable the core subsystem properties before creating containers on the system by either using the `srp_sys` command (with the `-enable` or `-setup` option) or by using the Container Manager.

To enable HP-UX Containers technology using the Container Manager, follow these steps:

1. Go to the **System Properties** tab.
2. Click the **Enable** button for the *SRP core subsystems* property.
3. Wait for the successful completion message to appear in the result window.
4. Enable the `compartment login` feature, the `PRM` service, and the `sshd` configuration properties. Optionally, you can enable other properties.
5. Reboot the system to enable the required properties.

NOTE: The **Container Listing** tab will only be displayed if the `SRP core subsystem` property status is set to **OK**.

Figure 4.3 shows the **System Properties** tab after the user enabled the selected system properties and rebooted the system.

Figure 4.3 Enabling the system properties

The screenshot shows the HP System Management Homepage. The main content area is titled 'Container Manager' and has a sub-tab 'System Properties' selected. Below this, there is a table of system properties. The table has four columns: Properties, Status, Detail, and Action. The 'Action' column contains 'Enable' and 'Disable' buttons for each property.

Properties	Status	Detail	Action
SRP core subsystems	OK	SRP Default Service list: cmpt,admin,int,login,prm,network,provision Security Containment Compartment: OK Multiple namespace support: OK Network Configuration: OK SRP system services: OK SRP User and Group: OK	Enable Disable
compartment login feature	OK	SRP Default Service List: login service enabled	Enable Disable
PRM service	OK	PRM Memory record status: Enabled SRP Default Service List: prm service enabled	Enable Disable
IPFilter module	Not Enabled		Enable Disable
IPsec module	Not Installed		Enable Disable
sshd configuration	OK	Global view sshd currently listens on these addresses: 192.0.2.0	Enable Disable

NOTE: From the Container Manager help files, select the **Properties** help menu item for more information on the System Properties page.

4.5 Container Manager – creating a container

You can use the Container Manager to create a new container on your system. When you create a container, you select the type of container: workload or system. For both types of containers, you can specify optional services (such as PRM, IPFilter, and IPsec). For workload containers, you can specify additional application templates (SSHD, Apache, Tomcat, Custom, and Oracle).

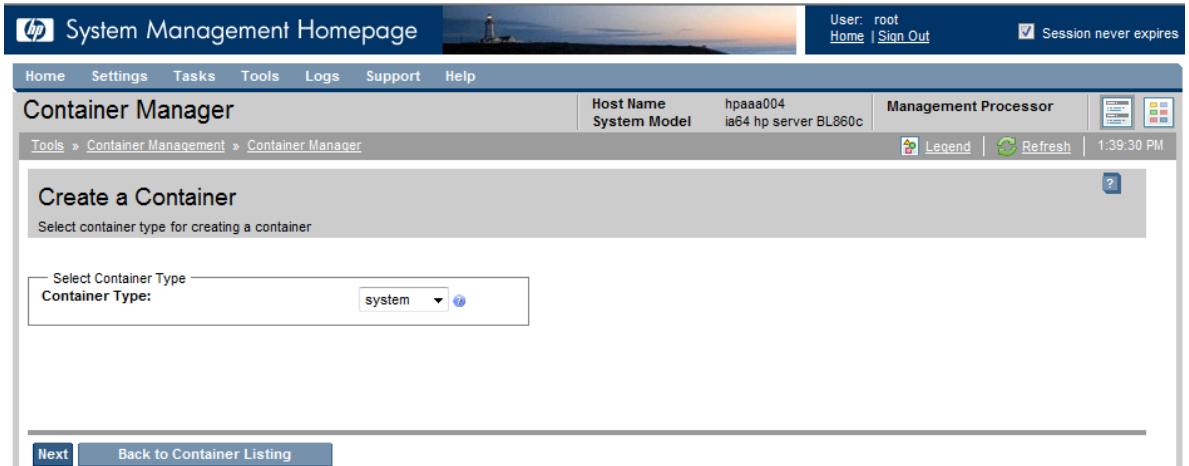
To create a container, follow these steps:

1. From the Container Manager home page, click **Create a container**.
2. Select the **Container Type** that you want to create, either **system** or **workload**. (see Figure 4.4)
3. Click **Next** to continue.
4. Enter a **Container Name** (see Figure 4.5).
NOTE: You cannot use the keywords `system` or `workload` as container names.
5. Modify any fields as desired.
6. Click **Create**. A result window pops up and shows the create command being executed and the logs being generated on the host. Once the Create operation is complete, an operation success or failure message is displayed.
7. Click **Back to container listing** and close the result window.

NOTE: If you created a workload container and you want to add an application template, go to **View and Modify Container** that is below the **Containers Listing** tab and select the desired template tab. Enter your selection and click **Add Configuration**.

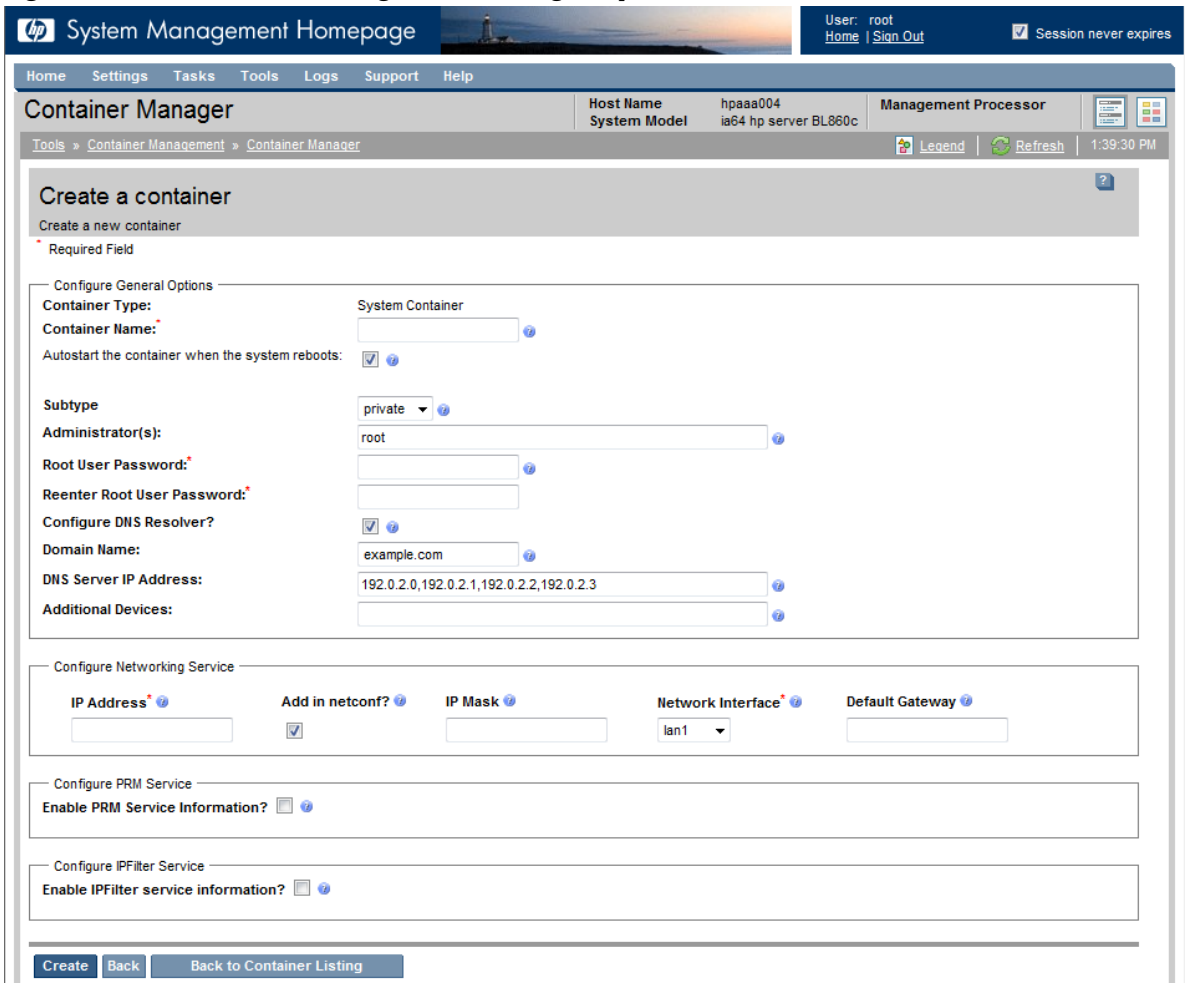
In Figure 4.4, the user selects to create a system container.

Figure 4.4 Container Manager – creating a system container: selecting a type



In Figure 4.5, the user enters the required fields and modifies any optional fields as desired.

Figure 4.5 Container Manager – creating a system container



NOTE: From the Container Manager help files, select the **Create a container** help menu item for more information on creating a container.

4.6 Container Manager – viewing or modifying a container

You can view or modify the configuration of a container once it has been created. Modifying some parameters (such as networking parameters) requires that the container be in the `stopped` state. To add, delete, or modify an application template configuration, you must restart the container.

Follow these steps to view or modify a container:

1. From the Container Manager home page, click the **Container Listing** tab and select the container that you want to modify. The detailed view for the selected container is displayed below the **Container Listing** tab.

The **Overview** tab displays the key properties for the selected container. The **Process View** tab lists all the process running in the container. The **System** tab (for system containers) or **Workload** tab (for workload containers) provides detailed configuration and lets you enable various services for the selected container.

2. Select the configuration tab you wish to view or modify.
3. To apply your changes, click **Modify Container**. A result window displays the output of the modification. After the operation completes, click **Close This Window**.

IMPORTANT: You must click on **Modify Container** before clicking on a new tab, or your changes will not be applied, and the configuration fields will be reset to the original values.

In Figure 4.6, the user selects to view the **System** tab for `Container_001`.

Figure 4.6 Container Manager – viewing or modifying a container

The screenshot displays the HP System Management Homepage interface for the Container Manager. At the top, the user is logged in as 'root' with a session that never expires. The main navigation bar includes Home, Settings, Tasks, Tools, Logs, Support, and Help. The page title is 'Container Manager', and it shows the host name 'hpaaa004' and system model 'ia64 hp server BL860c'. The breadcrumb trail is 'Tools > Container Management > Container Manager'. The main content area is titled 'HP-UX Containers' and has tabs for 'Container Listing', 'System Properties', and 'Manager Logs'. A 'Refresh' button is present, and the last screen refresh was on Wed Jul 6 12:18:53 2011. A table lists 4 containers:

Name	Type	Status	CPU Usage	CPU Usage/Entitle	CPU Entitle	Mem Usage	Mem Usage/Entitle	Mem Entitle	IP address
Container_001	system	started	6.98%	7.14%	7.14%	8.21%	11.11%	11.11%	192.0.2.0
Container_002	workload	started	1.00%	7.14%	7.14%	8.00%	22.22%	22.22%	192.0.2.1
Container_003	workload	started	5.40%	7.14%	7.14%	18.00%	22.22%	22.22%	192.0.2.2
Container_004	system	stopped	0.00%	7.14%	7.14%	0.00%	12.22%	12.22%	192.0.2.3

Below the table, the 'View and Modify Container "Container_001" Properties' section is shown. It has tabs for 'Overview', 'Process View', 'System', and 'Custom'. The 'System' tab is selected, and a green checkmark indicates it is configured. Under 'Configure General Options', the following settings are visible:

- Autostart the container when the system reboots:
- Subtype: private
- Administrator(s): root
- Configure DNS Resolver?:
- Domain Name: example.com
- DNS Server IP Address: 192.0.2.0, 192.0.2.1, 192.0.2.2, 192.0.2.3

NOTE: From the Container Manager help files, select the **Viewing or Modifying a Container** help menu item for more information on viewing or modifying a container.

You can also configure multiple IP addresses for a container. In Figure 4.7, the user clicks on **+ add new instance** in the **Configure Networking Service** section, to add an IP address to Container_001.

Figure 4.7 Container Manager – adding an IP address to a container

The screenshot displays the HP System Management Homepage interface for the Container Manager. The top navigation bar includes 'Home', 'Settings', 'Tasks', 'Tools', 'Logs', 'Support', and 'Help'. The user is logged in as 'root' with a session that never expires. The main header shows 'Container Manager' with host information: 'Host Name: hpaaa004', 'System Model: ia64 hp server BL860c', and 'Management Processor'. A table lists four containers with their respective types, statuses, and resource usage.

Name	Type	Status	CPUUsage	CPUUsage/Entitle	CPUEntitle	MemUsage	MemUsage/Entitle	MemEntitle	IP address
Container_001	system	started	6.98%	<div style="width: 6.98%;"></div>	7.14%	8.21%	<div style="width: 8.21%;"></div>	11.11%	192.0.2.0
Container_002	workload	started	1.00%	<div style="width: 1.00%;"></div>	7.14%	8.00%	<div style="width: 8.00%;"></div>	22.22%	192.0.2.1
Container_003	workload	started	5.40%	<div style="width: 5.40%;"></div>	7.14%	18.00%	<div style="width: 18.00%;"></div>	22.22%	192.0.2.2
Container_004	system	stopped	0.00%	<div style="width: 0.00%;"></div>	7.14%	0.00%	<div style="width: 0.00%;"></div>	12.22%	192.0.2.3

Below the table, the 'View and Modify Container "Container_001" Properties' section is shown. It includes tabs for 'Overview', 'Process View', 'System', and 'Custom'. The 'System' tab is active, showing configuration options for general options and networking services.

Configure General Options

- Autostart the container when the system reboots:
- Subtype: private
- Administrator(s): root
- Configure DNS Resolver?:
- Domain Name: example.com
- DNS Server IP Address: 192.0.2.0, 192.0.2.1, 192.0.2.2, 192.0.2.3

Configure Networking Service

Instance ID	IP Address	Add in netconf?	IP Mask	Network Interface	Default Gateway	Delete
1	192.0.2.0	<input checked="" type="checkbox"/>	255.0.0.0	lan1	192.0.2.0	
		<input checked="" type="checkbox"/>		lan1		X

A '+ add new instance' button is located at the bottom right of the networking service configuration area.

4.7 Container Manager – starting a container

You can start a container if it is currently in the `stopped` state. The current state of each container is displayed on the Container Manager home page.

From the Container Manager home page, follow these steps to start a container:

1. Click on the container that you want to start.
2. Click **Start** located in the right hand task bar.

In Figure 4.8, the user selected `Container_004`.

Figure 4.8 Container Manager – starting a container

The screenshot shows the HP System Management Homepage interface. The main content area is titled "Container Manager" and displays a table of HP-UX Containers. The table has columns for Name, Type, Status, CPU Usage, CPU Usage/Entitle, CPU Entitle, Mem Usage, Mem Usage/Entitle, Mem Entitle, and IP address. Container_004 is selected and highlighted in blue, and its status is "stopped". The actions menu for Container_004 is open, showing options: Start..., Stop..., Delete..., and Export....

Name	Type	Status	CPU Usage	CPU Usage/Entitle	CPU Entitle	Mem Usage	Mem Usage/Entitle	Mem Entitle	IP address
Container_001	system	started	6.98%	7.14%	7.14%	8.21%	11.11%	11.11%	192.0.2.0
Container_002	workload	started	1.00%	7.14%	7.14%	8.00%	22.22%	22.22%	192.0.2.1
Container_003	workload	started	5.40%	7.14%	7.14%	18.00%	22.22%	22.22%	192.0.2.2
Container_004	system	stopped	0.00%	7.14%	7.14%	0.00%	12.22%	12.22%	192.0.2.3

In Figure 4.9, once the user selected Container_004 and clicks on **Start...**, the following status window is displayed.

Figure 4.9 Container Manager – starting a container

Start the container


Run the startup script for the container

Action: Start container Container_004
Command: srp -b -start Container_004
Result: Started executing the SRP command...

```
HP-UX SRP Container start-up in progress

Setting up Containers ..... OK
Remounting Root File System ..... N/A
Setting hostname ..... OK
Start containment subsystem configuration ..... OK
Start Utmp Daemon : manages User Accounting Database .... OK
Recover editor crash files ..... OK
List and/or clear temporary files ..... OK
Clean up old log files ..... OK
Start system message logging daemon ..... OK
Checking user database ..... OK
Configuring DHCPv6 Interfaces ..... OK
Starting HP-UX Secure Shell ..... OK
Start NFS core subsystem ..... OK
Start NFS IPv6 subsystem ..... OK
Start enhanced NFS IPv6 subsystem ..... OK
Start NIS server subsystem ..... OK
Start ldap client daemon ..... N/A
Start NIS client subsystem ..... OK
Start lock manager subsystem ..... OK
Start NFS client subsystem ..... OK
Start AUTOFS subsystem ..... OK
Finish containment subsystem configuration ..... OK
Start Internet services daemon ..... OK
Start remote system status daemon ..... N/A
Starting sendmail [Done] Starting sm-client [Done] ..... OK
Starting the password/group assist subsystem ..... OK
Start print spooler ..... N/A
Start clock daemon ..... OK
PA performance software is being started. .... OK
Initialize Software Distributor agent daemon ..... OK
Starting the Winbind Daemon ..... N/A
Starting HP-UX Apache-based Web Server ..... N/A
Starting HP-UX Tomcat-based Servlet Engine ..... N/A
Starting HP-UX Webmin-based Admin ..... N/A
Starting the HP-UX Webproxy subsystem ..... N/A
Starting HP-UX XML Web Server Tools ..... OK
The HP-UX SRP Container is ready.

start SRP: Container_004 successful.
```

 **Operation Successful.**

[Close This Window](#)

NOTE: From the Container Manager help files, select the **Starting a Container** help menu item for more information on starting a container.

4.8 Container Manager – stopping a container

You can stop a container if it is currently in the `started` state. The current state of each container is displayed on the Container Manager home page.

From the Container Manager home page, follow these steps to stop a container:

1. Click on the container that you want to stop.
2. Click **Stop** located in the right hand task bar.

In Figure 4.10, once the user selected `Container_004` and clicks on **Stop...**, the following status window is displayed.

Figure 4.10 Container Manager – stopping a container

Stop the container



Run the shutdown script for the container

Action: Stop container Container_004
Command: srp -b -stop Container_004
Result: Started executing the SRP command...

HP-UX SRP Container stop in progress

```
Stopping HP-UX Apache-based Web Server ..... OK
Stopping HP-UX Tomcat-based Servlet Engine ..... N/A
Stopping HP-UX Webmin-based Admin ..... OK
Stopping the HP-UX Webproxy subsystem ..... OK
Stopping HP-UX XML Web Server Tools ..... OK
Shutting down the Winbind Daemon ..... OK
Shutting down Perf Agent software ..... OK
Stop clock daemon ..... OK
Stop print spooler ..... OK
Stopping HP-UX Secure Shell ..... OK
Shutting down sendmail [Done] Shutting down sm-client [Done] .... OK
Stopping remote system status daemon ..... N/A
Stopping Internet services daemon ..... OK
Stop AUTOFS subsystem ..... OK
Stop NFS client subsystem ..... OK
Stop lock manager subsystem ..... OK
Stop NIS client subsystem ..... OK
Stop ldap client daemon ..... OK
Stop NIS server subsystem ..... OK
Stop NFS core subsystem ..... OK
Unconfiguring DHCPv6 Interfaces ..... OK
Stop system message logging daemon ..... OK
Stop Software Distributor agent daemon ..... OK
Stop Utmp Daemon ..... OK
Killing user processes ..... OK
Starting the fsdaemon for transition into single-user mode ..... OK
Srp container cleanup ..... OK
HP-UX SRP Container transition to run-level 0 is complete.
```

stop SRP: **Container_004** successful.



Operation Successful.

Close This Window

NOTE: From the Container Manager help files, select the **Stopping a Container** help menu item for more information on stopping a container.

5 Configuring HP-UX Containers using the `srp_sys` command

The `/opt/hpsrp/bin/srp_sys` command is used to enable, disable, view, and update system-wide configuration properties that affect the HP-UX Containers product.

The `srp_sys` command has the following syntax:

```
srp_sys [-s[etup]]
srp_sys [-e[nable]]|[-d[isable]] [subsystems[,...]] [variable=value [...]]
srp_sys [-l[ist]] [subsystems[,...]] [-v[erbose]]
srp_sys [-h[elp]]
```

Where:

<i>setup</i>	Interactive option - Performs system level configuration and verification for HP-UX Containers.
<i>enable</i>	Non-interactive option - Performs the necessary configuration and enablement actions for the specified subsystem. If no subsystem is specified, the default subsystems required for HP-UX Containers will be enabled.
<i>disable</i>	Disables various system wide HP-UX Container configuration properties. If no subsystem is specified, all the subsystems will be disabled.
<i>list</i>	Lists the configuration status of system wide configuration properties used by HP-UX Containers.
<i>help</i>	Displays usage information for the <code>srp_sys</code> command.
<i>verbose</i>	Displays detailed information for the <code>-list</code> operation.
<i>subsystem</i>	A specific group of configurations to be enabled. Valid values: <code>coreset</code> , <code>migrate</code> , <code>cmptlogin</code> , <code>prm</code> , <code>ipfilter</code> , <code>ipsec</code> , and <code>sshd</code> .
<i>variable</i>	Valid variables: <code>sshdlistenip</code> and <code>prmsubsys</code> .

See `srp_sys(1M)` for more information.

5.1 Configuring the system for HP-UX Containers

After the HP-UX Containers product installation completes, you must prepare the system for HP-UX Containers using the `srp_sys` command. The HP-UX Containers features require a set of subsystems to be configured on the system. You can enable the subsystems required for HP-UX Containers in one of two ways:

- Using `srp_sys` with the `-enable` option:

The `srp_sys -enable` option allows you to run the `srp_sys` command in the non-interactive mode. If no subsystem is specified, a default set of required HP-UX subsystems is enabled (`coreset`, `migrate`, `cmptlogin`, `prm`, and `sshd`).

- Using `srp_sys` with the `-setup` option:

The `srp_sys -setup` command ensures that the system is in an appropriate state to successfully configure containers. If a subsystem is not enabled, `srp_sys` prompts you to specify if you want to enable the service. It also prompts for subsystem startup data, such as configuration directories and `autostart` parameters. The `srp_sys -setup` command also prompts you for the HP-UX Containers services that you want to enable.

HP requires that you run `srp_sys -enable` or `srp_sys -setup` after you install HP-UX Containers. You can run `srp_sys` with either option at anytime that you want to change the default parameters for the HP-UX Containers product.

The services you enabled using `srp_sys` become the default services for the templates (only valid services will be applied for any given template). The `srp_sys` command also modifies the HP-UX Containers default template with these subsystem startup data.

The `srp_sys` command manages the following subsystems:

- **Core container subsystem (`coreset`)**

This subsystem is mandatory and sets all the core system properties required for the HP-UX Containers product. The components that will be modified include:

- **Security Containment compartments**

The Security Containment compartments feature provides the base functionality that HP-UX Containers utilizes to provide isolation and namespaces for containers.

- **Container required kernel tunables**

These tunables are required to enable HP-UX Containers functionality:

- **`ip_strong_es_model`**

Required for the HP-UX Containers product when using networking. Enables symmetric routing on the system which causes connection based protocols such as TCP to use the same interface for both inbound and outbound traffic. Note that enabling the strong ES model makes the system unable to function as an IP router.

- **`ip_ire_cmpt_route_lookup_policy/`
`ip6_ire_cmpt_route_lookup_policy`**

Required for the HP-UX Containers product when using networking. Controls the route lookup logic in the compartment-enabled environment. Set this feature to 0 to enable the strong security model which requires strict route lookup logic; set this feature to 1 to disable the strong security model.

- **`cmpt_allow_local`**

Allows containers on the same server to communicate via network protocols without requiring additional security configuration. Sets the default rule for inter-compartment loopback communications that are addressed to local network interfaces or IP addresses. The default rule only applies if there is no

explicit network compartment network rule matching the communication attempt.

- **cmpt_namedstrs**
Configures named streams to be compartment aware.
- **cmpt_restrict_tl**
Required for the HP-UX Containers product to support ONC services. Configures the streams loopback driver (TL) to be compartment aware.

- **Compartment login (cmptlogin)**

This feature is required for configuring workload containers. Enabling this feature configures the system to control user based authentication (including login) on a per container basis by enabling the `CMPT_LOGIN` flag in the `/etc/cmpt/cmpt.conf` file and verifying that `/etc/pam.conf` includes the required `pam_hpsec` module. See `compartment_login(5)` for more information.

IMPORTANT: If the compartment login feature is enabled, only users in the `srpgrp` group are allowed to login to the global view. By default, all users (with a password) defined in the `/etc/passwd` file are allowed to login to the global view when the `cmptlogin` subsystem is enabled using the `srp_sys` command. To allow additional users to login to the global view, edit the `/etc/group` file and add the new users to the `srpgrp` group.

- **PRM Service (prm)**

This subsystem allows you to associate a PRM group to allocate CPU and memory for a container using HP Process Resource Manager (PRM). The `srp_sys` command enables or disables this service using the `prmconfig(1)` command.

If PRM is configured for a container, PRM can guarantee both a minimum and optionally (if configured) a maximum amount of resource available to a container. If the container has not been configured with PRM service, it automatically uses the resource allocated to the default PRM group, `OTHERS` (in `/etc/prmconf`). By default, both CPU and memory allocation will be enabled. Alternatively, the PRM service can be enabled for allocating CPU only. With the `-setup` option, the user is prompted to choose CPU allocation only. With the `-enable` option, the `prmsubsys` variable can be used to enable CPU allocation only.

- **IPFilter module (ipfilter)**

This service allows you to control the network traffic of the container according to the packet attributes using HP-UX IPFilter. Enabling this service allows you to configure IPFilter rules for the container. Containers created with the IPFilter service will have all their inbound networking traffic blocked and must be enabled on a per container basis.

IMPORTANT: Enabling or disabling IPFilter briefly brings down all IP interfaces on the system, then brings up only the IP interfaces configured in the `/etc/rc.config.d/netconf` and `/etc/rc.config.d/netconf-ipv6` files. HP recommends that you do not enable or disable IPFilter when critical network applications are running, enable or disable IPFilter only when interrupting the network connectivity is not disruptive.

- **IPSec module (ipsec)**

Enabling this service allows you to configure HP-UX IPSec policies for the container. If `IPSec module` is enabled on the system using `srp_sys`, then you can configure the container to

apply IPSec policies to encrypt and authenticate packets between the container IP address and a remote IP address.

- **Secure Shell Daemon for Global View (sshd)**

The Secure Shell daemon (`sshd`) in the global view listens to all IP addresses. These interfere with the Secure Shell daemons in the container. To prevent the Secure Shell Daemon from listening on the containers IP addresses, this service restricts `sshd` to listen to a specific global view assigned IP address.

The default IP address that the `sshd` will listen on is the system primary IP address. You can specify more than one IP address in the global view (see *12.4 Network configuration for the global view* to determine the IP addresses in the global view). If you are using the `srp_sys` command with the `-enable` option, you can use the `sshdlistenip` variable to specify multiple global view IP addresses on which the `sshd` can listen.

- **Migrate Workload Containers (migrate)**

This subsystem attempts to modify the configuration of any existing HP-UX SRP A.02.02 workload SRPs so that it can be supported in the HP-UX Containers A.03.00 and later environment.

NOTE: The `coreset` and `cmptlogin` subsystems are not dynamic features. Enabling or disabling `coreset` or `cmptlogin` requires a reboot.

Example: Enabling default subsystem in interactive mode.

In this example, the user executes the `srp_sys -setup` command to enable HP-UX Containers. The default values are accepted for each prompt by pressing **RETURN**.

```
# /opt/hpsrp/bin/srp_sys -setup

Configure all SRP related subsystems? [y] RETURN

Selected SRP subsystem(s) are: migrate,prm,ipsec,ipfilter,coreset,sshd,cmptlogin

#####
#
# Core subsystems
#
#####
Checking SRP core subsystems ... [ Not Enabled ]

Enable SRP support in core subsystems? [y] RETURN
Enabling Security Containment Compartments ... [ OK ]
Enabling multiple namespace support ... [ Enable On Boot ]
Enabling network strong ES model ... [ OK ]
Enabling network compartment IPv4 routing policy ... [ OK ]
Enabling network compartment IPv6 routing policy ... [ OK ]
Enabling network kernel tunable cmpt_allow_local ... [ OK ]
Enabling network kernel tunable cmpt_namedstrs ... [ Enable On Boot ]
Enabling network kernel tunable cmpt_restrict_tl ... [ OK ]
Enabling SRP system services ... [ Enable On Boot ]
Adding SRP user and group ... [ OK ]

#####
#
# Migrate Workload Containers
#
#####
```


The SRP product has detected at least one workload container that needs to be migrated.

Migration can be performed at anytime by executing the command
'/opt/hpsrp/bin/util/srp_migrate'

Migrate all existing workload containers? [y] n **RETURN**

Specify which workload containers to migrate? [n] **RETURN**

#####

#

cmpt Login configuration

#

#####

Checking compartment login feature ... [Not Enabled]

Enable the Compartment Login feature? [y] **RETURN**

Add "login" to the default service list? [y] **RETURN**

Note: By default, once compartment login is enabled, only the root user (user name of "root") will be allowed to login to the global view.

A login group (default:srpgrp) with access to the global view may be used to allow local users (not named "root") to login to the global view.

Grant a login group access to the global view? [y] **RETURN**

Login group name to be granted access to the global view? [srpgrp] **RETURN**

Allow local users to login to the global view by assigning them to the login group "srpgrp"? [y] **RETURN**

Adding RBAC role (SRPlogin-init) for global view login ... [OK]

Note: Users defined in /etc/passwd are allowed to login to the global view.

To update the list of users (not named "root") allowed to login to the global view, edit /etc/group, and modify the list assigned to the group "srpgrp".

Any service monitored by the pam_hpsec account management module is enabled with compartment login enabled.

The current PAM configuration file (/etc/pam.conf) is the same as the system default PAM configuration file (/usr/newconfig/etc/pam.conf). You can keep it for compartment login purpose.

Updating SRP default services: add login [OK]

#####

#

PRM Setup

#

#####

Checking PRM service ... [Not Enabled]

Enable PRM? [y] **RETURN**

Add "prm" to the default service list? [y] **RETURN**

Enabling PRM service ...

[Enable On Boot]

Updating SRP default services: add prm

[OK]

Enabling PRM autostart at boot-up ...

[OK]

#####

```

#
# sshd configuration
#
#####
Checking sshd configuration ... [ Not Enabled ]

The Secure Shell daemon (sshd) in the global view is listening to all IP
addresses.
This will interfere with Secure Shell daemons in SRP containers.

Restrict the IP addresses that sshd listens to in the global view? [y] RETURN

Enter IP addresses, separated by comma ',': [16.92.124.238]
sshd will then listen on these interfaces: 16.92.124.238
Saving changes to /opt/ssh/etc/sshd_config [ OK ]
Restarting Secure Shell daemon ... [ OK ]

#####
#
# IPFilter Setup
#
#####
Checking IPFilter module ... [ Not Enabled ]

HP recommends that you do not enable or disable HP-UX IPFilter when critical
network applications are running. HP recommends that you schedule enabling or
disabling IPFilter when interrupting network connectivity is not disruptive.

Enable IPFilter for SRP? [n] RETURN

#####
#
# IPsec configuration
#
#####
Checking IPsec module ... [ Not Installed ]

#####
#
# SRP setup completed.
#
#####

Warning: compartment feature change will not take effect until the
system is rebooted.
Warning: requested kernel tunable changes will not take effect until the
system is rebooted.
Reboot system now (cd /; shutdown -r now)? [y] RETURN

```

5.2 Displaying subsystem properties

You can use the `srp_sys -list` command to review the current settings for system-wide configuration options affecting HP-UX Containers. For each of the subsystems configured with the `srp_sys -enable` or `srp_sys -setup` command, the `srp_sys -list` command displays the current configuration status. The possible configuration status values are:

Status	Description
OK	The subsystem is installed, enabled and configuration has been validated.
Enable on Boot	The subsystem requires a reboot, and will be enabled once a reboot has been performed.
Invalid Config	The subsystem is installed and enabled but has failed the configuration

Status	Description
	validation check, or is configured with an invalid option for the HP-UX Containers.
Disable on Boot	The subsystem has been marked to be disabled, but needs a reboot to fully be disabled. Until the reboot has happened, parts of the subsystem will be still enabled.
Not Enabled	The subsystem is installed, but has not been enabled
Not Installed	The subsystem software has not been installed

You can use the `-verbose` option to obtain more detailed information

Example: Listing the subsystem status

```
# /opt/hpsrp/bin/srp_sys -l
Checking SRP core subsystems ...           [ OK ]
Checking compartment login feature ...     [ OK ]
Checking PRM service ...                   [ OK ]
Checking IPFilter module ...               [ Not Enabled ]
Checking IPsec module ...                  [ Not Installed ]
Checking sshd configuration ...            [ OK ]
```

5.3 Disabling HP-UX Containers subsystems

You can use the `srp_sys -disable` command to unconfigure and disable specific HP-UX Containers subsystems, as follows:

- Using the `srp_sys` command with the `-setup` option (`srp_sys -setup`) interactively prompts the user for disabling one or more subsystem.
- Using the `srp_sys` command with the `-disable` option (`srp_sys -disable`) non-interactively disables all the specified subsystems. If no subsystem is provided, it disables all the subsystems enabled for HP-UX Containers.

NOTE: Disabling the `coreset` subsystem will change all the containers state to `stopped`.

Example: Disabling only PRM subsystem in the non-interactive mode.

```
$ srp_sys -disable prm

#####
#
# PRM Setup
#
#####
Checking PRM service ...           [ OK ]
Disabling PRM autostart at boot-up ... [ OK ]
Disabling PRM service ...         [ OK ]
Updating SRP default services: remove prm [ OK ]

#####
#
# SRP setup completed.
#
#####
```

6 Managing containers using the `srp` command

The `srp` command is the command level interface available to configure containers. It allows you to add, update, delete, list, and manage containers (see `srp(1M)`).

6.1 Templates, services, and variables

The `srp` command uses templates to determine the type and actions to be performed on a container. Templates consist of modular units called services which groups the actions performed on each functional component of a container. Variables are elements of a service that you can use to configure or update parts of a service.

Templates

There is a primary template (such as `workload` and `system`) for each container type. When you create a container, you must specify the template using the `-template` option. After you created a container with the primary template, secondary templates can be applied to further define and provision a container. You can also apply the same template multiple times to apply actions that can be performed multiple times per container, such as adding or removing additional network interfaces. See *Part III: Container type specific* for information on the templates provided for each container type.

Services

When you specify the `-service` option, the `srp` command executes an operation on a specified component. HP-UX Containers supports the following services, each service roughly maps to a subsystem used in constructing the container:

Service Name	Description
<code>cmpt</code>	Manages configuration data for an HP-UX Security Containment compartment, which forms the core of the container.
<code>admin</code>	Defines the users and groups allowed to execute the <code>srp</code> command options: <code>start</code> , <code>stop</code> , and <code>status</code> .
<code>login</code>	Defines the system's users and groups that are allowed to login to a container. Applied only for container types that share the file system namespace and service daemons, such as the workload container.
<code>prm</code>	Manages the configuration of the PRM group for the container. You can specify the PRM CPU and memory allocations for the group.
<code>provision</code>	Executes a script containing user provided content to be executed during <code>srp</code> operations excluding <code>start</code> and <code>stop</code> .
<code>network</code>	Manages the network configuration of a container.
<code>init</code>	Manages the core file directory layout and provisioning of system files for the container.
<code>ipfilter</code>	Configures IPFilter rules for the container that restrict inbound IP packets to the container's IP interface.

Service Name	Description
ipsec	Configures HP-UX IPSec policies for the primary network interface of the container.

Variables

You can set variables when you create a container and modify them once the container is created. Examples of variables include `ip_address` and `iface`. *15.8 System templates* lists the variables valid for each system container template. *16.8.1 Workload template* lists the variables valid for each workload container template.

6.2 Creating a container

The `srp -add` command creates a container. The `-t` template option specifies the type of container you are creating. If you do not specify a template, the `srp` command by default creates a workload container.

Use the following command to create a container:

```
srp -a[dd] container_name [-t template] [variable=value]...
```

Where:

container_name Specifies the container name. You cannot use the keywords `system` and `workload` as container names.

template Specifies the template name.
Valid Input: `system`, `workload`.
Default: `workload`.

variable=value The variable parameter makes the `srp` command skip the prompt for the specified variables in the interactive dialog and use the specified value. You can specify one or more variable parameters. For example, you can specify the IP address and interface for a container as follows:

```
# srp -add myContainer ip_address=192.0.2.1 iface=lan1:1
```

The `srp` command will skip the prompt for the `ip_address` variables in the interactive dialog and use the specified value.

This feature is often used with the `-batch` option, as described in *6.6 Non-interactive srp command invocation*.

To create a system container, enter:

```
# /opt/hpsrp/bin/srp -add container_name -t system
```

To create a workload container, enter:

```
# /opt/hpsrp/bin/srp -add container_name -t workload
```

In both cases, the *container_name* is the name of the container that you want to create.

In example 6.1, a system container is being created. The user specifies `myContainer` for the container name. In the `srp` dialog, the user specifies `shared` for the container's subtype, the IP address `192.2.2.1` for the container address, and `lan1` for the network interface. The user accepts the default values for all other variables. By default, the system container subtype is `private`.

Example 6.1 Creating a system container using the srp command

```
# /opt/hpsrp/bin/srp -a myContainer -t system
Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

Services to add: [cmpt,admin,init,prm,network,provision] RETURN
Container's subtype (private, shared): [private] shared
Autostart container at system boot? [yes] RETURN
Root user password :
Reenter root user password:
Configure DNS Resolver? [no] RETURN
List of Unix user names for container administrator: [root] RETURN
PRM group name to associate with this container: [myContainer] RETURN
PRM group type (FSS, PSET): [FSS] RETURN
PRM FSS group CPU shares: [10] RETURN
PRM FSS group CPU cap (press return for no cap): []RETURN
PRM group memory shares: [10] RETURN
PRM group memory cap (press return for no cap): []RETURN
PRM group physical memory (press return for no dedicated memory): RETURN []
IP address: 192.2.2.1
Add IP address to netconf file? [yes] RETURN
IP subnet mask (press return to accept default): []RETURN
Network interface name: [] lan1
Gateway server IP address: [192.2.2.1] RETURN
```

The following template variables have been set to the values shown:

```
iface          = lan1
ip_address     = 192.2.2.1
root_password  = *****
```

Press return or enter "yes" to make the selected modifications with these values. Do you wish to continue? [yes] RETURN
add compartment rules succeeded

```
add RBAC admin role for compartment succeeded
add prm rules succeeded
copying from /etc to /var/hpsrp/myContainer/etc
copying from /opt to /var/hpsrp/myContainer/opt
copying from /sbin/init.d to /var/hpsrp/myContainer/etc/init.d
copying from /sbin/rc0.d to /var/hpsrp/myContainer/etc/rc0.d
copying from /sbin/rc1.d to /var/hpsrp/myContainer/etc/rc1.d
copying from /sbin/rc2.d to /var/hpsrp/myContainer/etc/rc2.d
copying from /sbin/rc3.d to /var/hpsrp/myContainer/etc/rc3.d
copying from /sbin/rc4.d to /var/hpsrp/myContainer/etc/rc4.d
copying from /var to /var/hpsrp/myContainer/var
Mounting loopback (LOFS) filesystems ...
copying newconfig directories ...
Copying run level scripts ...
Creating userdb files ...
Changing root user password...
Configuring srp user and group ids ...
Configuring sshd...
Configuring software distributor ...
Configuring RBAC ...
Configuring device files ...
Configuring container products ...
Unmounting loopback (LOFS) filesystems ...
add compartment network service rules succeeded
```

6.3 Starting and stopping a container

If a container is configured to autostart during creation, then it is automatically started at system startup time. All the containers are automatically stopped at system shutdown time. You can use the `srp -start` command to start a container and the `srp -stop` command to stop a container.

To start a container, enter:

```
srp -start container_name
```

To stop a container, enter:

```
srp -stop container_name
```

Where `container_name` specifies the name of the container.

An application startup and shutdown can be included in the container startup and shutdown sequence, see *11 Container startup and shutdown*.

Upon successful completion of a start, the container state will be transitioned to the `started` state, as shown in Example 6.2. After successful completion of a stop, the container will be transitioned to the `stopped` state, as shown in Example 6.3. See the `/var/adm/syslog/syslog.log` file for errors reported by the `srp` command during the start or stop execution. In the case of a system container, the container `/etc/rc.log` file and the container `syslog.log` file will contain the startup and shutdown errors encountered inside the system container.

Example 6.2 Starting a container

```
# srp -start myContainer
```

```
HP-UX SRP Start-up in progress
```

```
Setting up Containers ..... OK
Remounting Root File System ..... N/A
Setting hostname ..... OK
Start containment subsystem configuration ..... OK
Start Utmp Daemon : manages User Accounting Database .... OK
Recover editor crash files ..... OK
List and/or clear temporary files ..... OK
Clean up old log files ..... OK
Start system message logging daemon ..... OK
Checking user database ..... OK
Configuring DHCPv6 Interfaces ..... OK
Starting HP-UX Secure Shell ..... OK
Start NFS core subsystem ..... OK
Start NFS IPv6 subsystem ..... OK
Start enhanced NFS IPv6 subsystem ..... OK
Start NIS server subsystem ..... OK
Start ldap client daemon ..... N/A
Start NIS client subsystem ..... OK
Start lock manager subsystem ..... OK
Start NFS client subsystem ..... OK
Start AUTOFS subsystem ..... OK
Finish containment subsystem configuration ..... OK
Start Internet services daemon ..... OK
Start remote system status daemon ..... N/A
Starting sendmail [Done] Starting sm-client [Done] ..... OK
Starting the password/group assist subsystem ..... OK
Start print spooler ..... N/A
Start clock daemon ..... OK
Initialize Software Distributor agent daemon ..... OK
Starting the Winbind Daemon ..... N/A
Starting HP-UX Apache-based Web Server ..... N/A
```

```
Starting HP-UX Tomcat-based Servlet Engine ..... N/A
Starting HP-UX Webmin-based Admin ..... N/A
Starting the HP-UX Webproxy subsystem ..... N/A
Starting HP-UX XML Web Server Tools ..... OK
```

```
The HP-UX SRP Container is ready.
#
```

Example 6.3 Stopping a container

```
# srp -stop myContainer
```

```
SRP stop in progress
```

```
Stopping HP-UX Apache-based Web Server ..... OK
Stopping HP-UX Tomcat-based Servlet Engine. .... N/A
Stopping HP-UX Webmin-based Admin ..... OK
Stopping the HP-UX Webproxy subsystem ..... OK
Stopping HP-UX XML Web Server Tools ..... OK
Shutting down the Winbind Daemon ..... OK
Stop clock daemon ..... OK
Stop print spooler ..... OK
Stopping HP-UX Secure Shell ..... OK
Shutting down sendmail [Done] Shutting down sm-client [Done] .... OK
Stopping remote system status daemon ..... N/A
Stopping Internet services daemon ..... OK
Stop AUTOFS subsystem ..... OK
Stop NFS client subsystem ..... OK
Stop lock manager subsystem ..... OK
Stop NIS client subsystem ..... OK
Stop ldap client daemon ..... OK
Stop NIS server subsystem ..... OK
Stop NFS core subsystem ..... OK
Unconfiguring DHCPv6 Interfaces ..... OK
Stop system message logging daemon ..... OK
Stop Software Distributor agent daemon ..... OK
Stop Utmp Daemon ..... OK
Killing user processes ..... OK
Starting the fsdaemon for transition into single-user mode ..... OK
Srp container cleanup ..... OK
```

```
HP-UX SRP Container transition to run-level 0 is complete.
```

6.4 Displaying the status of a container

Use the `srp -status` command to display a status summary for the specified configured container:

```
srp -status [[container_name] [-verbose|-xml[output]]
```

Where:

container_name Specifies the name of an existing container. If you do not specify a container name, `srp` displays information about all the containers configured on the system.

verbose Verbose mode. Displays detailed status data.

xmloutput Display output in XML format.

In Example 6.4, the `myContainer` system container is in the `started` state and has the `shared` subtype.

Example 6.4 Displaying the status of a container

```
# srp -status myContainer
```

NAME	TYPE	STATE	SUBTYPE	ROOTPATH
myContainer	system	started	shared	/var/hpsrp/myContainer

6.5 Updating container configuration

You can update an existing container configuration as follows:

- Update services (such as change network configuration and PRM configuration)
- Update variables (such as IP addresses and network interface)
- Add new services (such as IPFilter and IPSec)
- Delete services (such as IPFilter and IPSec)
- Apply secondary templates to workload containers (such as the Apache template)

The `srp` command options `-add`, `-replace` and `-delete` are used to update a container configuration. Use the `-add` option to add new services or templates, the `-replace` option to update services or variables, and the `-delete` option to delete a service or an entire container configurations and its contents.

To add template or service data to a container, enter:

```
srp -a[dd] container_name [-t template[,template]...] [-s service[,service]...] [-i[d] instance] [variable=value]...
```

To delete template or service data to a container, enter:

```
srp -d[delete] container_name [-t template[,template]...] [-s service[,service]...] [-i[d] instance]
```

To replace template or service data to a container, enter:

```
srp -r[eplace] container_name [-t template[,template]...] [-s service[,service]...] [-i[d] instance]
```

Where:

container_name Specifies the name of an existing container.

template Specifies the template names.
Valid Input: workload, system, apache, tomcat, custom, oracledb, sshd.
Default: Template configured for the container.

service Specifies the name of the service.
Default: All services configured for the template.

instance Unique string identifier used to identify an instance of a template usage for templates that can be applied multiple times (such as adding IP addresses)
Valid Input: A text string with alphanumeric, dash (-), or underscore (_) characters. The maximum length is 20 characters.
Default: None.

`variable=value` A valid variable for the specified service when used with the specified template and the value for the variable. To list the valid variable names for a service used with a given template, you can use the `srp -h -v template template_name -service service_name` command.

See 6.7 *Displaying help text for input parameters* to display help text for the input parameters template, service, instance and variable.

CAUTION: If you do not specify the `-template` and/or `-service` arguments with the `-delete` option, `srp` deletes all templates and/or services for the container. For example, the command `srp -delete myContainer` deletes the entire `myContainer` container configuration, and optionally the file system contents.

The `srp -replace` command prompts for new data for each specified service and replaces the entire data set for the service with the replacement data. If you do not supply new values for an option or variable, the HP-UX Containers default value will be used (not the currently configured value).

In Example 6.5, the user adds the `sshd` template to the `myContainer` workload container. The user specifies the `cmpt` and `provision` services and accepts the default values for all variables.

Example 6.5 Adding the `sshd` template to a workload container

```
# srp -add myContainer -t sshd
```

Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

```
Services to add: [cmpt,provision] RETURN
sshd data path: [/var/hpsrp/myContainer/opt/ssh] RETURN
sshd executable path: [/opt/ssh] RETURN
Copy SSH config data from path: [/opt/ssh/newconfig] RETURN
sshd port number: [22] RETURN
```

```
Press return or enter "yes" to make the selected modifications with these
values. Do you wish to continue? [yes]
add compartment rules succeeded
add provision service succeeded
#
```

In Example 6.6, the user wants to replace the PRM data and increases the number of CPU shares to 20. The PRM data is configured with the workload template. The template is derived from the container configuration, the user does not have to specify the template name with the `-replace` operation.

Example 6.6 Replacing PRM data of a workload container

```
# /opt/hpsrp/bin/srp -replace myContainer -s prm
```

Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

```
PRM group name to associate with this SRP: [myContainer] RETURN
PRM group type (FSS, PSET): [FSS] RETURN
PRM FSS group CPU shares: [10] 20 RETURN
PRM FSS group CPU cap (press return for no cap): []RETURN
PRM group memory shares: [10] RETURN
PRM group memory cap (press return for no cap): []RETURN
PRM group shared memory (press return for no dedicated memory): []RETURN
```

The following template variables have been set to the values shown:

```
prn_cpu_shares = 20
```

```
Press return or enter "yes" to make the selected modifications with these
values. Do you wish to continue? [yes] RETURN
replace prn rules succeeded
#
```

In Example 6.7, the user deletes the container `myContainer` that is in the stopped state but keeps the container's local files and directories.

Example 6.7 Deleting a workload container

```
# /opt/hpsrp/bin/srp -d myContainer
Do you wish to delete the compartment "myContainer"? (yes/no) : [no] y

Processing SRP template sshd ...
delete compartment rules succeeded
delete ipfilter rules succeeded
delete provision service succeeded

Processing SRP template base ...

Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

Delete the container's local files and directories? [no] RETURN

Press return or enter "yes" to process this template.
Do you wish to continue? [yes] RETURN
delete compartment rules succeeded
delete RBAC admin role for compartment succeeded
delete RBAC compartment login role succeeded
delete prn rules succeeded
delete ipfilter rules succeeded
delete ipsec rules succeeded
delete compartment network service rules succeeded
delete ipaddress succeeded
delete compartment service succeeded
#
```

6.6 Non-interactive srp command invocation

To run the `srp` command in the non-interactive mode, use the `-batch` or `-b` option. Instead of prompting the user for input, `srp` uses the default values for input variables. If there is no input default value for an input variable, you must specify the value in the command line. For example:

```
# /opt/hpsrp/bin/srp -add myContainer -batch ip_address=192.0.2.2 iface=lan1
```

6.7 Displaying help text for input parameters

Use the following command to display `srp` help text and information about input parameters:

```
srp -h[elp] [-t template[,template]...] [-s service[,service]...]
```

Where:

<i>template</i>	Specifies the templates for which you want to display parameters. Valid Input: <code>workload</code> , <code>system</code> , <code>apache</code> , <code>tomcat</code> , <code>custom</code> , <code>oracledb</code> , <code>sshd</code> . Default: <code>workload</code>
<i>service</i>	Specifies the services for which you want to display parameters. <i>15.8 System templates</i> lists the services valid for each system container template. <i>16.8.1 Workload template</i> lists the services valid for each workload container template. Default: The default services that are valid for the template. The factory configured default services are: <code>admin</code> , <code>cmpt</code> , <code>init</code> , <code>login</code> , <code>network</code> , and <code>prm</code> . You can configure an alternate set of default services via <code>srp_sys -setup</code> , as described in <i>9 Using the srp_ps command</i> .

6.8 Listing container configuration information

Use the following command to display the configuration summary or details for the containers configured on the system:

```
srp -l[ist] [container_name] -v[erbose] [-t template[,template]...] [-s service[,service]...] [-i[d] instance][-x[mloutput]]
```

Where:

<i>container_name</i>	Specifies the name of an existing container. If you do not specify a container name, <code>srp</code> displays information for all the containers configured on the system.
<i>verbose</i>	Verbose mode. Displays detailed configuration data. If not specified, the <code>list</code> operation will display only the names of templates and services configured.
<i>template</i>	Specifies the templates for which you want to list configuration. Valid Input: <code>workload</code> , <code>apache</code> , <code>tomcat</code> , <code>custom</code> , <code>oracledb</code> , <code>sshd</code> . Default: All templates currently applied to the HP-UX Containers.
<i>service</i>	Specifies the services for which you want to display configuration. Default: All services currently applied to the HP-UX Containers.
<i>instance</i>	Unique string identifier used to identify an instance of a template usage for templates that can be applied multiple times. This is valid for the custom template only and is ignored for all other templates. Default: All instances for the specified service(s) and template(s).
<i>xmloutput</i>	Display output in XML format.

6.9 Copying containers by exporting and importing

You can export and import a container across systems by using the `srp -export` and `srp -import` commands. These commands allow you to accomplish the following:

- Create a clone of a container on a secondary system for high availability or load balancing purposes.

- Migrate a container across systems: export and import a container, then delete the original container.
- Create a new container by copying an existing container: export a container, then import the container with a new name and network configuration.
- Create a copy of a container for archival purposes. Similarly, a container can be taken offline by exporting and deleting the original container.

In Example 6.8, the user runs the `srp -export` and `srp -import` commands to copy the container named `myContainer` to a new container named `newContainer`.

When you copy a container with a new name, it is by default configured to start at run-level 2 instead of the run-level configured in the original container. You should reconfigure the services in the new container, as needed, and update the run-level in the container's `/etc/inittab` file to the desired value.

Example 6.8 Copying a container

```
# srp -export myContainer -xfile myContainer.x

/var/hpsrp/myContainer will require 4829788K octets in the exchange file

Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

Save SRP container directories in exchange file? [yes] RETURN

Press return or enter "yes" to process this template.
Do you wish to continue? [yes] RETURN
export compartment rules succeeded
export RBAC admin role for compartment succeeded
export prm rules succeeded
export ipfilter rules succeeded
export ipsec rules succeeded
archiving etc home net opt tmp var bin lib .profile .sw .srp ...
export system product list succeeded
export compartment network service rules succeeded
export provision service succeeded

# srp -import newContainer -xfile myContainer.x

running fitness tests ...

=====
Container "myContainer" exported from mySystem on Thu Jun 30 18:41:17 PDT 2011
=====

Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

Default run-level: [2] RETURN
IP address: 192.2.2.2
Add IP address to netconf file? [yes] RETURN
IP subnet mask (press return to accept default): []RETURN
Primary network interface name: [] lan0
Gateway server IP address for default route (0 to skip) [192.2.2.2] RETURN

The following template variables have been set to the values shown:

    iface          = lan0
    ip_address     = 192.2.2.2

Press return or enter "yes" to make the selected modifications with these
values. Do you wish to continue? [yes] RETURN
```

```

import compartment rules succeeded
import RBAC admin role for compartment succeeded
import prn rules succeeded
import ipfilter rules succeeded
import ipsec rules succeeded
creating mount points ...
restoring filesystems: this will take some time ...
Running product fitness test ...
Scanning the products in "newContainer" please be patient....
Scanning the patches in "newContainer" please be patient....
Scan complete, "newContainer" matches the global products/patches.
Configuring device files ...
Configuring srp user and group ids ...
import compartment network service rules succeeded
import ipaddress succeeded
import provision service succeeded
=====
NOTE: the original container's fstab has been copied to
"/var/hpsrp/newContainer/etc/fstab.orig" and a new fstab file has been created.
=====
#

```

6.9.1 Using the `srp -export` command

The `srp -export` command exports the configuration data and optionally, specified directories, for the specified container. To export the container, the `srp -export` command copies the container's data and stores it in the specified exchange file. The exchange file is used by the `srp -import` command.

The `srp -export` command has the following syntax:

```

srp -export container_name [-xfile <exchange_file>] [-b[atch]]
    [variable=<value>...]

```

Where:

<i>container_name</i>	Specifies the name of an existing container.
<i>exchange_file</i>	Specifies the exchange file name. The file is created if it does not already exist. The default is <code>srp.exchange</code> .
<i>batch</i>	Run <code>srp -export</code> in batch mode. The command will not prompt for arguments or template variable values.
<i>variable=value</i>	The most commonly used variables for the <code>export</code> operation are <code>ok_export_dirs</code> and <code>export_copy_dirs</code> . Setting <code>ok_export_dirs=yes</code> causes the command to export the <code>root</code> directory for the container without specifying <code>export_copy_dirs</code> . Otherwise, set <code>export_copy_dirs</code> to a comma separated list of directory names to be copied to the exchange file. If both <code>ok_export_dirs</code> and <code>export_copy_dirs</code> are not specified, no directories are exported, and only the system configuration that defines the container is included in the exchange file. This is useful for archiving the container definition, and for cloning a container across systems where the entire container home directory will be mounted via shared storage.

NOTE:

- To export the directories mounted via the `fstab` (`/var/hpsrp/container_name/etc/fstab`) file of the container, you must mount the directories prior to executing the `srp -export` command, as follows:
 1. Start the container:
`#srp -start container_name`
 2. Export the container:
`#srp -export container_name`
 3. Stop the container:
`#srp -stop container_name`
- For workload containers: User and group definitions are system properties that are not container specific and therefore are not exported with the container. The compartment login permissions for users and groups allowed to login to the container will be exported, but you must ensure that the required users and groups are configured on the target system, or are accessible via a common name service, such as LDAP-UX.

For example, to export `myContainer` and all directories under the `myContainer` home directory, enter:

```
# srp -export myContainer export_copy_dirs=/var/hpsrp/myContainer
```

6.9.2 Using the `srp -import` command

The `srp -import` command imports the container contained in the specified exchange file, optionally assigning the imported container a new name. The exchange file contains the previously exported container's configuration, and possibly filesystem directories. The `srp -import` command validates the ability of the target system to accept the exchange file and configures the new container. You can only import a container that does not exist on the target system.

NOTE: If the container in the exchange file exists on the system, the `import` command must specify a new container name that does not exist on the system.

You can save the exchange file for archival purposes, create a copy of the original container by importing it with a different container name, or you can import the container on another system without changing the container name.

For system and workload containers: If the new container's name is different from the source container's name in the exchange file, then the source container's network configuration will not be imported. Instead, you will be required to provide new network configuration in the same manner as the `add` operation. If the imported container has the `ipsec` and `ipfilter` services configured, these will be updated to use the newly provided IP address for the container.

The `srp -import` command has the following syntax:

```
srp -import container_name [-b[atch]] [-preview] [-xfile <exchange_file>]  
[variable=<value>...]
```

Where:

`container_name` Specifies the name of the new container.

`batch` Runs `srp -import` in batch mode. The command will not prompt for arguments or template variable values.

<i>preview</i>	Checks whether this system will accept the exchange file for import. If you specify the <code>preview</code> option, only import validation is performed.
<i>exchange_file</i>	Specifies the existing exchange file name. The <code>srp -export</code> command creates the exchange file. The default is <code>srp.exchange</code> .
<i>variable=value</i>	The most commonly used variables for the <code>import</code> operation are as follows: <ul style="list-style-type: none"> • <code>iface</code> The primary network interface name, that is a name without a colon such as <code>lan0</code>, on which to assign the imported IP address of the container. Imported IPv4 network interfaces will be assigned a secondary interface name, for example <code>lan0:1</code>, if the primary is already configured in the <code>/etc/rc.config.d/netconf</code> file. • <code>ip_address</code>, <code>ip_mask</code>, <code>iface</code>, <code>gw_ip_address</code>, and <code>assign_ip</code> When the container name is different from the name in the exchange file, you must provide the network configuration for the new container. • <code>runlevel</code> The default run level for the container's <code>/etc/inittab</code> file. When a container is copied with a new name, it is by default configured to start at run-level 2 which causes the container to start with minimal services. You should reconfigure the services in the new container, as needed, and then update the run-level in the container's <code>/etc/inittab</code> file to the desired value. If the container is not imported with a new name, the default value for the run level is 3. • <code>workload containers: allow_sw_mismatch</code> Set to <code>yes</code> to force an import operation to succeed if the required products for the container are not present or if there is a software version mismatch from the source to the target system. The default is <code>no</code>.

You can use the following notation to assign a value to a variable:
`name=value`, `name='value'`, or `name="value"`

6.9.3 Best practices for exporting and importing a container

To simplify the export and import of a container across systems, HP recommends you to keep the properties of containers atomic by not sharing files and data with other containers unnecessarily. The following are the best practices for using the `srp -export` and `srp -import` commands:

- **Maintain consistent OS versions and patch levels across systems**
System containers require consistent OS versions and patch levels across the systems where a container is being cloned. For workload containers, HP-UX Containers have minimal requirements for OS levels and patches to be synchronized across systems, applications utilized within the containers may have more specific requirements. By maintaining consistency across systems, you will not have to manually track application dependencies

when determining a target system for the container to import.

- **Consider using shared storage or file systems when creating a container that will be cloned**

For workload containers: By using shared storage for a container's home directory and any file systems mounted within the container, you will not need to export and import file sets, and the data between containers will remain consistent.

For system containers: Containers should only share application directories and data across systems.

- **Keep files and directories used by the container within the container's home directory**

For workload containers: locating files within the container home directory

(`/var/hpsrp/container_name`) will simplify exporting or mounting container file sets. For

system containers: All container files and directories must be located inside the container home directory.

- **Avoid cross-compartment security relationships**

If you customize the container's compartment configuration to include access rules specifying other compartments, the other compartments must exist on the target system where the container will be imported.

- **Synchronize or centralize user and group management**

For workload containers: By ensuring that all HP-UX Containers installed systems have the same users and groups, a container's login user and groups, as well as file and directory ownership, will be consistent after importing a container to a different target system.

Consider defining a single group for all users of a given workload container.

For system containers: By ensuring that all system containers and the global view share the same user and group definitions, mapping of user and group IDs to names from the global view will be consistent with that of each system container.

- **Use the `-preview` option to identify a suitable target system**

You can use the `srp -import` command with the `-preview` option to validate if a target system will accept an exchange file for import.

For system containers: You must import the new container on a system that has the same version of the HP-UX 11i v3 Operating Environment as the version that was on the system where the source container's exchange file was created. Otherwise, you will not be able to start the new container until the target system is updated with the `update-ux` command.

The `-preview` option lets you verify whether the Operating Environment for both systems are identical.

- **Adjust device and file system configuration after import**

Configuration that specifies physical paths such as network interface devices and file system mount points require manual configuration changes after import. HP recommends that you adjust these device configurations after completing the import operation.

7 Using the `srp_init` command

The `/sbin/srp_init` command serves as a daemon and as a command to interact with the daemon. The `srp_init` daemon is the first process launched inside a container when the container is started. The `srp_init` daemon is a container process spawner that launches processes based on the entries in the container's `/etc/inittab` file. It also monitors the processes it spawns during the lifetime of the container.

7.1 Changing the run level of a container

You can use the `srp_init` command to communicate with the `srp_init` daemon to change the run level of a container. The `srp_init` command must be executed from within a container to target the correct `srp_init` daemon. Similar to the `init(1M)` command, it accepts a one-character argument and signals the `srp_init` daemon with the `kill(2)` system call to perform the appropriate action. The default run level is set to 3 and is defined in the `/etc/inittab` file (`/var/hpsrp/container_name/etc/inittab` for a workload container).

The `srp_init` command has the following syntax:

```
/sbin/srp_init 0|1|2|3|4|5|6|Q|q
```

Where `0|1|2|3|4|5|6|Q|q` identifies the specified run level.

NOTE: The container run level is independent of the system run level described in `rc(1M)`. Containers cannot be started on the system until the system level `init` daemon has reached run level 3.

The `srp_init` has the following restrictions:

- `S|s,a,b,c` run level options supported by system `init` (see `init(1M)`) are not recognized by the `srp_init` command.
- Boot authentication as described in `security(4)` is not supported
- The `getty` entries in the container's `inittab` file are ignored
- As HP-UX Containers does not support per container console, console interaction including internally restarting the container after `srp_init 0` is not supported. After executing `srp_init 0` from within the container, you must stop and restart the container from the global view via the `srp` command.
- The process ID of `srp_init` will not be 1. The process ID of `srp_init` is non-deterministic.

The `SIGPWR` signal is masked (ignored) by `srp_init`. All other signals remain unmasked by `srp_init`, hence `srp_init` can be killed by an appropriately privileged process.

If the `srp_init` daemon is not running in a container, you can restart it in one of the two ways:

1. `/sbin/srp_init` - Restarts `srp_init` at the last recorded run level in the per container `utmp` database.
2. `/sbin/srp_init <run level>` - Restarts `srp_init` to a specified run level. In this case, `srp_init` starts with the last recorded run level in the per-container `utmp` database, and then transition to the new run level specified.

NOTE: When running HP-UX commands in the workload container that reference the `utmp`, `wtmp`, and `btmpt` database files (for example, `who`, `last`, `write`, and `login`), the database files referenced are in the global view. Because workload containers do not have private copies of these database files, the output is not specific to the container, but instead to the global view.

For a workload container, the old and new run levels will be updated in the `/var/hpsrp/container_name/var/opt/hpsrp/tmp/utmp.runlvl` file.

7.1.1 Log files

Log files can include the following information:

- The `srp_init` messages such as change in run level, warnings, and errors are logged to `/srp.log`.
- The output from the scripts in the `inittab` file executed by the `srp_init` daemon is logged in the `/etc/srp_console.log` file.
- The startup and shutdown log messages are logged in the `/etc/rc.log` file.

For workload containers, prefix the file name with the path `/var/hpsrp/container_name/` to view the log files.

8 Using the `srp_su` command

The `srp_su` command executes the `su(1)` command in the specified container. It can be used to login to a container or execute a single command before returning to the global view. You must execute the `srp_su` command from within the `global` view.

The `srp_su` command has the following syntax:

```
srp_su container_name [su_arguments]
```

Where:

container_name Specifies the name of the target container.

su_arguments Specifies any valid `su(1)` arguments.

8.1 Allowing additional users to use the `srp_su` command

Only users with the `hpux.srp.exec` authorization are allowed to use the `srp_su` command. By default, only the `root` user has this authorization for all containers on the system.

To allow additional users to use the `srp_su` command, you must create a new RBAC role and assign the role to the additional users, as follows:

1. Assign a role to each user:

```
# roleadm assign user_name newRole
```

NOTE: Repeat step 1 for each additional user.

In this example, the `root` user establishes a session as `root` in the target container. The `root` user logs in to `myContainer` container from the global view:

```
# srp_su - myContainer
```

To assign authorization to a non-root user `admin1` to login to a container using the `srp_su` command, follow these steps:

1. Assign the `SRPsu-myContainer` role to user `admin1`:

```
# roleadm assign admin1 SRPsu-myContainer
```

To verify that the role was assigned to `admin1` in the global view, change the user ID to `admin1`.

Then use the `srp_su` command to create a login session in `myContainer`, as follows:

```
# srp_su myContainer - admin1
```

The correct `admin1` user password will allow `admin1` to login to the container `myContainer`.

NOTE: The audit records of the `srp_su` target user are attributed to the source user (the user running the `srp_su` command). To attribute the audit records to the target user instead of the source user, add the line `SU_AUDIT_TAG=1` in the global `/etc/default/security` file.

9 Using the `srp_ps` command

When you run the standard `ps` command in the global view, it will report all the processes on the system including all the processes in containers. To report process status for a specific container from the global view, use the `/sbin/srp_ps` command. The `srp_ps` command invokes the `ps` command with the provided `ps_arguments` and filters the `ps` output for the desired container:

```
srp_ps [container_name] [ps_arguments]
```

When you run the `srp_ps` command in the global view and specify a `container_name`, `srp_ps` prints the process status for the specified container. If you do not supply a `container_name`, the `srp_ps` command prints process status for the global view only, excluding processes from all the containers. When you run `srp_ps` within a container, only the process status for the current container is reported.

To report the process status for a container, login to the global view and enter:

```
# srp_ps myContainer -eaf
```

NOTE: Reports from the global view that include processes running in a system container will display user, group, and command string information in an altered form. See [15.2.1 Displaying system container user, group, and process names from the global view](#) for more information.

To report process status for the global view, login to the global view and enter:

```
# srp_ps -ef
```

10 Establishing a user session in a container

HP recommends the following methods to establish a user session in a container:

- **srp_su**
Use this command from the global view to establish a user session within any type of container. Note that by default, this command is restricted to the `root` user.

For example:

```
# srp_su myContainer
```

See *8 Using the srp_su command* for instructions on how to use the `srp_su` command.

- **Secure Shell (SSH)**
SSH is the only supported method of establishing a remote session into a workload container. Since each container has an exclusive IP address for access, this IP must be used to reach the SSH service inside the desired container.
- **telnet/rlogin**
Both `telnet` and `rlogin` services are supported inside a system container to establish a remote user session. As with workload containers, each system container has an IP address configured for exclusive access to it. Use the container's hostname or specific IP address to establish a connection with the `rlogin` or `telnet` service of the desired system container.

NOTE: For workload containers, login access will be restricted to a set of HP-UX users and groups specified with the login service in the `workload` template. If you did not apply the login service to the workload container, then login access is unrestricted.

11 Container startup and shutdown

The HP-UX Containers startup and shutdown script (`/sbin/init.d/srp`) executes when the system transitions to run level 3 during startup, and executes when the system transitions down to run level 2 during shutdown. The following startup and shutdown scripts are linked to `/sbin/init.d/srp`:

```
/sbin/rc3.d/S999srp
/sbin/rc2.d/K001srp
```

Containers that are configured to autostart will be started when the `/sbin/rc3.d/S999srp` script is run at system startup. You can start any container manually using the `srp -start` command after the system has reached run level 3. During system shutdown, the `/sbin/rc2.d/K001srp` script is run, which will automatically stop all containers that are started.

Container startup and shutdown works similarly to the system startup and shutdown. Each container home directory contains a run level subtree hierarchy at `/var/hpsrp/container_name/sbin` similar to the system `/sbin` subtree, and includes the directories `init.d`, `rc0.d`, `rc1.d`, `rc2.d`, `rc3.d`, and `rc4.d`. When the container's `srp_init` daemon is launched on startup, it processes the container `inittab(4)` file and spawns the configured run level entries for the startup run level (run level 3). One of the default `/etc/inittab` entries is `/sbin/srp_rc`, which is similar to the `/sbin/rc` script, which invokes the scripts configured in the run level subtree in the container `/sbin` directory. See `rc(1M)` for more information.

When you execute either `srp -start container_name` or `srp -stop container_name`:

- In the case of `-start`, the `srp` command launches `srp_init` inside the container and `srp_init` transitions from run level 0 to the default run level specified in the container's `/etc/inittab` file. In the case of `-stop`, the `srp` command instructs `srp_init` to transition from the container's current run level, down to run level 0.
- For each run level, `srp_init` runs the `/sbin/srp_rc` `inittab` entry, which executes the scripts in the container's `/etc/rcN.d` directory in alphanumeric order (where `N` is a run level number). When the container starts, `srp_rc` traverses the `rc` directories in ascending order and executes the scripts in the directories. When the container shuts down, `srp_rc` traverses the `rc` directories in descending order and executes the scripts in the directories.

Under normal circumstances, the container's run level processing handles terminating any remaining user processes via the `/sbin/rc0.d/K800killall` script. If there are any processes still running in the container after all run-level processing is completed, the `srp` command will forcibly terminate all the remaining processes. If after repeated attempts one or more processes still remain in the container, the stop operation will fail.

You can automate the activities that you want performed when a container is started or stopped by adding your startup/shutdown scripts in the container `rcN` directories. HP recommends that you follow the practices recommended in the `rc(1M)` man page for creating and deploying your start and shutdown scripts.

NOTE: Container run level is independent of system run level. Containers cannot be started on the system until the system level init daemon has reached run level 3.

The container setup and shutdown script (`/var/hpsrp/container_name/.setup/setup`) are executed in the global view before and after startup and shutdown of the container. You can modify the script to perform container management activities at container start and stop time that cannot be performed inside the container, such as notifying management or auditing systems, or mounting the container home directory (`/var/hpsrp/container_name`).

NOTE: If you are using shared storage to mount the container home directory to facilitate cloning of a container, consider using the container setup script to automatically mount and unmount the container home directory. In addition to ensuring that the home directory is available when the container is started, it will also help prevent the accidental deletion of the home directory when deleting one of the container clones.

12 Networking with containers

All container types require an IP address to allow network interaction with remote systems. All network activity between a container and other network endpoints will use the IP address dedicated to the container. Once you have assigned an IP address to a container, the IP address can only be used while the container is started. When the container is stopped, the IP address is unavailable.

When you run applications and services in a container that perform network interface and endpoint queries, the container reports only interfaces and endpoints that are associated with the container. For example, running `netstat -i` or `netstat -r` generates output that is context relevant for the container in which you run the command.

You can use the `srp` command to perform basic network administration of a container, such as assign an IP address to a container, and unconfigure a container's IP address when the container is deleted. The `srp` command automatically configures the IP address up and down when the container is started and stopped.

For each container, when you assign an IP address, you can also configure a default route entry. The `srp` command activates the container route entries along with the IP address whenever the container is started, and removes the route entries when the container is stopped. You can manually configure additional route entries for the container.

An additional routing feature available to HP-UX Containers is the ability to designate routes between containers on the system server as Force-to-Wire routes. Force-to-Wire route entries cause all network traffic passing between two different IP addresses on the same server to be forced down through the physical network layer. Then go onto the network where the packets will traverse through any firewalls and routers before being delivered back to the server through the interface where the target container IP address is assigned. You can use this feature to ensure that remotely implemented firewall policy and network auditing is applied to all container network traffic.

Network administration is only supported from the global view.

12.1 Configuring the first network interface

When you run the `srp -add` command to create a container, you must specify an IP address for the container. You will be prompted to enter the necessary information. The parameters that you must specify to configure the primary network interface for a container include:

<i>ip_address</i>	An IPv4 or IPv6 formatted address
<i>ip_mask</i>	(IPv4 only) A mask of the form 255.255.255.0
<i>iface</i>	Physical interface name of the form <code>lan0</code> or <code>lan0:2</code>
<i>gw_ip_address</i>	Gateway IP address for the default route entry

The IP address that you specify must not already be configured in the `/etc/rc.config.d/netconf[-ipv6]` file, or the `srp` command will prompt you to correct your entry. The `ip_mask` parameter only supports IPv4 addresses.

If an IPv6 address is specified, a default prefix length (64) will be used when the interface is configured. You can manually edit the `/etc/rc.config.d/netconf-ipv6` file entry to include the nonstandard `IPV6_PREFIXLEN` as needed once the entry is created.

You can specify any valid physical interface for the `iface` parameter that the `ifconfig(1M)` command will accept. You do not have to specify a logical interface format (`lan0:x`); the `srp` command will find the next available, unassigned logical interface for the physical interface that you specify. Once assigned, however, the logical interface is statically configured in the `/etc/rc.config.d/netconf[-ipv6]` configuration (unless you opted to not manage the network interface in the `netconf` file). If you specify a global unicast IPv6 address for the container, and the specified interface has not already been configured with a link-local address, then the `srp` command will create the primary link-local IPv6 address for the interface, and assign it to the global view before finding an available logical interface to assign the specified container IPv6 address.

You should configure a default gateway entry for your container if you want the container to be able to route outside of the local network. For IPv4 interfaces, the default value is the primary IP address of the container (which will configure a metric count of zero [0]). Your network must have an intelligent router attached in order to successfully use this type of default gateway configuration. If your router does not support this feature, then specify the appropriate gateway IP address for the router associated with your container's subnet. If you do not want to specify a default route (for example to control or limit the connectivity for your container), you can specify zero (0) for the `gw_ip_address` value, and the default gateway entry will be omitted from the container configuration.

Refer to the `route(1M)` and `ifconfig(1M)` for more information about how interfaces and routes are managed.

12.2 Configuring an additional network interface

When you require a container to own more than one IP address, use the `srp -add containername -service network -id <instance>` command syntax to prompt you through the additional values needed to create an additional interface. Any instance value not already in use for that container will work (2 or greater, since 1 is always used for the first IP instance). The prompts and variables are the same for additional instances as specified above for the first IP address.

Example 12.1: Adding the address 192.168.1.22 to the existing container myContainer

```
# srp -a myContainer -s network -id 2 ip_address=192.168.1.22 iface=lan0
ip_mask=255.255.255.0
add compartment network service rules succeeded
add ipaddress 192.168.1.22 succeeded
```

Example 12.2: Adding the address fe80::3 to the existing container myContainer

```
# srp -a myContainer -id 3 -s network -b ip_address=fe80::3 iface=lan0
add compartment network service rules succeeded
add ipaddress fe80::3 succeeded
```

Example 12.3: Displaying network configuration

```
# srp -l myContainer -v -s network

Name: myContainer  Template: system Service: network ID: 1
-----

Compartment Configuration (/etc/cmpt/myContainer.rules):
// owns the IP address
interface          16.92.121.29

Netconf Configuration:
INTERFACE_NAME="lan0:1"
```

```

INTERFACE_SKIP="true"
IP_ADDRESS="16.92.121.29"
TYPE="ipv4"
SUBNET_MASK="255.255.252.0"
INTERFACE_STATE="up"
BROADCAST_ADDRESS=""
DHCP_ENABLE="0"
INTERFACE_MODULES=""
CMGR_TAG="compartment="myContainer" template="system" service="network" id="1"
ROUTE_DESTINATION="default"
ROUTE_SKIP="true"
ROUTE_MASK=""
ROUTE_GATEWAY="16.92.121.29"
ROUTE_COUNT="0"
ROUTE_ARGS=""
ROUTE_SOURCE="16.92.121.29"

```

Name: myContainer Template: system Service: network ID: 2

Compartment Configuration (/etc/cmpt/myContainer.rules):

```

// owns the IP address
interface          192.168.1.22

```

Netconf Configuration:

```

INTERFACE_NAME="lan0:4"
INTERFACE_SKIP="true"
IP_ADDRESS="192.168.1.22"
TYPE="ipv4"
SUBNET_MASK="255.255.255.0"
INTERFACE_STATE="up"
BROADCAST_ADDRESS=""
DHCP_ENABLE="0"
INTERFACE_MODULES=""
CMGR_TAG="compartment="myContainer" template="system" service="network" id="2"

```

Name: myContainer Template: system Service: network ID: 3

Compartment Configuration (/etc/cmpt/myContainer.rules):

```

// owns the IP address
interface          fe80::3

```

Netconf Configuration:

```

INTERFACE_NAME="lan0:1"
INTERFACE_SKIP="true"
IP_ADDRESS="fe80::3"
TYPE="ipv6"
INTERFACE_STATE="up"
IPV6_PREFIXLEN=""
DHCP_ENABLE="0"
CMGR_TAG="compartment="myContainer" template="system" service="network" id="3"
#

```

Example 12.4: Displaying the status of the container myContainer

```
# srp -status myContainer -v
```

```
SRP Status:
```

```
----- Status for SRP:myContainer -----
```

```
Status:STARTED
```

```
Type:system Subtype:shared Rootpath:/var/hpsrp/myContainer
```

```
IP:16.92.121.29 Interface:lan0:1 (UP)
```

```
IP:192.168.1.22 Interface:lan0:4 (UP)
```

```
IP:fe80::3 Interface:lan0:1 (UP)
```

```
MEM Entitle:25.00%    MEM Max:(none)    Usage:6.85%
CPU Entitle:7.69%    CPU Max:(none)    Usage:0.00%
```

If you need to modify any of the managed networking service fields for an existing container instance, use the `srp -replace containername -service network -id <instance>` command to alter one or more of the existing values for the container instance. The instance value should exist in the container configuration. Refer to the `srp -list containername -service network -v` output for appropriate instance reference.

Example 12.5: Replacing the network configuration for id 3

```
# srp -r myContainer -s network -id 3 ip_address=fe80::4 iface=lan0
```

Enter the requested values when prompted, then press return.
Enter "?" for help at prompt. Press control-c to exit.

```
Add IP address to netconf file? [yes]
Gateway server IP address: [0]
```

```
Press return or enter "yes" to process this template.
Do you wish to continue? [yes]
replace compartment network service rules succeeded
replace ipaddress fe80::4 succeeded
```

12.3 Configuring additional routing entries

You can configure additional routing entries for each container by modifying the global view `/etc/rc.config.d/netconf[-ipv6]` configuration file. HP recommends that you copy an existing route entry for the chosen container, and then modify it so it reflects the desired settings for the container's new route entry. The `ROUTE_SOURCE` field for the new entry must match a valid IP address for the container (this serves as the tag for route entries). Also, to make sure that the route entry is only applied when the IP address is available for the container, configure the `ROUTE_SKIP` field for the new route entry to "true". Edit the appropriate `netconf` file (`netconf` for IPv4 addresses and `netconf-ipv6` for IPv6 addresses) as follows:

1. Copy an existing routing entry that matches the container IP address for which you want to add a new routing entry.

For example:

```
ROUTE_DESTINATION[1]="default"
ROUTE_SKIP[1]="true"
ROUTE_MASK[1]=" "
ROUTE_GATEWAY[1]="<GatewayIP>"
ROUTE_COUNT[1]="1"
ROUTE_ARGS[1]=" "
ROUTE_SOURCE[1]="<container_IP>"
```

2. Change the array index ([1] in the example above) for each variable in the new block to an unused number (for example, [10] – they do not have to be sequential).

3. Modify the necessary fields to create a net route entry for the 192.168.1.0 network, keeping the container IP address of the `ROUTE_SOURCE[]` entry intact for the matching container.

For example:

```
ROUTE_DESTINATION[10]="192.168.1.0"
ROUTE_SKIP[10]="true"
```

```
ROUTE_MASK[10]="255.255.255.0"
ROUTE_GATEWAY[10]="<Gateway2_IP>"
ROUTE_COUNT=[10]="1"
ROUTE_ARGS[10]=" "
ROUTE_SOURCE[10]="<container_IP>"
```

The `<Gateway2_IP>` field must be reachable from the `<container_IP>` address by being on the same subnet as the `<container_IP>` address.

NOTE: You should be familiar with shell scripting when managing the network configuration, as well as syntax for the `route(1m)` and `ifconfig(1m)` commands.

12.4 Network configuration for the global view

Any IP address that is not associated with a container will be managed as belonging to the global view. When you run the `srp_sys -setup` command, the tool defines a compartment rule that associates all global view IP addresses with the `INIT-private` compartment. When a process running in the global view needs to connect to another system, the source IP address is chosen from a set of global view IP addresses. To see the list of IP addresses that are active for the global view, run `getrules -T INIT-private` to generate a list of addresses that are available.

Several fields in the `/etc/rc.config.d/netconf` file have significance in the HP-UX Containers enabled configuration. Fields to note include:

<code>INTERFACE_SKIP</code>	Determines whether <code>/sbin/net</code> script will enable the interface entry. A default value of <code>"false"</code> implies the entry will be created on system boot and when the script is manually run. A container interface should always be set to <code>"true"</code> .
<code>INTERFACE_STATE</code>	Determines the default state of an interface entry. Valid values are <code>"up"</code> , <code>"down"</code> , and <code>"force"</code> . The default is <code>"up"</code> . The <code>"force"</code> state is used in conjunction with interfaces that support force-to-wire behavior.
<code>ROUTE_SKIP</code>	Determines whether <code>/sbin/net</code> script will enable the route entry. A default value of <code>"false"</code> implies that the entry will be created on system startup and when the script is run manually. A container route entry should always be set to <code>"true"</code> .
<code>ROUTE_PARAMS</code>	Any trailing arguments that the <code>route(1m)</code> command supports can be included. The <code>"force"</code> option can be used for route entries where the force-to-wire behavior is needed.
<code>ROUTE_SOURCE</code>	Explicitly identifies the source IP address to be used for the route entry. This is required by all container route entries, and HP recommends it for all global view routes as a best practice.
<code>IPV4_CMGR_TAG</code> and <code>IPV6_CMGR_TAG</code>	Container specific field inserted by the <code>srp</code> command to manage all container related entries in the <code>netconf</code> file or <code>netconf-ipv6</code> file. This field is required for every container interface entry to identify the associated container and the container instance id.

When manually managing route and interface entries in the `/etc/rc.config.d/netconf` and `/etc/rc.config.d/netconf-ipv6` files, pay attention to the index of each array variable that is specified. There is no correspondence in the index used for route entries and the index used for interfaces. The `IPV4_CMGR_TAG` and `IPV6_CMGR_TAG` values are used to identify container specific interface entries; the associated index for each `INTERFACE_*` variable is applied to the IP

address configuration. The `ROUTE_SOURCE` IP address is used to identify the correct container route entries.

Connectivity between containers and the global view are permitted by default. The routing between each area is managed internally without going out on the physical network. If routing between containers, or between a container and the global view is desired to leverage the physical interface and any firewall rules that may be enforced, then a `force` option exists that allows this behavior to be configured (see *12.6 Cross-container network traffic and Force-to-Wire*).

12.5 Address collisions with `INADDR_ANY` and `IN6ADDR_ANY` sockets in the global view

Service daemons that bind to the wildcard `INADDR_ANY` address (or `IN6ADDR_ANY` for IPv6) and run in the global view can inadvertently provide an access point for connection attempts to one of the active container IP addresses, if that container does not also bind to the same TCP or UDP service port. For this reason, HP recommends that you configure service daemons running in the global view to bind to a specific global view IP address instead of `INADDR_ANY`. For services that do not support an explicit IP address binding configuration, HP recommends configuring the service in a workload container.

12.6 Cross-container network traffic and Force-to-Wire

Containers provide isolated networking environments. By default, a container is configured so that the only networking traffic allowed is through the container-specific IP interface. Inter-system container networking can be configured through loopback networks. HP-UX Containers A.03.00 and later includes the Force-to-Wire feature where system inter-container network traffic can be routed to the external network. This can facilitate network monitoring, among other uses.

When connectivity between two locally assigned IP addresses needs to be forced to the network, both IP addresses must be tagged with the `force` option. Also, the desired route to be forced must be tagged in both directions. The `ifconfig(1m)` and `route(1m)` commands both support the `force` option to enable this configuration.

For example:

Container1 has IP 10.1.1.2 on lan0

Container2 has IP 10.2.2.5 on lan1

Router IP1 on 10.1.1.0 subnet, 10.1.1.1

Router IP2 on 10.2.2.0 subnet, 10.2.2.1

```
# ifconfig lan0 10.1.1.2 force
# ifconfig lan1 10.2.2.5 force
# route add host 10.1.1.2 10.2.2.1 1 source 10.2.2.5 force
# route add host 10.2.2.5 10.1.1.1 1 source 10.1.1.2 force
```

NOTE: To automate the `force` option for an interface entry, configure the corresponding `INTERFACE_STATE[]="force"` in the `netconf` file. You can add a new entry in the `netconf` file for the route entry as follows:

```
ROUTE_DESTINATION[9]="host 10.1.1.2"
ROUTE_SKIP[9]="true"
ROUTE_MASK[9]=" "
ROUTE_GATEWAY[9]="10.2.2.1"
ROUTE_COUNT[9]=1
ROUTE_ARGS[9]=" "
ROUTE_SOURCE[9]="10.2.2.5"
```

```
ROUTE_PARAMS[9]="force"
```

NOTE: Configuring cross-container rules can interfere with the ability to import containers to another system. See *6.9 Copying containers by exporting and importing* for more details.

12.7 IP Routers and strong end system (ES) model

To ensure proper routing, HP-UX Containers configures the system to use the strong end system (ES) model, as described in RFC 1122 to provide symmetric routing of connection based network traffic. When the strong ES model is used, a system cannot act as an IP router. A system with the strong ES model silently drops incoming IP packets with destination IP addresses that do not match the interface address. Outbound IP packets must use the interface address as the source IP address.

12.7.1 Application gateway servers

Although HP-UX Containers systems cannot be used as IP routers, they can be used as application gateway servers. Application gateway servers receive IP packets sent to a local IP address, process the packets at an upper layer, and retransmit the packets using the local IP address as the source address.

13 Using Serviceguard with containers

Serviceguard allows you to create high availability clusters of HP 9000 or HP Integrity Servers. A high availability computer system allows application services to continue in spite of a hardware or software failure. Highly available systems protect users from software failures as well as from failure of a system processing unit (SPU), disk, or local area network (LAN) component. In the event that one component fails, the redundant component takes over. Serviceguard and other high availability subsystems coordinate the transfer between components.

You can use Serviceguard to provide high availability to your HP-UX Containers deployment. Serviceguard can manage a Serviceguard package executing within a container, or manage the container itself as a Serviceguard package.

13.1 Choosing a model

HP offers two different models when using Serviceguard with HP-UX Containers: the **classic model** and the **container package model**.

In the classic model, the container is in the `started` state and Serviceguard has not yet started managing the application inside the container. This model is most compatible with the existing Serviceguard packages.

In the container package model, the container itself is the Serviceguard package. This model takes advantage of the capabilities of HP-UX Containers by simplifying the Serviceguard scripts and allowing application startup and shutdown to be managed by HP-UX Containers. Serviceguard starts and stops the container; and the container initialization and shutdown process starts and stops the applications within the container. This model simplifies the Serviceguard packages and requires less maintenance and administration of startup and shutdown activities. With this model, you can choose either Serviceguard or HP-UX Containers to control the file system mounting and the network interface management.

13.2 Creating a container to use with Serviceguard

If you want to create a container that will use Serviceguard, you must first determine how HP-UX Containers and Serviceguard will interact together. The following steps will give you the information that you need to configure a container appropriately:

1. **Select the model**

If you have existing Serviceguard control scripts that you want to leverage, HP recommends that you use the classic model. For a new deployment of a Serviceguard package, HP recommends that you use the container package model as it is easier to create.

2. **Select which application will have the control**

Determine whether HP-UX Containers or Serviceguard will control the mounting of file systems and management of the network interface, as follows:

- If you selected the classic model in Step 1, HP recommends using Serviceguard to control the mounting of file systems and management of the network interface. If you selected the container package model in Step 1, HP recommends using HP-UX Containers to control the file system mounting and management of the network interface.

- If you want to use the Serviceguard network failover capability, then Serviceguard must control the management of the network interface.

IMPORTANT: Unlike HP-UX Containers, Serviceguard does not support the system network configuration files `/etc/rc.config.d/netconf` and `netconf-ipv6`. Therefore, a Serviceguard package during startup can unknowingly use container assigned network interfaces which are not active when the package is started, but are configured in `/etc/rc.config.d/netconf` or `netconf-ipv6` for a container's use. When the container with the conflicting network interface is started, the active Serviceguard package can fail or result in loss of network connectivity. As a rule, a Serviceguard managed container and a non-Serviceguard managed container on the system must not share the same physical network interface.

3. Create the container

When you create a container that will use Serviceguard, you must indicate in the Container Manager or the command line interface to support the desired Serviceguard behavior as follows:

Add IP address to netconf file? [yes]

Press **Enter** to instruct HP-UX Containers to control network interface management.

Enter **no** to defer control of network management to Serviceguard.

NOTE: If you use the `srp` command for configuration, you can use the variable `assign_ip=yes|no` to specify the behavior.

This option informs HP-UX Containers whether or not the container controls the starting and stopping of the assigned network interface. Either option may be used with Serviceguard, but selecting "no" allows Serviceguard to control the interface, allowing support of network interface failover.

Autostart SRP container at system boot? [yes]

Press **Enter** for the classic model.

Enter **no** for the container package model.

NOTE: If you use the `srp` command for configuration, you can use the variable `autostart=yes|no` to specify the behavior.

For Serviceguard network failover capability, you need to create a secondary (failover) container. To create a secondary container, you can use the export and import features to clone the container on a secondary system. See *6.9 Copying containers by exporting and importing* for more details.

The sharing of container home directory (using Serviceguard volume) between cloned containers in different physical systems is only supported for workload containers and is not supported for system containers.

13.3 Adapting Serviceguard scripts for the classic model

The example in this section demonstrates the classic model approach to modify an existing Serviceguard script to work with HP-UX Containers. If you want Serviceguard to manage or monitor the applications executing within the managed container, use the `srp_su` command to let

Serviceguard access the container. You must prepend the `srp_su` command to the command that requires execution within a container.

In the following example, the representative Serviceguard package was modified to control `myContainer`, a package executing in the container. The `service_cmd` value is the only value that changed in the script:

Before:

```
service_name      service_ping
service_cmd       "/usr/sbin/ping node_a"
service_restart   unlimited
service_fail_fast_enabled no
service_halt_timeout 300
```

After:

```
service_name      service_ping
service_cmd       "/opt/hpsrp/bin/srp_su myContainer root -c
`/usr/sbin/ping node_a`"
service_restart   unlimited
service_fail_fast_enabled no
service_halt_timeout 300
```

Either HP-UX Containers or Serviceguard can manage the network interfaces. If Serviceguard is managing the network interfaces, HP recommends that the package is configured to create the default route for any container IP address. In the following example, the representative Serviceguard package was modified to add a default route, `external_script`:

Before:

```
# SG ip address
ip_subnet          192.10.25.0
ip_address         192.10.25.12
```

After:

```
# SG ip address
ip_subnet          192.10.25.0
ip_address         192.10.25.12
# srp_route_script configures the required source based routing entries for
# the SG managed IP addresses
external_script    /etc/cmcluster/pkg1/srp_route_script
```

See *Appendix A: Container default route script for Serviceguard* for an example of the `srp_route_script` script.

13.4 Creating Serviceguard scripts for the container package model

The container package model manages the container as a Serviceguard package. At package start time, Serviceguard launches the package, which starts the container and relies on the container initialization process to mount any application specific file systems and start the applications. Similarly, at stop time, the package initiates the container stop request, relying on HP-UX Containers to perform any application shutdown and file system unmounting.

Monitoring of the package is accomplished using the same method as described for the classic model.

With workload containers, the container package model allows you to share the entire container home directory between primary and secondary containers. If you choose to share a common home

directory, then the Serviceguard package should mount and unmount the home directory before starting and stopping the container.

Either HP-UX Containers or Serviceguard can manage the network interfaces. Similar to the classic model, if Serviceguard is managing the network interfaces, HP recommends that the package is configured to create the default route for any container IP addresses. See *Appendix A: Container default route script for Serviceguard* for an example.

You can find a reference implementation of the container package model at:
`/opt/hpsrp/example/serviceguard/srp_as_sg_package/`. The `README` in the directory provides instruction to create a functional example of container package model.

Part III: Container type specific management

14 Container types

HP-UX Containers supports deployment of containers of different types on the same system. In order to choose the container type best suited for your workload, you should determine which container properties best meet your needs.

14.1 System containers

System containers provide additional virtualization and private namespace capabilities over workload containers that give users the look and feel of a private operating system instance. Most applications can be deployed in a system container environment using default installation values.

Each system container has:

- A unique host, node, and domain name
- Local users and groups
- Local NIS or LDAP domain
- Local password policies
- Local file system view (private or shared)
- Local system services (for example, `init`, `sshd`, `pwgrd`, `syslogd`, and `inetd`)
- Private network interface
- Private IPC namespace
- NFS client and AutoFS support

14.2 Workload containers

Workload containers provide a lightweight workload hosting environment. All workload containers share file system view, hostname, IPC namespace, and service daemons with the global view, providing a restricted view of these resources, rather than a virtualized view. This restricted view of a workload container can be customized to meet the workload's requirements.

Like all containers, a workload container has a private directory under `/var/hpsrp` that is only accessible by that container. Because workload containers share the entire file system with the global view and other workload containers, they may share application directories with other workload containers, for example, `/opt/my_application` is shared across all workload containers. See [16.1 File](#) for a detailed layout of a workload container file system access.

Workload containers can be created very quickly and require little disk space. Upfront application deployments may require customization compared to other container types, however ongoing maintenance of a workload container will typically be less than other container types as most system administration activities for workload containers are shared with the global view.

14.3 Comparison of system and workload types

The following table compares the properties for workload containers, system containers with the private file system subtype, and system containers with the shared file system subtype. These subtypes are described in the following section.

Property	Workload Container	System Container (private FS)	System Container (shared FS)
Disk space consumed per container	Negligible	9 GB	4.5 GB

Property	Workload Container	System Container (private FS)	System Container (shared FS)
Memory overhead per container	Negligible		
CPU/networking/storage access overhead per container	Negligible		
CPU and memory allocation controls	Guaranteed minimum or dedicated		
Private namespace support	Network portspace	<ul style="list-style-type: none"> • Hostname/Nodename/Domainname • IPC • File system (chroot based) • Network portspace • Loopback IP address portspace 	
Processes	Isolated	Isolated	
System services provisioned per container	Secure Shell (optional)	<ul style="list-style-type: none"> • Secure Shell (optional) • inetd • syslogd • utmpd • pwgrd • cron • swagentd • rpcbind • NFS client daemons 	
Lifecycle	Per container init processing, start, stop, import, and export.		
User management	Managed from the global view. The system administrator identifies users allowed to login to the global view and/or containers.	Managed within the container. Per container /etc/password, /etc/group, /etc/nsswitch.conf, /etc/pam.conf files that are provided in the container.	
IPC considerations	IPC objects are accessible from the global view.	IPC objects such as semaphores, message-queues, and shared memory are exclusive to the container and are not visible in the global view.	
NFS considerations	None.	Mounts done within the container are exclusive to the container; they are not visible or accessible from the global view.	
Device management	No restrictions.	Devices can be created only in the global view.	

Property	Workload Container	System Container (private FS)	System Container (shared FS)
		To mount a device within a container, you must first provision the device to the container from the global view.	
SD software installation	Installed once from the global view. Products with targeted install location may be installed into container.	Installed from the global view and pushed to each container.	Installed from the global view and pushed to each container. Containers must be in the stopped state.
SD Product Installation restrictions	None	Only products that are on the allowed list will be pushed to the containers.	
Import target system software requirements	HP-UX Containers product version compatibility.	Full SD database equality test (source and target system must match).	
Cloning considerations	Full container directory tree may be shared.	Container directory tree must be copied (application data may be shared).	
Networking considerations	<ul style="list-style-type: none"> • Managed from the global view • Loopback portspace shared among all workload containers and the global view 	<ul style="list-style-type: none"> • Managed from the global view • Fully private network portspace for each system container 	
Default security properties	<ul style="list-style-type: none"> • Cannot access other container files • Read-only access to most system files • Process view restricted to the container • IPC restricted to the container and the global view • No disallowed kernel privileges 	<ul style="list-style-type: none"> • Cannot access other container files • Read-only access /stand • Process view restricted to the container • IPC restricted to the container • Disallowed kernel privileges, for sensitive operations 	<ul style="list-style-type: none"> • Cannot access other container files • Read-only access to system files • Process view restricted to the container • IPC restricted to the container • Disallowed kernel privileges, for sensitive operations
Security hardening/softening	Security settings can be altered (see <i>16.3 Security</i>).	Modifying security settings is not supported.	

14.4 Choosing a container type

HP-UX Containers provide multiple container types to meet consolidation needs of the environment. However, it is important to choose the right container type for the workload before embarking on your consolidation effort.

A workload can be defined as a related set of applications and supporting services – far more than just an application. A consolidation project should start at the workload level not the application to ensure all environmental variables are considered during the planning phase.

The following section attempts to provide an initial guide to help the user determine which type of container is best for the workload in question.

14.4.1 When to use a system container

System containers provide a look and feel of a standalone system and many of the capabilities of a virtual machine guest without the associated management and performance overhead. System containers allow you to migrate more of the workload's operating environment, minimizing process or administrative changes required. When a user logs into a system container they will have private versions of many of the system services they have in a standalone operating system. System containers provide an easier up front migration path for workload consolidation by providing a similar environment to that where the workload came from. You should use system containers when:

- The workload requires a unique hostname
- Users/Group names need to be private to the container
- Application installation does not allow for alternate installation path
- Private system services are required
- Migrating from other container technologies
- Application deployment/modifications are common

A system container provides virtualization features not found in a workload container allowing for more services to be run in on a per-container basis.

14.4.2 When to use a workload container

Workload containers provide a filtered view of all resources on the system, including processes, file systems, and network interfaces. Workload container configuration is highly flexible and allows administrators to set access controls for a workload container that meets the workloads needs.

Because of this flexibility workload containers are ideal for workloads that:

- Require IPC access to other applications running in a separate workload containers
- Interact with services provided in the global view
- Need to share common IP address access with the global view or other workload containers
- Minimum disk space available
- Miss-matched software versions across HA nodes
- Quick setup time
- No requirement for:
 - Unique hostname
 - Private file system
 - Private IPC name space

15 System container

With a system container, you can perform various management tasks only from the global view (see 15.7 Limitations and disallowed), and others from within the container. The management tasks performed within the container are:

- Hostname, nodename and domain name configuration
- User/group management
- Startup and shutdown of service daemons
- Name service configuration
- Scheduling cron jobs
- RBAC
- File system mount (including NFS and dedicated partitions)
- Extended security attributes
- User auditing
- Application installation using methods other than SD

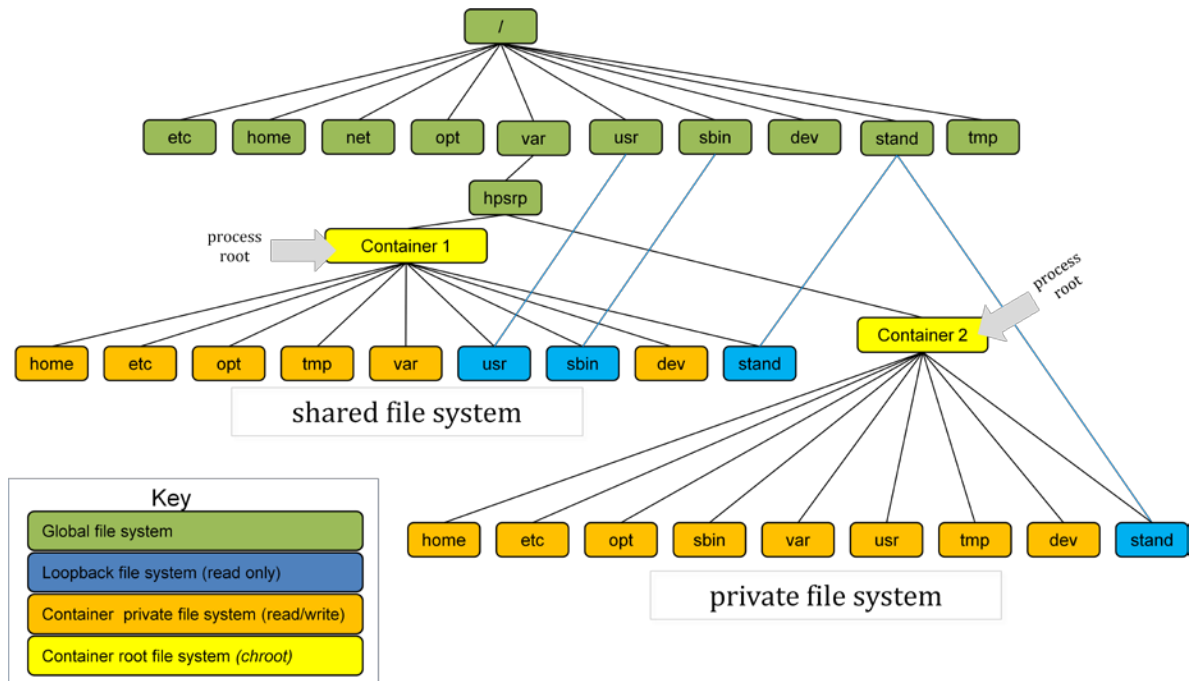
15.1 Managing file system

When you create a container, the `srp` command creates a new container root directory (`/var/hpsrp/container_name`). Access restrictions are set such that only the processes within the container (or the global view) can access files under that directory. All processes running in a system container are chrooted to the container root directory, thus providing a unique file system view for the container.

15.1.1 Choosing a file system subtype

You can create a system container with either a private or a shared file system view. If you create a system container with a **private file system**, it will be populated with its own read/write copy of the system directories, excluding the `/stand` directory. The `/stand` directory will be made available with a loopback mount from the global view `/stand` directory. If you create a system container with a **shared file system**, it will be populated with its own read/write copies of the system directories, excluding the `/stand`, `/usr`, and `/sbin` directories; these directories will be read-only loopback mounted from the corresponding global view directories.

The following diagram shows the file system layout for `Container 1`, a system container with a shared file system and `Container 2`, a system container with a private file system.



A system container with a shared file system subtype has a smaller disk footprint and is faster to create as the system binary directories `/usr` and `/sbin` are shared with the global view. It is ideal for application deployment that does not require write access to `/usr` and `/sbin` and the software version can be in sync with the global view version.

A system container with a private filesystem subtype provides write access to all the directories except `/stand`. This allows the container to have different version of software (except kernel component). It is ideal for application deployment that requires write access to `/usr` and `/sbin`.

See [15.5.1.5 Software management behavior for file system subtypes](#) for the differences in software management for the two types of containers.

15.1.2 Mounting file system

You must perform disk and partition administration from the global view. Typically, these partitions are specified in the global `/etc/fstab` file and are mounted and available at system boot time.

15.1.2.1 Making the container root path mountable

The container root path `/var/hpsrp/container_name` can either be a directory on an existing filesystem or a physical filesystem mount of a disk or its partition. If the root path is a mount point, then the root path must have a corresponding entry in the global `/etc/fstab` file and it must be mounted before you create the container.

15.1.2.2 Adding a new mount for the system container

A container specific `/var/hpsrp/container_name/etc/fstab` file is provisioned in every container. You can configure the entries in the container specific `/etc/fstab` file to do nfs mounts, lofs mounts, hfs, vxfs and cdfs mounts within the container. The entries will be automatically mounted when the container is started and unmounted when the container is stopped.

To perform a physical filesystem mount of a disk or its partition within the system container, you must explicitly provision the corresponding block device (for example, `/dev/disk/disk10`) and the

character device (for example, `/dev/rdisk/disk10`) within the system container as described in *15.4 Managing_Devices* . Since operations such as creating a file system on a device using the `mkfs` command is not allowed within a system container, you must create the file system on a device before provisioning it in the container.

NFS mount must be done within a container. Accessing the NFS mount points created within a container from the global view will result in an error because the NFS mounts done within a system container are exclusive to that container. NFS mounting from the global view to a path under a container `root` directory (`/var/hpsrp/container_name`) is not supported. Mounting the global view files via NFS is possible from within a system container by targeting the global view host IP address when specifying the NFS resource, as the loopback IP interface is private to each system container.

NFS server is supported only in the global view (using the primary system IP address). You must configure directory shares as an absolute path from the global system root directory, and the directories must be physically mounted when the shares are exported.

CacheFS is not supported in a system container.

15.1.2.3 Mount table within a system container

A system container is provisioned with its own version of mounted filesystem table `mnttab` under the directory `/var/hpsrp/container_name/etc`. It contains a table of devices mounted by the `mount` command for the container both by the global administrator and the administrator within the container. The users in the global view have visibility to all the mounts created in the system inclusive of all system container specific mounts.

When a system container is in `started` state, its `mnttab` will have at least the following two entries:

- A root mount point (`/`) associated with the system container of type `srp_dummyvfs` which is automatically created when the container is started. This root entry cannot be mounted or unmounted by a user.
- A mount point (`/stand`) `lofs` mounted to the system's `/stand` directory, which gets mounted by the global administrator when the container is started.

All mount point paths within a system container are always displayed as a path relative to the container root path, while the same mount point paths are displayed as absolute paths in the global view. For example, a mount point created as `/var/hpsrp/container_name/stand` from the global view will be displayed as `/var/hpsrp/container_name/stand` in the global view and as just `/stand` within the system container `container_name`.

Any mount point created from the global view for the system container, though visible within the container, cannot be unmounted by the container administrator. In particular, all `lofs` mounts created by the global administrator will display both the filesystem and the mounted on directory to be identical. For the physical filesystem mounts like `vxfs` or `hfs`, the original block device path like `/dev/disk/disk48` is displayed as the filesystem even if the `disk/device` itself is not provisioned within the system container. As mentioned earlier, NFS mounting from the global view to a path under a container `root` is not supported. Even though the system container specific NFS mounts are still visible in the global view, they still cannot be accessed from the global view.

A sample output from the `bdf` command executed within the system container is shown below:

```
# bdf
Filesystem      kbytes    used    avail  %used Mounted on
/                8192000 4146427 3794023   52% /
/stand          1835008  535520 1289376   29% /stand
```

```
/dev/vg53/lvol4    4096000    30945 3811084    1% /httpperf
/etc/mymnt         8192000 4146427 3794023    52% /tmp/mymnt
```

```
# ls /dev/vg53/lvol4
/dev/vg53/lvol4 not found
```

In the previous sample output example:

- The first entry refers to the container root (/).
- The next two entries are mounts created for the container from the global view.
- The last entry refers to a lofs mount created within the container.

15.2 Users, groups and authentication

System containers are provisioned with a separate set of configuration files and service daemons to manage user and groups, login authentication, and name service resolution. Administration of these activities must be performed inside each system container. Consider using a network-based name service repository such as NIS or LDAP-UX to minimize per-container user administration activity.

While the name mappings of users and groups support per system container variations, the `uid` and `gid` values themselves are absolute with respect to process attributes. The restrictions defined for the container control the access of a process to files, other processes, and network interfaces outside of the container. These restrictions are not affected by the user and group IDs associated to a process (for example, root privilege will not override container restrictions.)

15.2.1 Displaying system container user, group, and process names from the global view

The following information about processes running in system containers is displayed in an altered form to protect against accidental global view manipulation:

- User name and `uid`
- Group name and `gid`
- Process name

When the HP-UX Containers product is enabled, the reserved user name `srp` and the reserved group name `srp` are registered in the global `/etc/passwd` and `/etc/group` databases, and they are propagated to each system container database as well. When the global view lists processes, each system container process is displayed with a user of `srp(24)` and a group of `srp(24)`. These values are intended for display purposes only – no process actually runs with these values. This mapping prevents the accidental manipulation of system container processes by the global view using a process list and pattern match for a legitimate user or group name.

Process listing from the global view displays an altered process name for processes running in a system container. This protection is provided to avoid accidental signaling from the global view by automated scripts that were designed to manage a single instance of a program through the use of pattern matching of the process name. The algorithm used to alter the process name is designed to allow for human recognition of the process while preventing conventional string pattern matching schemes.

While display of process information is altered, the global view ability to signal all processes on the system remains unchanged. Administrators who understand the consequences and want to eliminate the user and group display mapping or the process name display changes can update the tunable values as follows:

```
kctune srp_obfuscate_enabled=0    # to disable all process related obfuscation
kctune srp_obfuscate_enabled=1    # to enable UID and GID obfuscation only
kctune srp_obfuscate_enabled=2    # to enable process name obfuscation only
kctune srp_obfuscate_enabled=3    # to enable all process related obfuscation
(default)
```

15.3 Security features

15.3.1 Extended security attributes

Extended security attributes for binary files (see `setfilexsec(1M)` and `getfilexsec(1M)`) are maintained on a per-container basis for system containers. An administrator in a system container can set, delete, and display extended security attributes of binary files in the system container's filesystem view the same way an administrator manage them in the global view.

Binary files that are under the `/usr` and `/sbin` directories that are shared by the global view and system containers with shared file system can be assigned more than one set of extended security attributes. The attributes that take effect are determined based on where the binary file is executed. When a binary file is executed in the global view, the attributes set by an administrator in the global view take effect. When the same binary file is executed in a system container, the attributes set by an administrator in that system container take effect.

Processes running in the global view can access any file on the system including those in the file system view of a system container. However, extended security attributes set for binary files in a system container take effect only for a process running in that system container. They have no effect on processes running in the global view. This prevents unexpected side effects that might be caused if the binary files were executed with the attributes assigned in a system container. Likewise, extended security attributes set for binary files under the `/usr` and `/sbin` directories in the global view are not visible to and have no effect on processes running in a system container with shared file system.

Some of the privileges included in the extended security attributes are only required for system-wide operations and are disallowed to processes running in a system container for security reasons. If these privileges are assigned to binary files in a system container, they will not take effect. The new process will begin execution without these disallowed privileges. The target compartment information included in the extended security attributes is also ignored to prevent a process running in a system from changing its compartment. See *15.7 Limitations and disallowed operations*.

15.3.2 Audit

The AuditExt v4.0 product included in the HP-UX Containers A.03.01 product supports auditing in both the global view and in system containers.

15.3.2.1 Audit configuration and record generation

Most of the audit administration tasks must be performed by an administrator in the global view, because they require privileges that are disallowed in system containers. See *15.7 Limitations and disallowed operations*. These administration tasks are described in `audsys(1M)`, `audevent(1M)`, `audomon(1M)` and `audfilter(1M)`.

However, selection of users to be audited must be done by an administrator in each system container, because a separate set of users and groups are maintained in each system container. The `AUDIT_FLAG` attribute described in `security(4)` and `userdb(4)` controls which users should be audited.

Once auditing is configured, all audit records generated by processes in all system containers, as well as audit records generated by processes in the global, are written to the audit log files in the global view.

15.3.2.2 Audit record viewing in the global view

An administrator in the global view can use the `auditdp(1M)` command to view audit records generated by processes in any system container or in the global view. When viewed in the global view, audit records generated in system containers can have incorrect mapping between user/group IDs and names. This occurs because the ID-name mapping of an audit event is defined by the `/etc/passwd` (see `passwd(4)`) file that is specific to the container where the event was recorded, not by the one in the global view where the `auditdp(1M)` command is run. The `/opt/audit/AudReport/bin/srp_auditdp_global` sample script can be used to display audit records with correct ID-name mapping.

To view all audit records generated on the system, enter:

```
# auditdp -r global_log
```

To view audit records generated in a specific system container, enter:

```
# auditdp -r global_log -s "+cmpt=container_name;"
```

The `global_log` parameter is the audit log file specified by the `-c` option of the `audsys` command when auditing is started, and the `container_name` parameter specifies the name of the system container where the audit records were generated.

In the above examples, audit records generated in a system container may be displayed with incorrect ID-name mapping. Replace `auditdp` with `srp_auditdp_global` to view audit records with correct ID-name mapping.

15.3.2.3 Audit record viewing in system containers

Audit log files that reside in the global view are not accessible from processes running in system containers. To view audit records in a system container, they must first be copied to log files under the container's filesystem view by the global administrator.

To insure that audit records generated in one system container are not copied to the log files under the filesystem view of another system container, use the `/opt/audit/AudReport/bin/srp_auditdp_copy` sample script.

To copy audit records generated in all system containers, enter:

```
# srp_auditdp_copy -r global_log -R local_log
```

To copy audit records generated in a specific container, enter:

```
# srp_auditdp_copy -r global_log -R local_log -C container_name
```

The `global_log` parameter is the audit log file specified by the `-c` option of the `audsys` command when auditing is started, the `local_log` parameter is the pathname of the target audit log file relative to container's root directory, and the `container_name` is the name of the system container where the audit records were generated. Each system container will only receive those audit records that were generated in that container.

Once audit records are copied, an administrator in a system container can use the `auditdp` command or the `srp_auditdp_global` script to view the audit records generated in the container.

15.4 Managing devices

A system container is provisioned with devices that can operate within the scope of the container only. These devices are:

- Pseudo-transport devices (such as `/dev/tcp` and `/dev/ip`)
- Pseudo-terminal devices (such as `/dev/pty*`)
- Mount device (`/dev/mnttab`)
- Random number generator (`/dev/random`)
- Null device (`/dev/null`)
- Privilege-aware devices that can restrict the operations (such as `/dev/devkrs` and `/dev/config`).

Devices cannot be created from within a system container. Additional devices can be provisioned to a system container from the global view using the `srp` command with the `-tune` option. If the device does not exist on the system, it must be first created in the global view and then provisioned to a container.

To add a device `/dev/aaa` to a container `myContainer`, enter:

```
# mknod /dev/aaa c 88 4
# srp -add myContainer -tune device=/dev/aaa
Configuring device files ...
Done.
```

To delete the device `/dev/aaa` from `myContainer`, enter:

```
# srp -delete myContainer -tune device=/dev/aaa
Deleting /dev/aaa.....
Done.
```

When additional devices are provisioned to a container, the administrator is responsible for maintaining the exclusivity of the device to a container. For example, if a container requires exclusive access to a disk, the administrator must provision the disk to only one container and not provision the same disk device to other containers. All the device files provisioned in a container are accessible from the global view.

15.5 Software management

You can update both the operating system software and application software on a HP-UX Containers enabled system using the HP-UX Software Distributor (SD).

SD is part of the HP-UX operating system and consists of a set of commands. The SD commands `swinstall(1M)`, `swremove(1M)`, and `swconfig(1M)` are enhanced to detect the container configuration and apply the operation in the global view and automatically repeat the operation in all the configured system containers.

The following table shows the SD commands behavior in the global view and a system container.

SD command	Global view	System container
<code>swacl</code>	Unchanged	Unchanged
<code>swask</code>	Unchanged	Unchanged
<code>swcopy</code>	Unchanged	Unchanged
<code>swjob</code>	Unchanged	Unchanged
<code>swlist</code>	Unchanged	Unchanged
<code>swmodify</code>	Unchanged	Unchanged

SD command	Global view	System container
swpackage	Unchanged	Unchanged
swreg	Unchanged	Unchanged
swverify	Unchanged	Unchanged
swinstall	Enhanced	Not supported (blocked)
swremove	Enhanced	Not supported (blocked)
swconfig	Enhanced	Not supported (blocked)

15.5.1 Managing software

15.5.1.1 Installing software

The `swinstall` command is used to install the software selection from a software source. By default, the software is configured to use after installation. On HP-UX Containers enabled system, the software source must be a local software depot, it must not be a remote network registered depot (see *15.5.2 Remote network registered depots*).

The `swinstall` command executed in the global view will install and configure the software in the global view and in all the system containers in the system. For a software product that requires a reboot, the installation of the product occurs first in the global view, then in each system container, followed by a reboot of the entire system. Although the action is repeated in each system container, an HP-UX kernel is not created in each system container. The `/stand` directory that holds kernels is a read-only loopback mounted to the global view `/stand` directory.

NOTE: Only the non-interactive command-line interface (CLI) for SD commands is supported when one or more system containers are configured.

To install `MY_PRODUCT` in the system and then install `MY_PRODUCT` in each system container, enter:

```
# swinstall -s /tmp/myproduct.depot MY_PRODUCT
```

To install a patch that requires reboot, enter:

```
# swinstall -x autoreboot=true -s /tmp/patch.depot PHNE_12345
```

To view the detailed log of the action that was repeated in the system container, execute the `swjob` command inside the container using the `srp_su` command as follows:

```
# /opt/hpsrp/bin/srp_su container_A root -c "swjob -a log container_A-0004"
```

15.5.1.2 Listing software

The `swlist` command is used to display information about software products installed at or available from the specified target selection. See `swlist(1M)` for more information.

The `swlist` command when executed in the global view shows the products installed in the system and, when executed inside a system container, shows the products installed in the container. You can also specify a container hostname with the `-s` option in the global view to list the products installed in a container.

To view the list of installed products in container `myContainer_A`, run the `swlist` command inside the container:

```
# /opt/hpsrp/bin/srp_su myContainer_A
# swlist -l product
# exit
```


15.5.1.3 Removing software

The `swremove` command is used to remove software selection from the system. When removing installed software, `swremove` also unconfigures the software before it is removed. As in the case of installation of a software product that requires a reboot, the removal of the product occurs first in the global view, then in each system container, followed by a reboot of the entire system.

To remove a product from the system and all the configured system containers, execute the `swremove` command in the global view as follows:

```
# swremove MY_PRODUCT
```

15.5.1.4 Configure, unconfigure, or reconfigure installed software

The `swconfig` command is used to configure, unconfigure or reconfigure installed software. The operation is first performed in the global view and repeated in each system container on the system.

If you attempt to run the `swinstall`, `swremove`, or `swconfig` command in a system container, you will get the following error as these operations are not supported in system containers:

```
# srp_su myContainer_A
```

```
# swinstall -s /tmp/myproduct.depot MY_PRODUCT
```

```
ERROR: The command "swinstall" is not supported in local SRP.
```

15.5.1.5 Software management behavior for file system subtypes

The following table shows the differences in update requirements and behavior for a system container with a shared file system subtype and a private file system subtype:

Description	System container (shared FS)	System container (private FS)
Restriction on the container state to perform SD software maintenance	Must be stopped	No restriction (either started or stopped)
Varying software	All software in container must match global	Software that has a kernel component must match global
Filesystem paths that are not updated during the install and remove commands	/usr, /sbin, /stand (LOFS mounts)	/stand (LOFS mount)
Filesystem paths that are updated	/opt, /etc, /var, and /sbin/rc[1-4].d/	/opt, /etc, /var, /sbin, and /usr

15.5.2 Remote network registered depots

Remote network registered SD depots are not supported for software installation.

You will see the following error if a remote depot location is specified with the `swinstall` command:

```
# swinstall -s remote_machine:/depots/remote.depot PRODUCT
```

```
ERROR: The source depot specified using a host target selection (host:/path). Installing from a remote source is not supported in SRP environment. To install from a remote source, either mount it locally or copy the software locally using swcopy.
```

To solve this problem, follow these steps:

1. Use the `swcopy` command to copy the depot to the local system.

2. Copy the remote directory locally using other HP-UX tools.
3. Use NFS to mount the depot on the local filesystem.
4. If the depot is on media, physically place it in a local drive.

For example, you can copy a remote registered depot to the global system as follows:

```
# swcopy -s remote_machine:/depots/remote.depot PRODUCT @ /depots/myproduct.depot
```

15.5.3 Allowed products

The products that are installed in the system containers are a subset of the products installed in the global system. HP has a predefined list of allowed products, as well as a list of HP restricted products that can never be added. The list of allowed products is configurable using the utility `srp_allowed_product`. See `srp_allowed_product(1M)` for more information.

You can add a product to the allowed product list as follows:

```
# /opt/hpsrp/bin/srp_allowed_product -add MY_PRODUCT
```

In this example, after adding your product to the list, when installation is repeated in system containers, the `MY_PRODUCT` product will also be installed. When new system containers are created, `MY_PRODUCT` will be provisioned. Patches to `MY_PRODUCT` will install and be provisioned in system containers.

The products added to the allowed list must be SD product names, they cannot be bundle names, or filesets contained within the products.

To see a list of products in a depot, enter:

```
# /opt/hpsrp/bin/srp_allowed_product -list_depot /tmp/my_product.depot
Products in depot at /tmp/my_product.depot :
-----
MY_PRODUCT
```

To be prompted to add products from a depot, enter:

```
# ./srp_allowed_product -add_depot /tmp/my_product.depot
Products in depot at /tmp/my_product.depot :
-----
Add product MY_PRODUCT? (Enter q to Quit) [y]: y
Verifying MY_PRODUCT can be added to allowed list.
The product MY_PRODUCT was added to the allowed product list.
```

If a product has a dependency on other products then the other products must also be specified in the allowed products list.

15.5.4 Updating the global view and individual system containers

The SD commands `swinstall`, `swremove`, and `swconfig` are enhanced to allow targeting the global view and/or a set of (one or more) system containers.

Two new options `global_srp` and `local_srp_list` are added to support this functionality. The option `global_srp` is used in conjunction with `local_srp_list` to install, remove, or configure software in the global view and inside each system container. The `local_srp_list` contains space separated system container names enclosed in quotes and can be a subset of all configured system containers.

The behavior is as follows:

- If neither `global_srp` nor `local_srp_list` options are specified, then the install, remove, or configure will occur on the system and all system containers.
- If the `global_srp=true` and `local_srp_list` options are not specified, then the install, remove, or configure will occur only on the system.
- If the `global_srp=true` and `local_srp_list` options are set, then the install, remove, configure will occur on the system and the listed system containers.
- If the `local_srp_list` option is set and the `global_srp` option is not specified, then the install, remove, configure will occur only in the listed system containers.

You can use these new options to recover a container after software installation failure (see *15.5.5 Recovering unsynchronized containers*) or when installing/removing a product from the global view that is not part of the system containers.

For example, when removing a product from the global view that is not in the HP-UX Containers allowed product list (therefore, it is not installed in system containers), it would be quicker to target just the global, as follows:

```
# swremove -x global_srp=true PRODUCT
```

When installing a product that is not in the HP-UX Containers allowed product list (and that does not require a reboot), you can avoid having to stop the shared subtype containers by targeting just the global, as follows:

```
# swinstall -x global_srp=true -s /tmp/product.depot PRODUCT
```

15.5.5 Recovering unsynchronized containers

If a software maintenance operation fails in the global view, check the swagent log file `/var/adm/sw/swagent.log` to identify the problem, take corrective action and repeat the operation.

If a software maintenance operation succeeds in the global view but fails in one or more containers then the SD operation will display errors indicating the containers that are out of sync with the global view.

For example, if the installation of the product `MY_PRODUCT` succeeds in global view but fails in `container_A`, then `swinstall` will display the following error:

```
# swinstall -s /tmp/myproduct_version_2.depot MY_PRODUCT
```

```
Scanning the products in "container_A" please be patient...
The SRP product 'MY_PRODUCT' does not match global's version
SRP has version 'A.01.00'      Global has version 'A.02.00'
```

As the message indicates, the system container still has version 1 of the product, but the global view now has version 2 installed.

If the container file system subtype is shared, then the container cannot be started until the software in the container matches the software in the global view. If the file system subtype is private then the container can be started but depending on the type of the update (see *15.5.6 Advanced installation: varying software versions*) the container software must be kept in sync with the global view.

To take corrective actions:

1. Check the `/var/adm/sw/swagent.log` file in the container to identify the problem. Either follow the `ROOTPATH` of the container, or enter the container by changing its state to

maintenance (using `srp -maint container_A`) and then enter the container (using `srp_su -M container_A`).

2. Take corrective action based on the information in the `swagent.log` file.
3. Change the state of the container back to `stopped` using `srp -stop container_A`.
4. Install the product by targeting the container:

```
# swinstall -x local_srp_list="container_A" -s \  
/tmp/myproduct_version_2.depot MY_PRODUCT
```

For `swinstall`, you can reissue the command without targeting the container since the operation will proceed to the system container even if the product is installed in the global view. For `swremove`, you must target the system container that had the issue, since `swremove` will not proceed past a global system in which the remove has already occurred.

In some cases, it can be easier to remove the software from the global view and all system containers and then install it again in the global view and all system containers.

15.5.6 Advanced installation: varying software versions

For a system container with the private filesystem subtype, the version of software with a kernel component must match the global view. Versions of other software that contain only user-space commands, libraries, and GUI can be different from the global view.

During SD operation, you can use the `local_srp_list` option to skip the global view and target one or more containers.

The following example shows the `swinstall` command syntax to install different versions of `MY_PRODUCT` in two different containers:

```
# srp -status  
NAME          TYPE      STATE      SUBTYPE  ROOTPATH  
container_B   system   started    private  /var/hpsrp/container_B  
container_C   system   started    private  /var/hpsrp/container_C  
  
# swinstall -x local_srp_list="container_B" -s \  
/tmp/myproduct_version_1.depot MY_PRODUCT  
  
# swinstall -x local_srp_list="container_C" -s \  
/tmp/myproduct_version_2.depot MY_PRODUCT
```

15.5.7 Installation methods other than SD

For products that are managed through installation methods other than HP-UX Software Distributor, such as the `tar` command, you can install these products inside each individual system container. If the product installs files under the `/usr` or `/sbin` directories, HP recommends that you use a system container with a private file system.

15.5.8 Using `update-ux` with configured containers

The `update-ux` command lets you update your system to a newer HP-UX 11i v3 Operating Environment release. When using this command, perform the `update-ux` policies and procedures as usual, such as backing up the system and stopping production applications.

After executing the `update-ux` command, products and patches are installed in the global system and the system will reboot without prompting. For these reasons, you must stop all containers before running the `update-ux` command. The following error is displayed if there are running containers on the system:

ERROR: The update-ux command cannot continue because at least one containers is in the 'started' state.
Please run 'srp -stop -batch' to stop all running containers.

You can specify a remote depot when using the update-ux command. With HP-UX Containers enabled and with existing system containers configured, the system must have at least 5 GB of disk space available under the /var directory when specifying a remote depot. The following error is displayed if there is less than 5 GB of disk space:

ERROR: Not enough space detected for an update with SRP enabled. At least 5000 MB of free disk space is required in the filesystem where '/var' resides, but currently only 2429 MB is available.

To solve this problem, you can perform one of the following:

- Increase the available disk space under /var
- Provide the HP-UX 11i v3 OE depot as a locally accessible depot

Once you run the update-ux command, the following occurs:

1. Products and patches are installed in the global system
2. The system is rebooted
3. Products and patches are configured in the global system during the execution of the /sbin/rc2.d/S119swm.config run level script
4. An automatic attempt to update containers is done during the execution of the /sbin/rc3.d/S999srp run level script.

Dependent on the number and type of containers this process can take an extended period of time to complete. Allow the S999srp script to run to completion. The bootup progress message on the console stays at "Updating SRPs and Starting SRPs" until completion. Each container that is configured to autostart is started as soon as its update completes. Progress on the update is logged in the /etc/rc.log file, including failure details and corrective suggestions listed in the *Updating SRPs and Starting SRPs* section.

If system containers exist on the system and if a remote depot was specified, an attempt to swcopy the necessary products and patches to a local software distributor depot is made. If a failure occurs, the error message in the /etc/rc.log file indicates that the user needs to provide the HP-UX 11i v3 OE depot as a locally accessible depot and run the srp_sync command.

If system containers exist and if a failure occurs in one or more containers, the local depot is retained and the path logged. After validating the local depot and completing the update using the srp_sync command, you must remove the local depot.

NOTE: If the system includes configured containers, the reinstall=true option of the update-ux command is not supported.

15.6 Deploying applications

While deploying application in a container, be aware of the read-only restrictions for certain directories (/usr, /sbin, and /stand) based on the container's file system subtype.

Follow these best practices when deploying applications with system containers:

- **Install non-SD applications from inside the container.**
This allows the software installation and setup processing to utilize container local namespace values, such as host name, and file path.

- **If you have applied IPFilter for the container, ensure that any additional ports used by the application are allowed.**

When the `ipfilter` service is enabled for the container, by default the inbound network connections to the container are blocked. You must configure the `ipfilter` service to allow inbound connections to any network ports that the application will listen on.

- **Use the custom template to apply additional IPFilter capabilities to the container for the application.**

This will allow you to manage system configuration changes for the container on a per container basis. Use a recognizable identifier, such as the application name for the `instance_id` parameter when deploying the custom template. When deploying multiple applications within a container, consider applying the custom template (if needed) once per application.

15.7 Limitations and disallowed operations

All users in a system container (including `root`) are prevented from performing the following list of administrative tasks. These administrative tasks must be performed in the global view.

- Kernel configuration management
- Kernel tunable management
- System boot configuration
- Reading kernel memory
- Make kernel
- System crash configuration
- KRS (Kernel Registry Services)
- DLKM management
- Creating device files
- Changing system time
- Shutdown/reboot the physical system
- Swap space management
- Logical volume management
- Physical devices management
- Network interface card configuration
- IP Address configuration
- Network tunable configuration
- Compartment rule configuration
- Bypassing compartment rules using overriding privileges
- Enable/disable auditing
- Enable/disable accounting
- IPFilter configuration
- IPSec configuration
- SRP configuration
- Software installation (`swinstall/swremove/swconfig`)

In order to prevent the above disallowed operations, the following privileges (see `privileges(5)`) are disallowed in a system container. Commands and system calls performing the administrative tasks disallowed in a system container will return an error.

Disallowed Privilege	Description	Example
ACCOUNTING	Allows a process to control the process accounting system	acct(1M), acctsh(1M)
AUDCONTROL	Allows a process to start, modify, and stop the auditing system.	audsys(1M)
CHANGECMPT	Grants a process the ability to change its compartment.	privrun (1M)
CMPTREAD	Allows a process to open a file or directory for reading, executing, or searching, bypassing compartment rules.	
CMPTWRITE	Allows a process to write to a file or directory, bypassing compartment rules.	
COMMALLOWED	Allows a process to override compartment rules in the IPC and network subsystems.	
CORESYSATTR	Allows a process to manage system attributes such as kernel tunables and system time.	kctune(1M), date(1M)
DLKM	Allows a process to load a kernel module, change the global search path for DLKM.	kcmodule(1M)
FSS , FSSTHREAD	Allows a process or thread to configure fair share scheduler.	
MKNOD	Allows a process to create character or block special files.	mknod(1M)
MPCTL	Allows a process to change processor binding, locality domain binding, or launch policy of a process.	mpctl(2)
NETADMIN	Allows a process to perform network administrative operations such as configuring IP address and routing tables.	Add, delete, update options of ifconfig(1M), netstat(1M), route (1M)
NETPROMISCUOUS	Allows a process to configure an interface to listen in promiscuous mode.	tcpdump
PSET	Allows change to the system pset configuration.	
RDEVOPS	Allows a process to do device specific administrative operations such as tape or disk formatting.	

Disallowed Privilege	Description	Example
REBOOT	Allows a process to perform system reboot.	reboot(1M)
RULESCONFIG	Allows a process to add and modify compartment rules.	setrules(1M)
SPUCTL	Allows a process to perform certain administrative operations in the Instant Capacity product.	
SWAPCTL	Allows a process to manage and configure swap space.	swapctl(2), swapon(1M)
SYSNFS	Allows a process to export a file system.	
TRIALMODE	Allows a process to log privileges required to execute in the <code>syslog</code> file.	

15.7.1 Disallowed commands

The commands and system calls that fall into the category of disallowed administrative tasks will fail in a system container. The disallowed tasks can be part of a command (certain options) or can be the command itself. Some examples of the disallowed commands are: `accton(1M)`; `acctsh(1M)`; `date(1) -u, -a`; `getprivgrp(1M)`; `ied(1)`; `mknod(1M)`; `mpsched(1)`; `privgrp(4)`; `psrset(1M)`; `ptydaemon`; `reboot(1M)`; `sar(1M)`, `setboot(1M)`; `setprivgrp(1M)`; `setuname(1M) -s`; `shl(1)`; `timex(1) -o, -p`; `umodem(1)`; `uupath(1)`; `who(1) -A, -t`

15.8 System templates

The following table describes the templates that can be included for a system container:

Template	Description
system (required)	The <code>system</code> template is the primary template for system containers. The base configuration includes the following services: <code>cmpt</code> , <code>admin</code> , <code>init</code> , <code>network</code> , <code>prm</code> , <code>ipfilter</code> , <code>ipsec</code> , and <code>provision</code> .
custom (optional)	The <code>custom</code> template enables you to customize the IPFilter configuration. See <i>16.8.6 Custom template</i> .

The following table describes the services and variables included in the `system` template that can be used to update system container configuration:

Service	Variable	Description
admin*	admin_user	Comma separated list of existing user names to be granted the role of container administrators. Default: root
cmpt*	ok_export_dirs	Specify if the container <code>root</code> directories must be saved in exchange file for export operation (<code>srp -export</code>). See <i>6.9 Copying containers by exporting and importing</i> . Default: No
	export_copy_dirs	Comma separated list of fully qualified directory names

Service	Variable	Description
		to be copied during the import and export operations. Default: <code>var/hpsrp/container_name</code>
	<code>allow_sw_mismatch</code>	Allow import to proceed if software products on the source and destination systems do not match. Default: No
<code>init*</code>	<code>autostart</code>	Auto start container at system boot (Yes or No). Default: Yes
	<code>subtype</code>	Specify the container's subtype (private or shared). Default: private
	<code>configure_dns</code>	Specify if DNS resolver has to be configured (Yes or No). Default: Yes
	<code>root_password (+)</code>	Root user password for the container. Default: None (required)
	<code>change_password</code>	Specify whether the container's root user password should be changed when the container is replaced. Default: No
	<code>domain_name</code>	DNS administrative domain to which this container belongs. Default: Same as configured in global view (<code>/etc/resolv.conf</code>)
	<code>dns_server_ip</code>	Comma separated list of IP addresses of DNS Servers. Default: Same as configured in global view (<code>/etc/resolv.conf</code>)
	<code>device_list</code>	Path name of file containing list of devices to be provisioned in a container. Default: <code>/etc/opt/hpsrp/srpdevices.def</code>
	<code>device</code>	Comma separated list of device files to be provisioned. (see <code>srp -tune</code> in <code>srp(1M)</code>) Example: <code>/dev/xx</code> , and <code>/dev/yy</code> . Default: none
	<code>Provision_fs</code>	Specify whether to re-provision the container's files and directories if the container directory already exists. Default: No
	<code>delete_files_ok</code>	Specify whether to delete files and directories used by the container when the container is deleted (<code>srp -delete</code>) or replaced (<code>srp -replace</code>). (Yes or No) Default: No
<code>prm</code>	<code>prm_group_name</code>	Name of the Process Resource Management (PRM) group dedicated to this container. Default: The container name.
	<code>prm_group_type</code>	PRM CPU allocation type (PSET or FSS). Default: FSS
	<code>prm_cores</code>	Number of processor cores allocated (For PSET only). Default: 1
	<code>prm_cpu_shares</code>	Number of CPU shares allocated (For FSS only). Default: 10
	<code>prm_cpu_max</code>	Maximum percentage of CPU available (For FSS only). Default: No cap.
	<code>prm_mem_shares</code>	Specifies a max (upper bound) for memory consumption of system's memory for user processes.

Service	Variable	Description
		Default: 10
	prm_mem_max	Specifies a max (upper bound) for memory consumption of system's memory for user processes. Default: No cap
	prm_phys_mem	Memory in MB allocated for shared memory usage. Default: 0 (no dedicated physical shared memory)
network*	ip_address(+)	IP address to dedicate to the container. Default: None.
	assign_ip	Specify if IP Address has to be configured in network configuration file (/etc/rc.config.d/netconf or netconf-ipv6) . Default: Yes.
	iface(+)	Primary or secondary network interface name for the associated container IP address (for example, lan0). Default: None.
	ip_mask	IP subnet mask for this container (For IPv4 address only). Default: The network mask for the address class, as specified in RFC791 (IETF specification).
	gw_ip_address	Gateway IP address for the default route for the configured container IP address. Enter the value 0 to specify no gateway IP address. Default: Same as the IP address configured for this container.
ipfilter	ipf_for_ipsec	Specify whether to allow IPFilter rules to allow IPsec packets (Yes or No). Default: No.
ipsec	ipsec_peer_addr(+)	Destination IP address for the IPsec policies. Valid Input: An IPv4 address in dotted-decimal notation or an IPv6 address in colon-hexadecimal notation. Default: None
	ipsec_transform	Transform for IPsec host policy. Valid Input: ESP_AES128_HMAC_SHA1 ESP_AES128_HMAC_MD5 ESP_3DES_HMAC_SHA1 ESP_3DES_HMAC_MD5 ESP_NULL_HMAC_SHA1 ESP_NULL_HMAC_MD5 Default: ESP_AES128_HMAC_SHA1 .
	ipsec_psk(+)	Preshared key used by IPsec Peer. Parameter Name: ipsec_psk. Valid value: A text string, containing 1 - 128 ASCII characters (whitespaces are not allowed). Default: None.
provision	script	Run a provision script when a container is operated on by srp command. All input to the command are passed into the script. Default:/opt/hpsrp/bin/util/custom_srpsys_setup.

(*) Required services for a container (or to create a container).

(+) No default, these variables need a value to be assigned for adding or replacing the corresponding services.

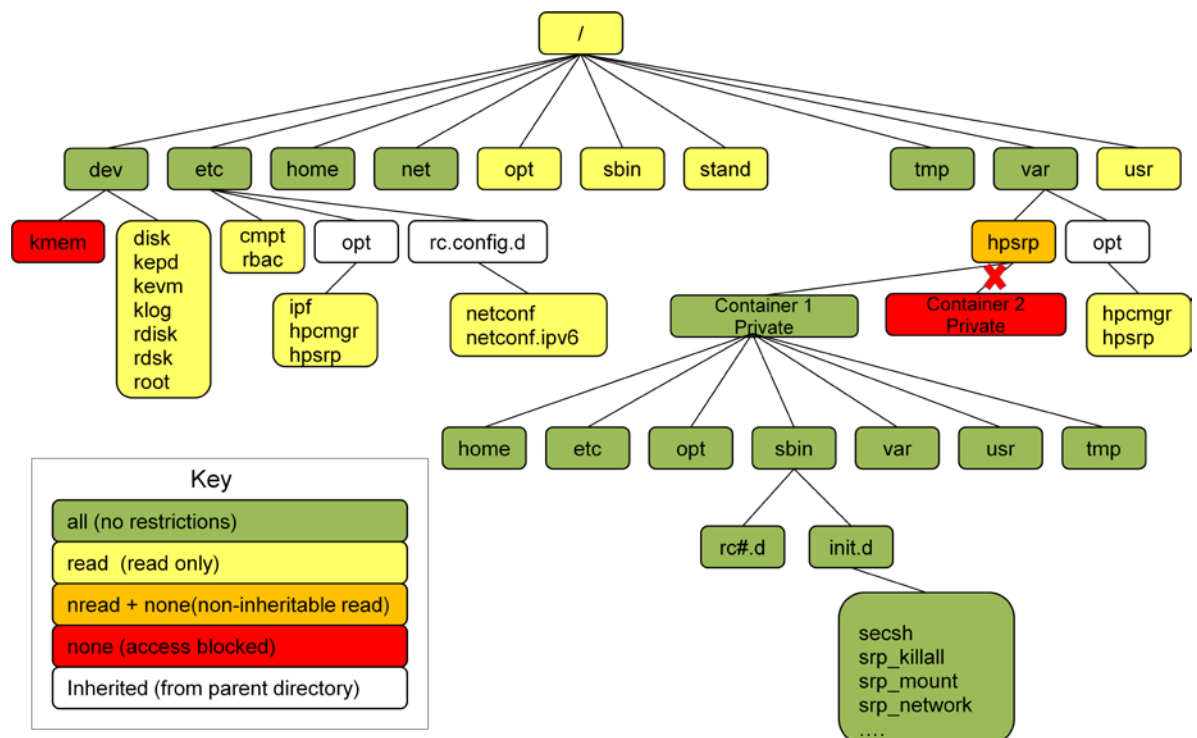
16 Workload Container

Workload containers provide access control based isolation of workload without utilizing namespace based isolation features. While not providing the user space virtualization properties of system containers, the absence of private file namespace usage allows the container to be more lightweight, and not require SD software synchronization with the global view, decreasing maintenance cost, and simplifying cloning and Serviceguard integration. By default, all administration tasks are allowed in a workload container.

16.1 File system

Workload containers have a restricted view of the entire file system, instead of the private, chroot-based view of the file system that system containers have. In addition to the container home directory, the workload container has a configurable view of other directories on the server.

The following diagram shows the file system layout for a workload container.



16.2 Users, groups and authorization

Workload containers use the HP-UX Containers login service to assign a set of HP-UX users and groups managed from the global view to log in to the container. When the compartment login feature is enabled, using the `srp_sys` command, only users in this set will be allowed to login to the container. HP recommends that you create a group for each container and apply this group to the HP-UX Containers login service.

NOTE: By default, RBAC configuration also authorizes the `root` user to log in to all containers.

16.3 Security features

HP-UX Containers provides a framework for managing container and networking security. This framework is primarily enforced with Security Containment compartment access rules. The default set of container access rules delivered with HP-UX Containers has been developed to favor functional isolation, application compatibility, and user session functionality over strong security containment. To meet the specific security requirements of your environment, you may need to replace these rules with security configurations to meet your application usage and local security policy, as described in *B.2.1 Securing containers with compartment rule Include files* (Not supported for system containers).

16.4 Devices

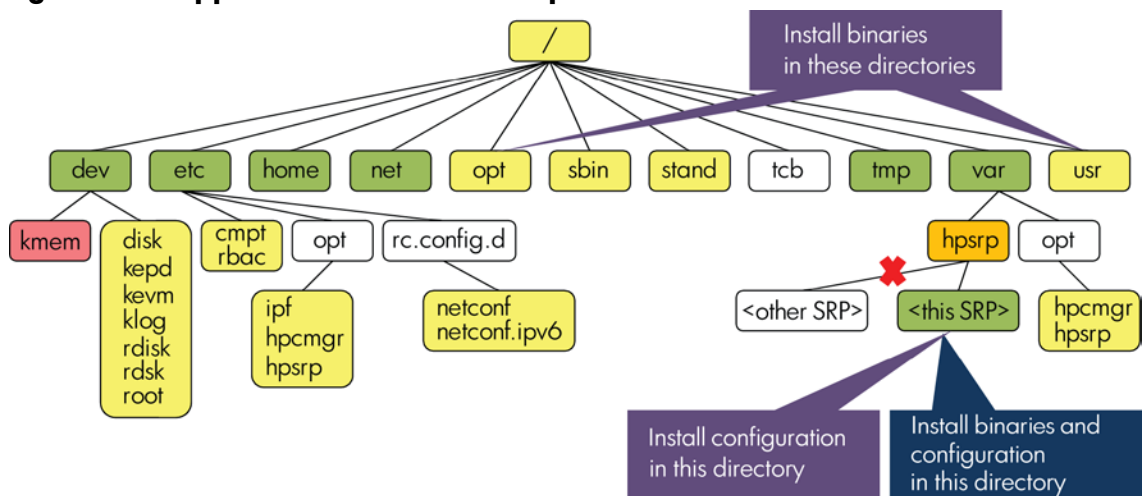
By default, all devices in the `/dev/` directory in the common global file system are available to workload containers.

16.5 Installing software

There is no synchronization required between the global view and workload containers for installation of SD based software. Therefore software installed on the system is not propagated to each workload container.

Access to software installed via SD is achieved by installing the software in a directory accessible to the container, or by modifying the container's file access rules to include the software via the custom template. For a visual representation of the file paths accessible to a workload container, see figure 16.1

Figure 16.1 Application installation map



Model	
Multi-install model	
Shared binary model	
Key	
all (no restrictions)	
read (read only)	
nread (read only - not inherited)	
none (access blocked)	
Inherited (rules from parent directory)	

16.6 Deploying applications

In order to determine how to deploy applications for use by workload containers, you must first determine if the application is supported for single or multi-instance deployment.

16.6.1 Single instance applications

While nearly all applications can be executed within a container environment, some applications do not support multiple instances of the same application executing on the same system concurrently. For these applications, HP recommends that you do not run multiple instances of these applications, even when the instances are located in separate containers, HP recommends that you install the application under the `/var/hpsrp/container_name/` directory if user-specified installation location is supported by the vendor.

If you need to execute single instance applications in more than one container, consider using system containers.

16.6.2 Multi-instance applications

There are two models to deploy multi-instance applications within a container environment:

- **Shared executable:** In this model, the application is installed once per system in a shared location. The application can require a set of files to be replicated and configured per instance. HP recommends that you locate per instance files under the `/var/hpsrp/container_name/` directory when supported by the vendor.
- **Per-container installation:** For applications that support multiple installations per system, you can install the application for each container in which it will be executed. HP recommends that you install the application under the `/var/hpsrp/container_name/` directory.

16.6.3 Deploying applications with the application templates

HP-UX Containers includes special templates for deploying key applications that use the shared executable model. The `ssh`, `apache`, and `tomcat` templates fully deploy these applications within the container using the shared executable model. The `oracledb` template configures the container for shared executable Oracle usage; however you must first install the Oracle database product on the system in the desired location. Optionally, you may also use the custom template to help deploy an Oracle database for your container. If you are installing an Oracle database under the `/var/hpsrp/container_name` directory, the `oracledb` or `custom` templates are not required.

16.6.4 Ensuring access to application files located outside the container home directory

If the application files are not all located under `/var/hpsrp/container_name/`, you must ensure that the compartment rules definition for the container includes sufficient capability to allow execution. For executable files, `READ` capability is generally sufficient, while configuration and data files will typically require `READ` and `WRITE` capability. See *15.8 System templates* and *16.8.6 Custom template* for information on using the custom template to define application specific compartment access rules for your container. In addition to any installed files, the application can also create files and directories during execution time. See *18 Verifying and troubleshooting* for instructions on using `Discover Mode` if you are unable to determine the access rules required by the application.

16.6.5 Best practices for application deployment with workload containers

Follow these best practices when deploying application with workload containers:

- **Deploy as much of the application as possible under the container home directory.**
This minimizes the need to customize compartment access rules. When the application is

installed entirely under the container home directory, customization of the container's compartment rules is usually not necessary. Life cycle management, including cloning and migration of the container will also be simplified as the application files will be managed as part of the container.

- Deploy files shared by multiple containers under the standard UNIX directories for hosting shared application files (for example, /opt/ and /usr/).**

By default, containers are configured for the `READ` capability for these directories, and will not need additional compartment rules configuration.
- If you have applied IPFilter for the container, ensure that any additional ports used by the application are allowed.**

When the `ipfilter` service is enabled for the container, by default the inbound network connections to the container are blocked. You must configure the `ipfilter` service to allow inbound connections to any network ports that the application will listen on.
- Use the custom template to apply additional capabilities to the container for the application.**

This will allow you to manage system configuration changes for the container on a per container basis. Use a recognizable identifier, such as the application name for the `instance_id` parameter when deploying the custom template. When deploying multiple applications within a container, consider applying the custom template (if needed) once per application.

16.7 Limitation and disallowed operations

By default, workload containers do not have disallowed privileges defined. You can choose to add disallowed privileges on a per container basis for workload containers by editing the Security Containment compartment definition.

16.8 Workload templates

The following table describes the templates that can be included for a workload container.

Table 16.1 Templates for workload container

Templates	Description
workload	(Required) The primary template for the workload containers. The workload template comprises of <code>admin</code> , <code>cmpt</code> , <code>login</code> , <code>init</code> , <code>prm</code> , <code>network</code> , <code>ipfilter</code> , and <code>ipsec</code> services. See <i>16.8.1 Workload template</i> .
apache	(Optional) Adds configuration data for hosting HP-UX Apache-based Web Server in a workload container. See <i>16.8.3 Apache template</i> .
tomcat	(Optional) Adds configuration data for hosting an HP-UX Tomcat servlet engine in a workload container. See <i>16.8.4 Tomcat template</i> .
sshd	(Optional) Configures and provisions an HP-UX Secure Shell daemon (<code>sshd</code>) in a workload container. See <i>16.8.2 SSHD template</i> 16.8.2 SSHD .
oracledb	(Optional) Manages configuration to enable the container to access an Oracle database installation intended to be shared by multiple containers. If you intend

Templates	Description
	to install a separate instance of the Oracle database software inside the container, you do not need to use this template. See <i>16.8.5 Oracle template</i> .
custom	(Optional) Accommodates additional application. Allows defining application specific compartment access rules, ipfilter rules and provisioning. See <i>16.8.6 Custom template</i> .

16.8.1 Workload template

The workload template includes the following services and variables:

Table 16.2 Services for the workload template

Services	Description
cmpt*	Manages configuration data for an HP-UX Security Containment compartment, which forms the core of the container.
admin*	Defines the users and groups allowed to execute the <code>srp start</code> , <code>stop</code> , and <code>status</code> operations.
login	Defines the users and groups that are allowed to login to a container. Applied only for container types that share the file system namespace and service daemons, such as the workload container.
prm	Manages the configuration of the PRM group for the container. You can specify the PRM CPU and memory allocations for the group.
network	Manages the network configuration of a container.
init*	Manages the core file directory layout and provisioning of system files for the container.
ipfilter	Configures IPFilter rules for the container that restrict inbound IP packets to the container's IP interface.
ipsec	Configures HP-UX IPsec policies for the primary network interface of the container.

(*) required services for workload container.

Table 16.3 Variables for the workload template

Service	Variable	Description
cmpt*	ok_export_dirs	Specifies whether the container directories will be saved in the exchange file for the export operation (<code>srp -export</code>) (Yes or No). Default: No
	export_copy_dirs	Comma separated list of fully qualified directory names to be copied during the import and export operations. Default: <code>var/hpsrp/container_name</code>
	allow_sw_mismatch	Allow import to proceed if software products on the source and destination systems do not match. Default: No
admin*	admin_user	Comma separated list of existing user names to be granted the role of container administrators.

Service	Variable	Description
		Default: root
login	login_group	Comma separated list of existing groups authorized to login to the container. Default: None
	login_user	Comma separated list of existing users authorized to login to the container. Default: root
init*	autostart	Auto start container at system boot (Yes or No). Default: Yes
	provision_fs	Provision the container's files if the container directory exists. Default: No
	delete_files_ok	Specify whether to delete files and directories used by the container when the container is deleted (<code>srp -delete</code>) or replaced (<code>srp -replace</code>). (Yes or No) Default: No
prm	prm_group_name	Name of the Process Resource Management (PRM) group dedicated to this container. Default: The container name.
	prm_group_type	PRM CPU allocation type (Processor Set (PSET) or Fair Share Scheduler (FSS)). Default: FSS
	prm_cores	Number of processor cores allocated (For PSET only). Default: 1
	prm_cpu_shares	Number of CPU shares allocated (For FSS only). Default: 10
	prm_cpu_max	Maximum percentage of CPU available (For FSS only). Default: No cap.
	prm_mem_shares	Specifies a max (upper bound) for memory consumption of system's memory for user processes. Default: 10
	prm_mem_max	Specifies a max (upper bound) for memory consumption of system's memory for user processes. Default: No cap
	prm_phys_mem	Memory in MB allocated for shared memory usage. Default: 0 (no dedicated physical shared memory)
network*	ip_address(+)	IP address to dedicate to the container. Default: None.
	assign_ip	Specify if IP Address has to be configured in network configuration file (<code>/etc/rc.config.d/netconf</code> or <code>netconf-ipv6</code>) . Default: Yes.
	iface(+)	Primary or secondary network interface name for the associated container IP address. For example, <code>lan0</code> . Default: None.
	ip_mask	IP subnet mask for this container (For IPv4 address only). Default: The network mask for the address class, as specified in RFC791 (IETF specification). Default: IETF standard mask is generated
	gw_ip_address	Gateway IP address for the default route for the configured container IP address. Enter the value 0 to specify no gateway IP address.

Service	Variable	Description
		Default: Same as the IP address configured for this container.
ipfilter	ipf_for_ipsec	Specify whether to allow IPFilter rules to allow IPSec packets (Yes or No). Default: No.
ipsec	ipsec_peer_addr(+)	Destination IP address for the IPSec policies. Valid Input: An IPv4 address in dotted-decimal notation or an IPv6 address in colon-hexadecimal notation. Default: None
	ipsec_transform	Transform for IPSec host policy. Valid Input: ESP_AES128_HMAC_SHA1 ESP_AES128_HMAC_MD5 ESP_3DES_HMAC_SHA1 ESP_3DES_HMAC_MD5 ESP_NULL_HMAC_SHA1 ESP_NULL_HMAC_MD5 Default: ESP_AES128_HMAC_SHA1 .
	ipsec_psk(+)	Preshared key used by IPSec Peer.Valid value: A text string, containing 1-128 ASCII characters (whitespaces are not allowed). Default: None.

(*) Required services for the workload container

(+) These variables need a value to be assigned for the adding or replacing the corresponding services.

16.8.2 SSHD template

The `sshd` template configures an HP-UX Secure Shell server daemon in the container. The following two tables describe the services and variables included with the `sshd` template.

Table 16.4 Services for the sshd template

Service	Description
<code>cmpt</code>	<p>The <code>cmpt</code> service for the <code>sshd</code> template configures Security Containment file system rules to allow the container to access the <code>sshd</code> directories in the global view specified in <code>exec_path</code> and <code>data_path</code> variables. The <code>srp</code> command add's the entries to the compartment rules file (<code>/etc/cmpt/container_name.rules</code>) that authorize access to these directories</p> <p>The <code>srp</code> command also adds an <code>include</code> statement to add the rules from the <code>/opt/hpsrp/etc/cmpt/sshd.srp_incl</code> file. As delivered by HP, this file is empty. You can edit this file to contain compartment rules to be applied when configuring the <code>cmpt</code> service with the <code>sshd</code> template.</p>
<code>ipfilter</code>	The <code>ipfilter</code> service for the <code>sshd</code> template adds rules to allow inbound requests to the specified ports used by the <code>sshd</code> server to pass. You can also specify additional inbound destination TCP port numbers for IPFilter pass rules.
<code>provision</code>	The <code>provision</code> service executes the <code>/opt/hpsrp/bin/util/secsh_setup</code> script to provision (deploy) an <code>sshd</code> in the container. By default, the tasks executed by the <code>/opt/hpsrp/bin/util/secsh_setup</code> script include:

Service	Description
---------	-------------

- Uses the SSH `ssh-keygen` utility to generate an RSA key pair to use for the `sshd` host key pair. These keys are stored in the container-specific `sshd` data path directory (`/var/hpsrp/container_name/opt/ssh`) with the following names:
 - `ssh_host_rsa_key` (RSA private key)
 - `ssh_host_rsa_key.pub` (RSA public key)
- Creates a container-specific copy of the `sshd` configuration file by copying the `sshd_config` file from the specified `data_src` directory to the `data_path` directory and modifying it with container-specific data, including setting the `HostKey` parameter to `/var/hpsrp/container_name/opt/ssh/ssh_host_rsa_key`.
- Creates container-specific initialization scripts and startup file to start the `sshd` with the container-specific `sshd_config` file when the container startup script is executed. The setup script:
 - Creates the container-specific startup configuration file, `/var/hpsrp/container_name/etc/rc.config.d/sshd`, which specifies the container-specific `sshd` configuration file as a startup argument for `sshd`.
 - Adds the startup and shutdown script `secsh` to the container-specific `init.d` directory, `/var/hpsrp/container_name/sbin/init.d`. This file is linked to the `/var/hpsrp/container_name/sbin/rc2.d/S393secsh` and `/var/hpsrp/container_name/sbin/rc1.d/K393sech` files.

Note on Completing the Configuration

Tasks you might need to perform to complete the configuration include the following:

- Editing the container `sshd_config` file (the default location is `/var/hpsrp/container_name/opt/ssh/sshd_config`).

If a client has the `StrictHostKeyChecking` directive set to `yes`, you must add the host public key file (`ssh_host_dsa_key.pub` or `ssh_host_rsa_key.pub`) to the client configuration, as described in the HP-UX Secure Shell documentation.

Table 16.5 Variables for the `sshd` template

Service	Variable	Variable Description
cmpt	<code>data_path</code>	Specifies the container-specific target directory for <code>sshd</code> configuration and key files. Default: <code>/var/hpsrp/container_name/opt/ssh</code> .
	<code>exec_path</code>	The location of the executables for the HP-UX Secure Shell product. Default: <code>/opt/ssh</code> .

Service	Variable	Variable Description
Ipfilter	ipf_tcp_ports	Specifies the local TCP port numbers for IPFilter rules that allow inbound packets. Variable Name: ipf_tcp_ports. Valid Input: One or more TCP port numbers each in the range 1-65535, separated by commas. Default: 22. This is the IANA registered port number for SSH remote login.
Provision	data_path	Same as described in cmpt service.
	exec_path	Same as described in cmpt service.
	data_src	Specifies the directory from which you want to copy SSH configuration data. In most cases, this should be the newconfig directory shipped with the HP-UX Secure Shell product. Default: /opt/ssh/newconfig.
	sshd_port	Specifies the TCP port number on which the container sshd will receive connection requests. Variable Name: sshd_port. Valid Input: A TCP port number in the range 1-65535. Default: 22, the IANA registered port number for SSH login.
	script_name	Specifies the provision script to be used to configure sshd server in the container. Variable Name: script_name Default: /opt/hpsrp/bin/util/secsh_setup

16.8.3 Apache template

The apache template configures an HP-UX Apache web server in the container. The following two tables describe the services and variables included with the apache template.

Table 16.6 Services for the apache template

Service	Description
cmpt	The cmpt service for the apache template configures Security Containment file system rules to allow the container to access the specified Apache directories in global view. The srp command adds the entries to the compartment rules file (/etc/cmpt/container_name.rules) that authorizes access to the directories specified in the exec_path and data_path variables.
ipfilter	The ipfilter service for the apache template adds rules to allow inbound requests to the specified ports used by the Apache server to pass. You can also specify additional inbound destination TCP port numbers for IPFilter pass rules. The srp command inserts these rules at the top of the IPFilter rules file and uses the quick keyword. The IPFilter configuration file already contains rules from the base template to allow all outbound TCP, UDP, and ICMP packets from the container IP address.
provision	The provision service executes the script /opt/hpsrp/bin/util/apache_setup to provision (deploy) an apache service in the container. By default, the tasks executed by the /opt/hpsrp/bin/util/apache_setup script include: <ul style="list-style-type: none"> • Creating bin, cgi-bin, conf, htdocs, and logs subdirectories below the container Apache home directory.

Service	Description
	<ul style="list-style-type: none"> • Creating a container-specific <code>http.conf</code> file with container-specific configuration data, such as setting data paths to the appropriate directories below the container Apache home directory and setting the IP address to the container IP address. Enables the <code>mod_ajp</code> module for Apache Tomcat integration. • Creating container-specific initialization scripts and startup file to start Apache with the container-specific <code>http.conf</code> file when the container startup script is executed. The setup script: <ul style="list-style-type: none"> ○ Modifies the container-specific <code>apachectl</code> file in the <code>bin</code> subdirectory below the <code>data_path</code> (by default, the path is <code>/var/hpsrp/container_name/opt/hpws22/apache/bin/apachectl</code>) to use the container-specific Apache data path directory as the <code>ServerRoot</code>. ○ Creates the container-specific startup configuration file, <code>/var/hpsrp/container_name/etc/rc.config.d/hpws22_apacheconf</code>, which specifies the container-specific Apache home directory. ○ Adds the startup and shutdown script <code>hpws22_apache</code> to the container-specific <code>init.d</code> directory, <code>/var/hpsrp/container_name/sbin/init.d</code>. This file is linked to the <code>/var/hpsrp/container_name/sbin/rc3.d/S823hpws22_apache</code> and <code>/var/hpsrp/container_name/sbin/rc3.d/K177hpws22_apache</code> files.

Note on Completing the Configuration

After you apply the `apache cmpt` service and the default `apache` provisioning script, you can start the container, and have a fully-functional HP-UX Apache-based Web Server in the container. You can further customize the Web Server as needed by editing the container-specific Apache configuration files (`/var/hpsrp/container_name/etc/rc.config.d/hpws22_apacheconf` and the container-specific `apachectl` file, located in the `bin` subdirectory below the `data_path`).

Table 16.7 Variables for the apache template

Service	Variable	Description
cmpt	wss_version	The HP-UX Web Server Suite version of Apache to be used to configure the template Default: 3.0.
	data_path	The root directory for Apache data. The <code>cmpt</code> service adds rules to allow the container all access to this directory. Users and processes in the container can read, write, traverse (<code>nsearch</code>), and delete (<code>ulink</code>) the contents of these directories. Default: <code>/var/hpsrp/container_name/opt/hpws22/apache</code> .
	exec_path	The root directory for Apache executables. The <code>cmpt</code> service adds rules to allow the container read access to this directory. Default: <code>/opt/hpws22/apache</code> .
ipfilter	http_port	Specifies the TCP port number on which the container Apache server will receive HTTP requests.

Service	Variable	Description
		Valid Input: A TCP port number in the range 1-65535. Default: 80, the IANA registered port number for HTTP.
	https_port	Specifies the TCP port number on which the container Apache server will receive HTTPS (SSL) requests. Valid Input: A TCP port number in the range 1-65535. Default: 443, the IANA registered port number for HTTPS.
	ajp_port	Specifies the TCP port number on which the container Apache server will send requests to a Tomcat server. Variable Name: ajp_port. Valid Input: A TCP port number in the range 1-65535. Default: 8009.
	ipf_tcp_ports	Specifies the local TCP port numbers for IPFilter rules that allow inbound packets. Valid Input: One or more TCP port numbers each in the range 1-65535, separated by commas. Default: 80, 443. These are the IANA registered port numbers for HTTP and HTTPS (SSL).
provision	wss_version,	Same as described for the <code>cmpt</code> services.
	data_path	Same as described for the <code>cmpt</code> services.
	http_port	Same as described for the <code>ipfilter</code> services.
	https_port	Same as described for the <code>ipfilter</code> services.
	ajp_port	Same as described for the <code>ipfilter</code> services.
	data_src	The directory from which you want to copy Apache data. The <code>provision</code> service creates a copy of this subtree and its contents and installs it in the specified <code>data_path</code> for use by the container. The input for this variable is typically the <code>newconfig</code> subdirectory under the Apache product directory. Default: <code>/opt/hpws22/apache/newconfig</code> .
	user	Specifies the UNIX user name under which the Apache processes in this container will run. Default: <code>www</code> .
	start_apache	Specifies if you want the <code>srp</code> command to add a script to the container <code>init</code> directory structure to start and stop Apache. The script is automatically executed when the container is started or stopped. Valid Input: <code>yes</code> or <code>no</code> . Default: <code>yes</code> .
	startssl_apache	Specifies if you want to start Apache in <code>ssl</code> mode. Variable Name: <code>startssl_apache</code> Valid Input: <code>yes</code> or <code>no</code> Default: <code>no</code>

16.8.4 Tomcat template

You can use the tomcat template to add configuration data for hosting an HP-UX Tomcat servlet engine in a workload container. The following two tables describe the services and variables included in Tomcat template.

Table 16.8 Services for the tomcat template

Service	Description
<code>cmpt</code>	The <code>cmpt</code> service for the <code>apache</code> template configures Security Containment file

Service	Description
	<p>system rules to allow the container to access the specified Apache directories in global view. The <code>srp</code> command adds entries to the container rules file (<code>/etc/cmpt/container_name.rules</code>) that authorizes access to the directories specified in <code>exec_path</code>, <code>data_path</code>, and <code>java_path</code> variables. The <code>srp</code> command also adds an <code>include</code> statement to add the rules from the <code>/opt/hpsrp/etc/cmpt/tomcat.srp_incl</code> file. As delivered by HP, this file is empty. You can edit this file to contain compartment rules to be applied when configuring the <code>cmpt</code> service with the <code>tomcat</code> template.</p>
<p><code>ipfilter</code></p>	<p>The <code>ipfilter</code> service for the <code>tomcat</code> template adds rules to allow inbound requests to the specified ports used by the Tomcat server to pass. You can also specify additional inbound destination TCP port numbers for IPFilter pass rules</p>
<p><code>provision</code></p>	<p>The <code>provision</code> service executes the script <code>/opt/hpsrp/bin/util/tomcat_setup</code> to provision (deploy) a <code>tomcat</code> service in the container.</p> <p>By default, the tasks executed by the <code>/opt/hpsrp/bin/util/tomcat_setup</code> script include:</p> <ul style="list-style-type: none"> • Creating <code>conf</code>, <code>logs</code>, <code>temp</code>, <code>webapps</code>, and <code>work</code> subdirectories below the container Tomcat home directory. • Creating a container-specific <code>server.xml</code> file with container-specific configuration data, such as setting tomcat http, ajp ports. • Creating container-specific initialization scripts and startup file to start Tomcat with the container-specific configuration files when the container startup script is executed. The setup script: <ul style="list-style-type: none"> ○ Modifies the initialization scripts to start/stop Tomcat application as the Tomcat user. Also, exported variables that define Tomcat's <code>CATALINA_HOME</code>, <code>CATALINA_BASE</code> and <code>JAVA_HOME</code> directory. ○ Creates the container-specific startup configuration file, <code>/var/hpsrp/container_name/etc/rc.config.d/hpws22_tomcatconf</code>, which specifies the container-specific tomcat home directory. ○ Adds the startup and shutdown script <code>hpws22_tomcat</code> to the container-specific <code>init.d</code> directory, <code>/var/hpsrp/container_name/sbin/init.d</code>. This file is linked to the <code>/var/hpsrp/container_name/sbin/rc3.d/S823hpws22_tomcat</code> and <code>/var/hpsrp/container_name/sbin/rc3.d/K177hpws22_tomcat</code> files.
	<p><i>Completing the Configuration</i></p> <p>After you apply the <code>tomcat cmpt</code> service and the default <code>tomcat</code> provisioning script, you can start the container, and have a fully-functional HP-UX Tomcat-based servlet engine in the container. You can further customize the servlet engine as needed by editing the container-specific Tomcat configuration files (<code>/var/hpsrp/container_name/etc/rc.config.d/hpws22_tomcatconf</code> and the container-specific <code>server.xml</code>, located in the <code>conf</code> subdirectory below the <code>data_path</code>).</p>

Table 16.9 Variables for the tomcat template

Service	Variable	Variable Description
cmpt	wss_version	The HP-UX Webserver Suite version of Tomcat Servlet Engine to be used to configure the template Default: 3.0.
	exec_path	The root directory for Tomcat executables. Default: /opt/hpws22/tomcat.
	data_src	The directory from which you want to copy Tomcat data. The provision service creates a copy of this subtree and its contents and installs it in the specified data_path for use by the container. The input for this variable is typically the newconfig subdirectory under the Tomcat product directory. Default: /opt/hpws22/tomcat/newconfig.
	data_path	The target directory for the copied Tomcat data. Default: /var/hpsrp/container_name/opt/hpws/tomcat.
	java_path	The Java home path. Default: /opt/java1.5
ipfilter	http_port	Specifies the TCP port number on which the container Tomcat server will receive HTTP requests. Valid Input: A TCP port number in the range 1-65535. Default: 8081.
	control_port	Specifies the TCP port number on which the container Tomcat server will receive request from apache webserver. Valid Input: A TCP port number in the range 1-65535. Default: 8005.
	ajp_port	Specifies the TCP port number on which the container Tomcat server will receive request from an Apache webserver. Valid Input: A TCP port number in the range 1-65535. Default: 8009.
	ipf_tcp_ports	Specifies the local TCP port numbers for IPFilter rules that allow inbound packets. Valid Input: One or more TCP port numbers each in the range 1-65535, separated by commas. Default: 8085,8081,8009.
provision	wss_version	Same as described for the cmpt service.
	exec_path	Same as described for the cmpt service.
	data_src	Same as described for the cmpt service.
	data_path	Same as described for the cmpt service.
	java_path	Same as described for the cmpt service.
	user	Specifies the user name under which the Tomcat processes in this container will run. Default: www.
	start_tomcat	Specifies if you want to add a script to the container startup/shutdown (init) directory structure to start and stop Tomcat. The script is automatically executed when the container is started or stopped. Variable Name: start_tomcat. Valid Input: yes or no. Default: yes.

16.8.5 Oracle template

The `oracledb` template allows you to configure a container to share a single set of Oracle executables with other containers. You do not need to use this template if you are installing a separate instance of the Oracle executables in the container.

Table 16.10 Services for the oracle template

Service	Description
<code>cmpt</code>	The <code>cmpt</code> service for the <code>oracledb</code> template configures Security Containment file system rules to allow the container to access the specified Oracle directories. The <code>srp</code> command adds entries to the container's compartment rules file (<code>/etc/cmpt/container_name.rules</code>) that authorizes access to the (global view) directories specified in the <code>exec_path</code> and <code>data_path</code> variables. The <code>srp</code> command also adds an <code>include</code> statement to add the rules from the <code>/opt/hpsrp/etc/cmpt/oracledb.srp_incl</code> file. As delivered by HP, this file is empty. You can edit this file to contain compartment rules to be applied when configuring the <code>cmpt</code> service with the <code>oracledb</code> template.
<code>ipfilter</code>	The <code>ipfilter</code> service for the <code>oracledb</code> template adds rules to allow inbound requests to the specified ports used by the Oracle database server to pass. You can also specify additional inbound destination TCP port numbers for IPFilter pass rules.
<code>provision</code>	The <code>provision</code> service executes the script provided to provision (deploy) an admin, login, and network service in the container. After applying the <code>cmpt</code> service and optionally the <code>ipfilter</code> service for the <code>oracledb</code> template, you can deploy an Oracle Database Server in the container. You may need to make a copy or link from the Oracle product installation directory to the <code>exec_path</code> configured for the <code>cmpt</code> service. You might also need to set up the Oracle configuration and schema under the <code>data_path</code> configured for the <code>cmpt</code> service. The <code>oracledb</code> template provides <code>provision</code> service optionally. You must write your <code>provision</code> script if you need one for any of your customized operations; such as to change the permissions or copy some files. You can also copy or create a startup and shutdown script for the Oracle processes, and install or link it to files in a <code>/var/hpsrp/container_name/sbin/rcN.d</code> directory.

Table 16.11 Variables for the oracle template

Service	Variable	Description
<code>cmpt</code>	<code>exec_path</code>	The root directory for Oracle executables. The <code>cmpt</code> service adds rules to allow the container read access to this directory. Because this parameter is configured per container, you can select different versions of the Oracle Database server product on the system. Default: <code>/var/hpsrp/container_name/opt/u01/home/oracle</code>
	<code>data_path</code>	The root directory for Oracle data. The <code>cmpt</code> service adds rules to allow the container all access to this directory. Users and processes in the container can read, write, traverse (<code>nsearch</code>), and delete (<code>ulink</code>) the contents of these directories. In most cases, you would set up the Oracle configuration and schema under this path, and set the value of the <code>ORACLE_HOME</code> environment variable to this path. Default: <code>/var/hpsrp/container_name/opt/u01/home/oracle</code> .

Service	Variable	Description
ipfilter	ipf_tcp_ports	Specifies the local TCP port numbers for IPFilter rules that allow inbound packets. Variable Name: ipf_tcp_ports. Valid Input: One or more TCP port numbers each in the range 1-65535, separated by commas. Default: 1521. This is the default port number for the Oracle Net Listener process (commonly referred to as the listener).
provision	exec_path	Same as described in cmpt service.
	data_path	Same as described in cmpt service.

16.8.6 Custom template

The `custom` template enables you to specify additional Security Containment file access rules and IPFilter rules for a container. You can use the `custom` template to accommodate additional applications in a container, or to add compartment or IPFilter rules to increase security controls for a container.

Table 16.12 Services for the custom template

Service	Description
cmpt	<p>The <code>cmpt</code> service for the <code>custom</code> template applies additional compartment rules to your container. You can specify a rules file to include and specify file system paths to configure for different access types.</p> <p>The <code>srp</code> (<code>-add</code>) command adds entries to the rules file for the container to authorize access according to the descriptions in the previous sections. The <code>srp</code> command also adds an <code>include</code> statement to add the rules from the files specified by <code>cmpt_rule_file</code>.</p>
ipfilter	The <code>ipfilter</code> service for the <code>custom</code> template enables you to allow inbound packets to specific TCP or UDP port numbers.
provision	The <code>provision</code> service executes the customizable script <code>/opt/hpsrp/bin/util/custom_setup</code> to provision (deploy) an additional application in the container. Allows users to write their own functionality for each of the operations such as add, delete, and replace.

Table 16.13 Variables for the custom template

Service	Variable	Description
cmpt	cmpt_rule_file	Specifies compartment rule files to include in the compartment rules file for this container. To specify multiple files, use commas to separate file names. Default: None.
	read_access	Specifies directories to configure with read access (<code>nsearch</code> and <code>read</code>) in the compartment rules file for this container. To specify multiple directories, use commas to separate directory names. Default: None

	<code>all_access</code>	Specifies directories to configure with <code>all</code> access in the compartment rules file for this container. To specify multiple directories, use commas to separate directory names. Default: None.
	<code>no_access</code>	Specifies directories to configure with <code>none</code> access in the compartment rules file for this container. This will disallow access to the specified directories unless an additional access rule has been specified for this path from this container. To specify multiple directories, use commas to separate directory names. Default: None
<code>ipfilter</code>	<code>ipf_tcp_ports</code>	Specifies the local TCP port numbers for IPFilter rules that allow inbound packets. Valid Input: One or more TCP port numbers each in the range 1-65535, separated by commas. Default: None.
	<code>ipf_udp_ports</code>	Specifies the local UDP port numbers for IPFilter rules that allow inbound packets. Valid Input: One or more UDP port numbers each in the range 1-65535, separated by commas. Default: None.
<code>provision</code>	<code>script</code>	The provision script path to use to configure additional set of applications. Allows users to write their own functionality for each of the operations like add/delete/replace. Default: <code>/opt/hpsrp/bin/util/custom_setup</code> .

17 Compatibility with other HP-UX products

17.1 Bastille revert feature

Bastille provides the ability to save and restore a baseline configuration of an HP-UX system. If you use the `bastille -r` command to revert to the Bastille baseline configuration, you can lose any IPFilter rules configured using HP-UX Containers that are not in the baseline configuration. HP recommends that you do not configure the IPFilter service with HP-UX Containers if you are using Bastille to manage IPFilter rules. If Bastille is managing IPFilter rules, the `/etc/opt/ipf/ipf.conf` or `/etc/opt/ipf/ipf.conf` file contains a statement similar to the following:

```
# WARNING: This file was generated automatically and will be replaced
# the next time you run Bastille. DO NOT EDIT IT DIRECTLY!!!
```

17.2 PRM HP-UX Containers commands

The HP PRM product includes the following commands to associate a Security Containment compartment with a PRM group:

- `prm2scomp`
- `scomp2prm`
- `srpgen`

HP requires that you use the `srp` command, instead of the PRM commands listed above, to create or modify the Security Containment compartment and PRM group configuration associated with an HP-UX container. Security Containment compartments created with the above commands will not work as HP-UX Containers but may coexist with HP-UX Containers on the same system.

17.3 Configuration Synchronization Manager (CMGR) utility and libraries

The Configuration Synchronization Manager (CMGR) product is included in the HP-UX-SRP bundle. The CMGR product includes the `cmgr` utility and libraries, which enables the `srp` command to coordinate the configuration of multiple subsystems. The `srp` command invokes the `cmgr` utility.

For more information about CMGR, refer to the *HP-UX CMGR Administrator's and Developer's Guide*.

18 Verifying and troubleshooting containers

This chapter contains procedures for verifying and troubleshooting containers. This chapter addresses the following topics:

NOTE: You can run system administration and performance tools (such as `glance`, `gpm`, `kprof`, `kgmon`, `ktrace`, and `caliper`) in the global view.

18.1 Quick verification procedure

You can use the `srp_sys` command with the `-l` option (or `-l -v` for more verbose information) to quickly verify the status of the subsystem data configured by HP-UX Containers.

The optional subsystems that you can configure are Compartment Login, PRM, IPFilter and IPsec. You must configure all the other subsystems for the HP-UX Containers environment to function correctly.

If a subsystem is not enabled, use the `srp_sys -enable [subsystem]` command to enable the subsystem. Refer to `srp_sys(1m)` for more information.

18.2 Troubleshooting scenarios

- **Scenario 1: A non-root user is unable to login to the global view of the HP-UX Containers enabled system.**

Symptom: `telnet` or `rlogin` fails with the following error:

```
Compartment access check failed: User is not authorized to login to
the compartment associated with this network service.
Connection to host lost.
```

Solution: Only users in the group `srpgrp` are authorized to login to the system. Add the user to the group `srpgrp`.

- **Scenario 2: Installing a product update fails.**

Symptom: The `swinstall` command fails with the error:

```
ERROR: Cannot continue "swinstall". The shared srp's must be in the
stopped state. container_name is in the started state.
```

Solution: Change the state of the container to `stopped` using the `srp -stop container_name` command.

- **Scenario 3: Installing a product update from a remote source fails.**

Symptom: `swinstall` fails with the error:

```
ERROR: The source depot specified using a host target selection
(host:/path). Installing from a remote source is not supported
in SRP environment. To install from a remote source, either
mount it locally or copy the software locally using swcopy.
```

Solution: Installing a software update from a remote source is not supported in the HP-UX Containers environment, the software must be available locally. To make the source depot available locally, do the following:

- Use the `swcopy` command to copy the depot to the local system (see `swcopy(1M)`).
- If the software is in a media, mount the depot locally.

- o Use NFS to mount the depot from the remote server to the local filesystem.

Once the software depot is available locally, run the `swinstall` command to point to the local source.

- **Scenario 4: The GUI version of the swinstall command does not work in the HP-UX Containers environment.**

Symptom: The `swinstall` command invoked with no command line options fails with the following error message:

```
# swinstall
ERROR: The interactive UI is not supported in SRP environment.
```

Solution: The GUI version of `swinstall` is not supported, instead use the command line interface in the HP-UX Containers environment.

- **Scenario 5: Container fails to start.**

Symptom: The `srp -start container_name` gives the following error:

```
# srp -start container_name
SRP container_name not started:
The SRP must be (re)synchronized with the system's installed product
database.
Run /opt/hpsrp/bin/util/srp_check to identify the list of products to
install or remove from this SRP.
```

Solution:

1. Run the `srp_check` command and identify the products that are out of sync with the global (see `srp_check(1M)`).
2. Check the `/var/adm/sw/swagent.log` file in the container to identify the problem. To login to the container, first change its state to maintenance using the command `srp -maint container_name` and then use `-M` option with the `srp_su` command as `srp_su -M container_name`.
3. Take corrective action (if any) based on the information in the `swagent.log` file.
4. Change state of the container back to stopped.
5. Install the patch targeting the container as:
`swinstall -x local_srp_list=container_name \`
`-s <depot location> Product name`

- **Scenario 6: Unable to telnet or rlogin to a container**

Symptom: Remote login to a container fails with one of the following messages:

```
# telnet container_name
Trying...
telnet: Unable to connect to remote host: Connection refused

# rlogin container_name
rcmd_af: connect: container_name: Connection refused
```

Solution: The container must be in `started` state to accept login requests. If the container is of type `workload`, then you can login to the container using `ssh` only. To verify if the container is of type `workload`, run the `srp -status` in the system where the container resides and check the second field `TYPE`.

- **Scenario 7: Process respawn does not work in the container.**

Symptom: Processes configured for respawn in the container's `/etc/inittab` file does not respawn.

Solution: Verify and confirm that the `srp_init` daemon is up and running inside the container by executing the following command in the container:

```
# ps -ef | grep srp_init
```

If the `srp_init` daemon is running, enter the following command to re-examine the `/etc/inittab` file entries without changing the run level:

```
# /sbin/srp_init q
```

If the `srp_init` daemon is not running, restart `srp_init` within the container using the `/sbin/srp_init` daemon.

18.3 Advanced verification procedures

This section includes advanced verification procedures to verify the subsystem data configured by HP-UX Containers.

18.3.1 Verifying Security Containment compartment data

Use the following procedures to verify Security Containment compartment configuration data:

- Verify that the compartment rules are loaded into the kernel.

Enter the following command:

```
# getrules -m container_name
```

- Manually test the file access rules. This verification procedure applies to workload containers only.

Login to the container and attempt file access operations that should succeed or fail, such as `cd` and `touch` commands for files not available from the container. From the global view, you can create a temporary file in a directory for which the container does not have `uLink` (delete) access. Login to the container and attempt to delete the file.

- Verify that the processes configured for the container are running in the compartment. This verification procedure applies to workload containers only.

Use the `ps -ef` command to find the PID for applications in your container. For example:

```
# ps -ef | grep sshd
root  968      1 0 Oct 14   ?           0:00 /usr/sbin/sshd
```

Use the `getprocxsec -c pid` command to get the compartment in which the process is running. For example:

```
# getprocxsec -c 968
cmpt= SRP2
```

- For workload containers, if an application is failing in a compartment and you want to determine if it is failing because of Security Containment rules, you can use the HP-UX audit utility to configure and view audit to see if operations are failing because of permission problems.

One method to reduce the number of unrelated audit entries is to disable auditing for all users, then enable auditing for the user ID used to execute the application. Next, configure auditing for failed attempts for common file and IPC operations.

For example:

```
audevent -F -e open -e create -e delete -e ipccreat -e ipcopen \
-e ipcclose -s kill
```

18.3.2 Verifying RBAC data

Use the following procedures to verify RBAC configuration data:

- Use the `authadm` command to verify the authorization information configured for the container:

```
authadm list object=container_name
```

For the `admin` service, you will see the following entry:

```
SRPadmin-container_name: (hpux.SRPadmin.container_name,container_name)
```

For the `login` service, you will see the following entry:

```
SRPlogin-container_name: (hpux.security.compartment.login, container_name)
```

Alternatively, you can enter the following commands to view the authorization information:

```
authadm list operation=hpux.SRPadmin.container_name
authadm list operation=hpux.security.compartment.login \
object=container_name
```

- To verify the users and user groups assigned to the roles used by the container, enter the following commands:

```
roleadm list role=SRPadmin-container_name
roleadm list role=SRPlogin-container_name
```

- To verify command privileges, view the `/etc/rbac/cmd_priv` file. If you configured the `init` service for a container, you will see an entry authorizing execution of the `srp_rc` script for an authorization granted to the container administrator as follows:

```
/opt/hpsrp/bin/util/srp_rc:dflt:(hpux.SRPadmin.container_name,*):0/0//:container_name:dflt:dflt
```

- You can also use the `rbacdbchk` utility to verify the contents of the RBAC database.

18.3.3 Verifying PRM data

Use the `prmlist` and `prmmmonitor` commands to verify that the PRM configuration is loaded for the group used by the container (the default PRM group name is the container name).

For example, the `prmlist -g -s` command displays configuration information for PRM groups (`-g`) and the PRM group for each Security Containment compartment (`-s`):

```
# prmlist -g -s
```

```
PRM configured from file: /etc/prmconf
File last modified:      Tue Oct 14 12:57:58 2008
```

PRM Group	PRMID	CPU Entitlement	CPU Max	LCPU Attr
EntDir	2	29.17%	80%	
MktDB	65536	12.50%		
MktWeb	3	21.88%	45%	
OTHERS	1	21.88%		
SRP2	4	14.58%	25%	

Compartment	Default PRM Group
EntDir	EntDir
MktDB	MktDB
MktWeb	MktWeb
SRP2	SRP2

The `prmmmonitor` utility displays statistics for each PRM group.

```
# prmmmonitor
```

```
PRM configured from file: /etc/prmconf
File last modified: Tue Oct 14 12:57:58 2008
```

```
HP-UX habs B.11.31 U ia64 10/14/08
```

```
Tue Oct 14 13:03:11 2008 Sample: 1 second
CPU scheduler state: Enabled
```

PRM Group	PRMID	CPU Entitle	CPU Max	CPU Used	LCPU State
OTHERS	1	21.88%		3.06%	
EntDir	2	29.17%	80%	24.10%	
MktWeb	3	21.88%	45%	12.36%	
SRP2	4	14.58%	25%	22.88%	
MktDB	65536	12.50%		12.46%	

```
PRM application manager state: Enabled (polling interval: 30 seconds)
```

18.3.4 Verifying network data

Use the `netstat -in` and `netstat -rn` commands to verify the container interface and route entries. If the commands are executed inside a container, they list the IP interfaces, and route configuration for the container only. If the commands are executed in the global view, the command lists all the IP interfaces and routes configured on the system. An asterisk next to the interface name indicates that the interface is configured, but its state is down. In the following example, the state for `lan1`, `lan1:1` and `lo0` is up, but the state for `lan1:2` is down.

```
# netstat -in
Name      Mtu  Network      Address      Ipkts      Ierrs Opkts      Oerrs Coll
lan1      1500 10.0.0.0     10.0.0.1    460732     0      279522     0      0
lan1:1    1500 192.0.2.0    19.0.2.1    32890      0      51537      0      0
lan1:2*   1500 192.0.2.0    19.0.2.2    0           0       0           0      0
lo0       32808 127.0.0.0    127.0.0.1   890170     0      890178     0      0
```

If a container is up and has a dedicated IP interface, the `netstat -rn` command shows a default route entry with the container IP address (192.0.2.1) as the gateway. For example:

```
# netstat -rn
Routing tables
Destination      Gateway          Flags Refs Interface  Pmtu
:
:
default          192.0.2.1       U      0      lan1:1     1500
```

To identify the IP addresses associated with each container from the global view, use the `srp -status -v` command.

18.3.5 Verifying IPFilter data

Use the following `ipfstat` command to view the active (loaded) inbound and outbound IPFilter rules:

```
ipfstat -io
```

For example:
ipfstat -io

```
pass out quick proto tcp from 192.0.2.1/32 to any keep state
pass out quick proto udp from 192.0.2.1/32 to any keep state
pass out quick proto icmp from 192.0.2.1/32 to any keep state
pass in quick proto icmp from any to 192.0.2.1/32
block in quick from any to 192.0.2.1/32
```

18.3.6 Verifying IPsec data

Enter the following IPsec commands to verify IPsec data:

- Use the following `ipsec_report` command to view the host rules:
`ipsec_report -host`

The output should include a host policy with the name `SRP-container_name-base-1`

For example:

```
----- Configured Host Policy Rule -----
Rule Name: SRP-web2-base-1      ID: 7      Priority: 30
Src IP Addr: 192.0.2.1  Prefix: 32  Port number: 0
Dst IP Addr: 10.2.2.2  Prefix: 32  Port number: 0
Network Protocol: All      Action: Dynamic key SA
Number of SA(s) Needed: 1 Pair(s)
Proposal 1: Transform: ESP-AES128-HMAC-SHA1
              Lifetime Seconds: 28800
              Lifetime Kbytes: 0
```

- Use the following `ipsec_report` command to view the IKE rules:
`ipsec_report -ike`

The output should include an IKE policy with the name `SRP-container_name-base-1`. For example:

```
----- IKE Rule -----
Rule Name: SRP-web2-base-1      Priority: 30      Cookie: 6
Remote IP Address: 10.2.2.2  Prefix: 32
Group Type: 2      Authentication Method: Pre-shared Keys
Authentication Algorithm: HMAC-MD5      Encryption Algorithm: 3DES-CBC
Number of Quick Modes: 100      Lifetime (seconds): 28800
Action: Secure
```

- Use the following `ipsec_config` command to view the authentication records:
`ipsec_config show auth`

The output should include an IKE policy with the name `SRP-container_name-base-1`. For example:

```
auth SRP-web2-base-1
-remote 10.2.2.2/32
-preshared myPresharedKey
-exchange MM
```

- You can also use the `ipsec_policy` utility to verify the IPsec host rule selected for a packet from the peer address. In the following example, the container address is `19.2.0.2.1` and the peer address is `10.2.2.2`. The `ipsec_policy` command queries IPsec to determine which IPsec and IKE policies are selected for an outbound packet (`-dir out`) with source IP address (`-sa`) `192.0.2.1` and destination IP address (`-da`) `10.2.2.2`.

```
# ipsec_policy -sa 192.0.2.1 -da 10.2.2.2 -dir out
```

```
----- Active Host Policy Rule -----
```

```

Rule Name: SRP-web2-base-1      ID: 8      Cookie: 3      Priority: 30
Src IP Addr: 192.0.2.1  Prefix: 32  Port number: 0
Dst IP Addr: 10.2.2.2  Prefix: 32  Port number: 0
Network Protocol: All      Direction: outbound
Action: Dynamic key SA      State: SPI(s) Not Established
Number of SA(s) Needed: 1 Pair(s)
Number of SA(s) Created: 0 Pair(s)
Kernel Requests Queued: 0
Proposal 1: Transform: ESP-AES128-HMAC-SHA1
              Lifetime Seconds: 28800
              Lifetime Kbytes: 0

```

```

----- IKE Rule -----
Rule Name: SRP-web2-base-1      Priority: 20      Cookie: 4
Remote IP Address: 10.2.2.2  Prefix: 32
Group Type: 2      Authentication Method: Pre-shared Keys
Authentication Algorithm: HMAC-MD5      Encryption Algorithm: 3DES-CBC
Number of Quick Modes: 100      Lifetime (seconds): 28800
Action: Secure

```

18.4 Advanced Troubleshooting Procedures

This section includes advanced troubleshooting procedures:

18.4.1 Using the Security Containment compartment discover feature (workload containers only)

In a secure environment, you can use the Security Containment discover feature to remove compartment restrictions and view the rules that are needed to allow access. (If you are not in a secure environment, you can use IPFilter to allow access from only trusted systems before removing compartment restrictions.)

You can use the discover feature as follows:

1. Stop the container:
`srp -stop container_name`
2. Edit the compartment rules file (`/etc/cmpt/container_name`), and tag the container definition at the beginning of the file with the `discover` keyword. This opens the container for all access. For example:

```
discover compartment myContainer {
:
:
}
```
3. Start the container:
`srp -start container_name`
4. Attempt to access the container applications. After you successfully access the applications, enter the following command to generate the rules used to access the container:
`getrules -m container_name`
5. Compare the output from the `getrules` command with the compartment rules file and make the necessary changes.
6. Stop the container, remove the `discover` keyword from the compartment rules file, and then restart the container.

18.4.2 Removing or disabling IPFilter

If you are using IPFilter with HP-UX Containers, you can see if IPFilter rules are blocking access to the container applications. You can do this by removing the `ipfilter` service from the container, as follows:

```
srp -d container_name [-t template] -s ipfilter
```

If you do not specify the `-t` argument, the `srp` command removes the IPFilter configuration for the template (`base` for the workload container and `system` for the system container).

To add the `ipfilter` service back to the container after you have completed your testing, enter:

```
srp -d container_name [-t template] -s ipfilter
```

Another method to test if IPFilter rules are blocking access to the container applications is by disabling the IPFilter module, as follows:

```
/opt/ipf/bin/ipfilter -d
```

To enable IPFilter after you have completed testing, enter:

```
/opt/ipf/bin/ipfilter -e
```

18.4.3 Removing or disabling IPSec

If you are using IPSec with HP-UX Containers, you can see if IPSec policies are blocking access to the container applications. One method to determine if IPSec policies are blocking packets is by removing the `ipsec` service from the container, as follows:

```
srp -d container_name -s ipsec
```

To add the `ipsec` service back to the container after you have completed testing, enter:

```
srp -d container_name -s ipsec
```

Another method to test if IPSec policies are blocking access to the container applications is by stopping the IPSec product, as follows:

```
/usr/sbin/ipsec_admin -stop
```

To restart IPSec after you have completed testing, enter:

```
/usr/sbin/ipsec_admin -start
```

18.5 Reporting Problems

If you are unable to solve a problem with HP-UX Containers, complete the following steps:

1. Read the published release notes for HP-UX Containers to see if the problem is known. If it is a known issue, use the prescribed solution.
2. Determine whether the product is still under warranty or whether your company purchased support services for the product. Your operations manager can supply you with the necessary information.
3. Access <http://www.itrc.hp.com/> and search the technical knowledge databases to determine if the problem you are experiencing has already been reported. The type of documentation and resources you have access to depend on your level of entitlement.

NOTE: The ITRC resource forums at <http://www.itrc.hp.com/> offer peer-to-peer support to solve problems and are free to users after registration.

If this is a new problem or if you need additional help, log your problem with the HP Response Center, either on line through the support case manager at <http://www.itrc.hp.com/>, or by calling HP Support. If your warranty has expired or if you do not have a valid support

contract for your product, you can still obtain support services for a fee, based on the amount of time and material required to solve your problem.

4. If you are requested to supply any information pertaining to the problem, gather the necessary information and submit it.

Include the following information:

- The output from the following command:
`srp -l container_name -v`
- The contents of the container initialization log file,
`/var/hpsrp/container_name/etc/rc.log`.
- A description of the applications hosted in the container, including version information.

Glossary

compartment	Security Containment compartments. Manages isolation and privilege restrictions for sets of HP-UX processes. Each container includes a corresponding compartment definition.
container	A container provides process view isolation, IPC isolation, and a dedicated IP address interface. HP-UX Containers includes two types of containers: system and workload.
container administrator	A global view user that has been granted the administrator role to manage one or more containers. This user can perform start, stop, list, and view on designated containers.
container state	<p>stopped: The container is not available. Its network interface is down, and no container filesystems are mounted. Accessing the container using <code>srp_su(1M)</code> in the stopped state is not allowed.</p> <p>started: The container is up and running, and it is accessible by users. The network interface is up, container filesystems are mounted, and container service daemons started.</p> <p>maintenance: SD is performing software management (install or remove software) in a container. Container service daemons are stopped and users are not allowed to access the container in this state.</p> <p>starting: The container is starting and is not accessible by users. The <code>srp</code> command will initiate default run level processing (see <code>init(1M)</code>) immediately before transitioning to the started container state. Run level processing will complete after the container enters the started state.</p> <p>stopping: The container is stopping and is not accessible by users. The <code>srp</code> command will attempt to gracefully shutdown all container processes by initiating run-level 0 processing (see <code>init(1M)</code>). All container processes not gracefully shutdown will be terminated. In the unlikely event that the normal shutdown process fails, the <code>srp</code> command will log the name and process id of all the processes it terminates in the system log file.</p>
global view	When you enable SRP on a system using the <code>srp_sys</code> command, all processes not executing within a container execute in the global view. The global view has no access restrictions, and therefore can view and manage processes in the global view and all containers. Processes in the global view are typically assigned to the compartment named <code>INIT</code> .
HP-UX Containers	Product family for containment technology on HP-UX, including Secure Resource Partitions (SRP) and HP9000 Containers.

private file system	A system container subtype. The private file system has only the <code>/stand</code> directory from the global view. All other directories are private to the container.
shared file system	A system container subtype. The shared file system has the <code>/usr</code> , <code>/sbin</code> , and <code>/stand</code> directories from the global view mounted as read-only. All other directories are private to the container.
system container	System containers provide process view isolation, IPC isolation, and a dedicated IP address interface. System containers have a private administrative domain and service daemons. System containers provide a private namespace for file system view, hostname, nodename, System V IPC, and loopback IP address. System containers require SD software installation synchronization with the global view.
workload container	Workload containers provide process view isolation, IPC isolation, and a dedicated IP address interface. Workload containers share the global view administrative domain and service daemons. Workload containers provide shared loopback IP address connectivity with other workload containers and the global view. Workload containers do not require separate software installation synchronization with the global view.

Appendix A: Container default route script for Serviceguard

The following script can be used by a Serviceguard package to assign a default route for an IP address associated with a container. This script is included with the HP-UX Containers Serviceguard Reference Implementation for containers and is installed with the HP-UX Containers product at: `/opt/hpsrp/example/serviceguard/srp_as_sg_package/srp_route_script`

```
# Copyright (c) 2009 Hewlett-Packard Development Company L.P.
#
# This script runs the 'route' command to manage source based routing entry
# for the SRP.
#
# This script should be configured into the package configuration file
# as the first "external_script" parameter entry. It will be executed
# right after Serviceguard IP addresses assignment during package start time,
# and before removing IP addresses during package halt time.
#
# This script uses the environment variable SRP_SG_MANAGED_IP and
# SRP_SG_GATEWAY. The environment variables must be set in the
# srp_script.incl file in the same directory as this script.
#

#####
# Source utility functions.
#####

if [[ -z $SG_UTILS ]]
then
    . /etc/cmcluster.conf
    SG_UTILS=$SGCONF/scripts/mscripts/utils.sh
fi

if [[ -f ${SG_UTILS} ]]; then
    . ${SG_UTILS}
    if (( $? != 0 ))
    then
        echo "ERROR: Unable to source package utility functions file:
${SG_UTILS}"
        exit 1
    fi
else
    echo "ERROR: Unable to find package utility functions file: ${SG_UTILS}"
    exit 1
fi

#####
#
# Get the environment for this package through utility function
# sg_source_pkg_env().
#
#####

sg_source_pkg_env $*

#####
#
# Get the SRP environment from "/etc/cmcluster/hpsrp/<srp>/srp_script.incl"
#
# Environment variable example: use a local gateway on the host
# SRP_SG_MANAGED_IP[0]="192.0.0.99"
# SRP_SG_GATEWAY[0]="192.0.0.99"
#
# Environment variable example: use a remote gateway
# SRP_SG_MANAGED_IP[1]="10.1.1.99"
```



```

# SRP_SG_GATEWAY[1]="10.1.1.1"

#
#####

. `dirname $0`/srp_script.incl

#####
#
# Functions
#
#####

# add routing entry
function srp_route_add
{
    # run 'route' command for each IP address
    rval=0
    index=0
    last_index=${#SRP_SG_MANAGED_IP[@]}
    while [ "$index" -lt "$last_index" ]
    do
        srp_ip="${SRP_SG_MANAGED_IP[$index]}"
        srp_gateway="${SRP_SG_GATEWAY[$index]}";
        if [ -z "$srp_ip" ] # skip empty slot in the array
        then
            let index=$index+1
            let last_index=$last_index+1
            continue
        fi
        if [ "$srp_ip" = "$srp_gateway" ]
        then
            # use local IP as gateway
            emsg=$(/usr/sbin/route add default $srp_gateway 0 \
                source $srp_ip 2>&1)
        else
            # use remote gateway
            emsg=$(/usr/sbin/route add default $srp_gateway 1 \
                source $srp_ip 2>&1)
        fi
        if (($? != 0)); then
            print "ERROR: $emsg" >$2
            rval=1
        fi
        let index=$index+1
    done
    return $rval
}

# delete routing entry
function srp_route_delete
{
    # run 'route' command for each IP address
    rval=0
    index=0
    last_index=${#SRP_SG_MANAGED_IP[@]}
    while [ "$index" -lt "$last_index" ]
    do
        srp_ip="${SRP_SG_MANAGED_IP[$index]}"
        srp_gateway="${SRP_SG_GATEWAY[$index]}";
        if [ -z "$srp_ip" ] # skip empty slot in the array
        then
            let index=$index+1
        let last_index=$last_index+1
        continue
        fi
        if [ "$srp_ip" = "$srp_gateway" ]

```

```

        then
            # use local IP as gateway
            emsg=$(/usr/sbin/route delete default $srp_gateway 0 \
                source $srp_ip 2>&1)
        else
            # use remote gateway
            emsg=$(/usr/sbin/route delete default $srp_gateway 1 \
                source $srp_ip 2>&1)
        fi
        if (($? != 0)); then
            print "ERROR: $emsg" >$2
            rval=1
        fi
        let index=$index+1
    done
    return $rval
}

#####
# main routine
#####

sg_log 5 "SRP routing entry configuration script"

#####
#
# Customer defined external script must be specified with three required
# entry points: start, stop, and validate.
#
# It's not recommended to add additional entry points to the script
# due to potential name space collision with future Serviceguard releases.
#
#####

typeset -i exit_val=0

case ${1} in
    start)
        srp_route_add
        exit_val=$?
        ;;

    stop)
        srp_route_delete
        exit_val=$?
        ;;

    validate)
        exit_val=0
        ;;

    *)
        sg_log 0 "INFO: Unknown operation: $1"
        ;;
esac

exit $exit_val

```

Appendix B: Direct customization of container properties

In most cases, the `srp` command is sufficient to modify the properties of a container. However, you can directly modify the container specific scripts of system configuration entries to:

- Execute customer defined operations from the global view when a container is created, deleted, or started and stopped (not supported for system containers).
- Customize security containment definition (not supported for system containers).
- Configure subsystem parameters not controlled by the `srp` command.

B.1 Execute customer defined operations via provision scripts

A provision script performs the tasks needed to provision a container or deploy an instance of an application in a container. These tasks can include copying data from an application's normal installation directory to the home directory for the container, or in the case of the `custom` template-apply customer defined operations. The `srp` command passes selected arguments and variables to the provision scripts, such as the `srp` operation, the container name, container IP address, container data and execution paths, and other application-specific variables.

You can modify the provision scripts to add tasks needed to deploy an application. The provision scripts provided with HP-UX Containers are:

- `apache`: `/opt/hpsrp/bin/util/apache_setup`
- `tomcat`: `/opt/hpsrp/bin/util/tomcat_setup`
- `ssh`: `/opt/hpsrp/bin/util/secsh_setup`
- `custom`: provided as an input variable to the `srp- add` operation

B.2 Customize security containment definition (not supported for system containers)

The `srp` command uses include files to configure Security Containment compartment rules. There is an include file for each template, including the primary template that defines the container type. If you modify the contents of an include file for a template, all containers that have applied the template will use the modified include file.

The include file names have the following format:

```
/opt/hpsrp/etc/cmpt/template_name.srp_incl
```

For example, `/opt/hpsrp/etc/cmpt/apache.srp_incl`.

B.2.1 Securing containers with compartment rule Include files (Not supported for system containers)

The primary template rules file delivered with the product provides a rule set designed to allow maximum application compatibility while providing restricted access to files not needed to be modified or accessed by applications or user sessions. To increase the security of your environment, you can replace this file with a more restrictive rule set tuned to your application requirements and local security policy.

You can create an environment with the minimal compartment access rights, as follows:

1. Make a copy of the default base compartment rules file, `/opt/hpsrp/etc/cmpt/base.srp_incl`. For example:

```
# cd /opt/hpsrp/etc/  
# cp base.srp_incl myCustom.srp_incl
```

2. Remove the rules in the original (`base.srp_incl`) file. This creates an empty security compartment rules file. A container that uses only this file for its compartment rule set will have no access to any files, system IPC, or network interfaces.

NOTE: Creating an empty security compartment rules file for the `base` template files affects all containers using this file, including those previously created. HP recommends this practice in a highly secure environment to ensure that all containers are specifically configured, and that no containers are continuing to execute with default rules.

3. Determine the minimum set of rules that you need for a compartment and add them to the new file (`myCustom.srp_incl` in this example). For more information on creating a deployment-specific compartment rules set, see *HP-UX System Administrator's Guide: Security Management: HP-UX 11i Version 3*.
4. Use the custom template to associate this new rules file to compartments requiring the specified access. For example:

```
# srp -a myContainer -template custom -id myID
```

When `srp` prompts for `Compartment rule files`, enter the name of the new file (`/opt/hpsrp/etc/myCustom.srp_incl` in this example)

B.2.2 Manually editing configuration data

HP-UX Containers marks the data it adds to subsystem configuration files and databases with *tags*, or text-string identifiers. It uses these tags when selecting data for the `srp replace` and `delete` operations. You can use these tags to identify and manually edit the HP-UX Containers configuration data and still use the `srp replace` and `delete` operations to manage this data if you retain the tag information.

To identify configuration data that HP-UX Containers manages, enter:

```
srp -l container_name -v
```

B.2.2.1 Tag Formats

The general format for most tags that indicate the start of HP-UX Containers data is as follows:

```
@tag-start 'compartment="container_name" template="template_name"  
service="service_name" id="version";
```

Where:

container_name Specifies the container name.

template Specifies the name of the template used to configure the data

service Specifies the service name.

id A string used to identify an instance of a service applied to a container. This field allows you to create multiple instances of related configuration entries for one or more services. All tags will include an `id`, with a default value of 1.

The specific tag format for each subsystem is described in the sections that follow.

B.2.2.2 Security Containment compartment tag format

NOTE: Customization of the Security Containment compartment rules file is not supported for system containers.

Data is stored in the `/etc/cmpt/container_name.rules` file by default. When the `srp` command adds data, it indicates the start of the data with the following tag:

```
//@tag-start 'compartment="container_name" template="template_name"
service="cmpt" id="instance";
```

The `srp` command indicates the end of the data with the following tag:

```
//@tag-end;
```

B.2.2.3 RBAC and compartment login tag format

Data is stored in files under the `/etc/rbac` directory. HP recommends that you use RBAC commands (`roleadm`, `authadm`, `cmdprivadm`) to modify RBAC data.

HP-UX Containers identifies RBAC data for the `admin` service by using the following values:

- Role name: `SRPadmin-container_name` for the container
- Authorization: `hpux.SRPadmin.container_name` for the container
- Command privilege: `hpux.SRPadmin.container_name` for the container

HP-UX Containers identifies RBAC data for the `login` service by using the following values:

- Role name: `SRPlogin-container_name` for the container
- Authorization: `hpux.security.compartment.login` for the container

B.2.2.4 Network configuration tag format

For IPv4 interfaces, the `srp` command adds the following entry to the `/etc/rc.config.d/netconf` file:

```
IPV4_CMGR_TAG[index]='compartment="container_name" template="base"
service="network" id="instance"'
```

Where `index` is the first available index number for interface parameters in the `netconf` file. HP-UX Containers uses the index number to identify the following interface parameters:

```
INTERFACE_NAME
INTERFACE_SKIP
IP_ADDRESS
SUBNET_MASK
INTERFACE_STATE
BROADCAST_ADDRESS
DHCP_ENABLE
INTERFACE_MODULES
```

HP-UX Containers uses the address configured for the `IP_ADDRESS` entry to identify the `ROUTE_SOURCE` entry for the container, and uses that index number to identify the corresponding route entries.

IPv6 Interfaces

The data is similar for IPv6 interfaces, with the following differences:

- The data is stored in the `/etc/rc.config.d/netconf-ipv6` file.
- The names of the interface parameters are correct for IPv6 interfaces, such as `IPV6_INTERFACE`, `IPV6_ADDRESS`, `IPV6_INTERFACE_STATE`.
- HP-UX Containers does not add or manage IPv6 route entries.

B.2.2.5 PRM Tag Format

Data is stored in the `/etc/prmconf` file by default. When the `srp` command adds data, it indicates the start of the data with the following tag:

```
#@tag-start compartment="container_name" template="base" service="prm"
id="instance";
```

The `srp` command indicates the end of the data with the following tag:

```
#@tag-end;
```

B.2.2.6 IPFilter Tag Format

Data is stored in the `/etc/opt/ipf/ipf.conf` file for IPv4 addresses and in `/etc/opt/ipf/ipf6.conf` for IPv6 addresses. When the `srp` command adds data, it indicates the start of the data with the following tag:

```
#@tag-start compartment="container_name" template="template_name"
service="ipfilter" id="instance";
```

HP-UX Containers indicates the end of the data with the following tag:

```
#@tag-end;
```

B.2.2.7 IPSec Tag Format

IPSec stores configuration data in the IPSec database, `/var/adm/ipsec/config.db`. To modify the contents of the IPSec database, you must use the `ipsec_config` utility.

The configuration objects (IPSec host policy, IKE policy, and authentication record) each have a name with the following format:

```
SRP-container_name-base-ipsec
```

Appendix C: Template services – detailed description

C.1 The cmpt service

The `cmpt` Service configures an HP-UX Security Containment compartment, which forms the core of each container. You must use the `cmpt` service when you create a container.

C.1.1 Configuration location

The `cmpt` service creates a home directory for the container using the following format:
`/var/hpsrp/container_name`

The `cmpt` service creates a home directory for the container (`/var/hpsrp/container_name`) and then creates a Security Containment compartment rules file (`/etc/cmpt/container_name.rules`). This file will contain `#include` references to files with additional rules for the container including the common rules files shared by all containers of the same type (for example, the `/opt/hpsrp/etc/cmpt/sysbase.srp_incl` file).

Compartment rules control:

- Access to absolute file system paths
- Ownership of IP network interfaces
- Disallowed kernel level privileges
- Loopback Network access to other containers
- IPC access to specified containers (shared namespace only)
- Disallowed kernel privileges

See *HP-UX System Administrator's Guide: Security Management: HP-UX 11i Version 3* for details on configuring Security Containment compartments.

C.2 The admin service

The `admin` service associates HP-UX users with an RBAC role that has authorization to administer the container from the global view. By default, this authorization enables the administrator to start, stop, and report status of the container.

C.2.1 Configuration location

The `admin` service uses RBAC commands to add information about the administrator. By default RBAC stores this information in the RBAC `/etc/rbac` configuration directory.

The `admin` service performs the following tasks:

- Creates a role with the name `SRPadmin-container_name` for the container. The RBAC `roleadm add` command is used to perform this task.
- Creates an authorization with the name `hpux.SRPadmin-container_name` with the object set to the container. The RBAC `authadm add` command is used to perform this task.
- Assigns the authorization `hpux.SRPadmin.container_name` to the role `SRPadmin-container_name`. The RBAC `authadm assign` command is used to perform this task.
- Associates the specified user name to the role `SRPadmin-container_name`. The user name must already exist. The RBAC `roleadm assign` command to perform this task.

- Assigns the authorization `hpux.SRPadmin-container_name` to execute the container master startup script `/opt/hpsrp/bin/srp_rc` in the container. This enables the administrator to start up and shut down the container. The RBAC `cmdprivadm add` command is used to perform this task.

Configuring an administrative user does not grant that user login or `srp_su` access to the compartment.

C.3 The prm service

The `prm` service creates a new PRM group for a container. The `srp` command does not provide an interface to add a container to an existing PRM group. To add a container to an existing PRM group, use the procedure described in *HP Process Resource Manager User's Guide*, "Assigning secure compartments to PRM groups."

C.3.1 Configuration data

By default, the `srp` command creates a new PRM group using the container name as the PRM group name.

By default, the PRM group information is stored in the `/etc/prmconf` file. You can change the filename by running the `srp_setup` command, see `srp_setup(1M)`. Refer to the *HP Process Resource Manager User Guide* for more information about the system resource allocation.

C.4 The network service

The `network` service configures an IP interface for the container. You do not have to use a dedicated network interface card for the container; you can create a logical IP interface on a network interface card.

C.4.1 Configuration location

The `srp` command configures IP interface information for the HP-UX Transport subsystem, the initialization and shutdown service, and for the container, as described in the following sections.

HP-UX transport

If you specify an IP address that is not already configured for the system, the `srp` command also configures the IP interface information for the HP-UX Transport subsystem.

IPv4 address

If you specify an IPv4 address, the `srp` command adds configuration data to the `/etc/rc.config.d/netconf` file.

If you specify a primary interface name, the `srp` command assigns the address to the primary interface if the primary interface does not already have an IP address configured. If the primary interface already has an IP address configured, the `srp` command creates a new secondary (logical) interface using the next available IP index number for that primary interface. If you specify a secondary interface name and the corresponding primary interface is not already configured, the `srp` command displays an error message.

The `srp` command adds the interface to the `/etc/rc.config.d/netconf` file with the `INTERFACE_SKIP` set to `true` and `INTERFACE_STATE` set to `up`. The interface is brought up when the container is started.

Route information

The `srp` command provides an option to add or modify the default gateway routing table entry for the container IP address. The container IP address is always used as the source IP address.

If no target default gateway IP address is provided, the container IP address is used, with a hop (route) count set to 0. If a target default gateway IP address is provided, the hop (route) count is set to 1. The `srp` command adds the routing configuration data to `/etc/rc.config.d/netconf` (for IPv4) and `/etc/rc.config.d/netconf-v6` (for IPv6).

IPv6 address

If you specify an IPv6 address, the `srp` command adds configuration data to the `/etc/rc.config.d/netconf-ipv6` file.

If you specify an address that is not a link local address and a primary interface that does not already have an IP address configured, the `srp` command configures the primary interface with the appropriate link local address, then configures a secondary interface with the specified IPv6 address.

The `srp` command adds the interface to the `/etc/rc.config.d/netconf-ipv6` file with the `INTERFACE_SKIP` set to `true` and `INTERFACE_STATE` set to `up`. The interface is brought up when the container is started.

Security Containment compartment

The `srp` command adds a network interface rule for the IP address to the compartment rule file (`/etc/cmpt/container_name.rules`). This allows the container access to its IP address.

C.5 The init Service

The `init` service creates startup and shutdown scripts for the container, and a container-specific `/var/hpsrp/<srp name>/sbin/init.d` directory structure that replicates the `/sbin/init.d` directory structure. The `srp` command also configures the `autostart` feature for the container so that the system startup and shutdown scripts automatically execute the container startup and shutdown scripts.

The `init` service will also populate the container with the appropriate set of system files for the container type specified.

C.5.1 Configuration location

The `srp` command configures the following data:

- Adds the following entries to the `/etc/opt/hpsrp/srps/container_name.srp` file to enable the `autostart` feature for the container:

```
srp_name="container_name"
autostart=1
```

- Creates SRP-specific `init` subdirectories below the `/var/hpsrp/container_name/sbin` that contain startup and shutdown scripts. For more information about the directory structure, files, and how they are executed at system startup and shutdown time, see

- 11 Container startup and shutdown.

C.6 The login service (workload containers)

The `login` service enables you to specify the set of users and user groups whose members are authorized to log in to the container. If you do not configure the `login` service and you are using the default RBAC system configuration, only the `root` user is authorized to log in to the container.

You can use the `login` service to grant non-root users the authorization to log in to the container.

C.6.1 Configuration location

The `login` service controls login access to the container using the Security Containment compartment login feature. It uses RBAC authorizations to allow specified users and group members to pass PAM authentication in the module `pam_hpsec`, which controls PAM-enabled authentication services (used by `login`, `ftp`, and other user session services) occurring within the container.

The `login` service performs the following tasks:

- Creates the role `SRPlogin-container_name`. The `srp` command uses the `roleadm add` command to perform this task.
- Assigns the specified user or group ID to the `SRPlogin-container_name` role. The `srp` command uses the `roleadm assign` command to perform this task.
- Assigns the `SRPlogin-container_name` role login authorization (the authorization `hpux.security.compartment.login`) for the container. The `srp` command uses the `authadm` command to perform this task.

C.7 The ipfilter service

The `ipfilter` service configures HP-UX IPFilter for the container. The IPFilter configuration for the primary template allows the following packets to pass to the container:

- All outbound packets from the container IP address
- Inbound TCP, UDP, and ICMP responses to packets sent from the container IP address.
- All inbound ICMP packets to the container IP address.

All other inbound packets are blocked.

You can also configure IPFilter to allow inbound and outbound IPsec packets to pass.

C.7.1 Configuration location

If the container address is an IPv4 address, the `srp` command adds IPFilter rules to the `/etc/opt/ipf/ipf.conf` file. If the container address is an IPv6 address, the `srp` command adds IPFilter rules to the `/etc/opt/ipf/ipf6.conf` file.

The `srp` command adds the following IPFilter rules for the container, where `cmpt_address` is the container IP address:

- Rules that allow all TCP, UDP, and ICMP outbound packets from the container IP address. These rules specify the `keep` state keywords to allow inbound replies for these packets:

```
pass out quick proto tcp from container_address to any keep state
pass out quick proto udp from container_address to any keep state
pass out quick proto icmp from container_address to any keep state
```

If the container address is an IPv6 address, the last rule is `pass out quick proto icmpv6` from `container_address` to any keep state.

- A rule that allows inbound ICMP packets from any address to the container IP address:

```
pass in quick proto icmp from any to container_address
```

If the container address is an IPv6 address, the rule is `pass in quick proto icmpv6` from any to `container_address`.

- A rule that blocks all inbound packets to the container IP address:

```
block in quick from any to container_address
```

Rule order and selection

By default, IPFilter selects a rule for a packet by reading the rules in a configuration file from top to bottom and selects the last rule that matches a packet. The `quick` keyword changes this behavior and causes IPFilter to immediately apply the rule to a packet if it matches the filter (instead of continuing to evaluate rules for the packet). When using the `quick` keyword, rules are generally ordered from most specific to least specific.

The `srp` command specifies the `quick` keyword in the IPFilter rules it configures. It inserts these rules at the top of the IPFilter configuration file in the order applied.

IPFilter rules

If you specify that you want to add IPFilter rules for IPsec, the `srp` command also adds IPFilter rules that allow IPsec Encapsulating Security Payload (ESP; protocol 50) and Authentication Header (AH; protocol 51) packets and IPsec control packets (Internet Key Exchange, or IKE; UDP port 500) to pass. These rules are inserted above the more general IPFilter rules for the container. For more information, see *12 Networking with containers*.

C.8 The ipsec service

The `ipsec` service configures HP-UX IPsec to encrypt and authenticate IP packets between the container IP address and a remote IP address.

C.8.1 Configuration location

The `srp` command adds IPsec configuration data using the `ipsec_config` utility. IPsec adds the data to the IPsec database, `/var/adm/ipsec/config.db`. To view the contents of the IPsec database, use the `ipsec_config` or the `ipsec_report` utility. To modify the contents of the IPsec database, you must use the `ipsec_config` utility.

The `srp` command adds the following IPsec configuration data:

- A host IPsec policy

The host policy specifies encryption and authentication using the specified transform between the specified remote IP address and the local (container) IP address. The default HP-UX IPsec values are used for all other parameters.

- An Internet Key Exchange (IKE) policy

The IKE policy specifies parameters used to establish an IKE security association with the specified remote IP address. The authentication method is PSK (preshared key). The default HP-UX IPSec values are used for all other parameters.

- An authentication record

The authentication record contains the specified remote IP address and preshared key value. The default HP-UX IPSec values are used for all other parameters.

HP-UX IPSec default parameter values

For IPSec parameters not directly managed by the `srp` command, default values are read from the IPSec profile file, `/var/adm/ipsec/.ipsec_profile`. You can view this text file to determine the default IPSec parameters and determine what values need to be configured on the peer system. Some of the main parameters and the default values set in the factory-installed profile file are as follows:

- IKE exchange type: Main Mode
- IKE hash algorithm: MD5
- IKE encryption algorithm: 3DES
- IKE Diffie-Hellman group: 2

Policy selection and priority

When IPSec selects policies, it selects the first policy that matches the search criteria. Because of this selection algorithm, IPSec policies are typically ordered from most specific to least specific. The `srp` command adds the policies using the IPSec automatic priority increment mechanism, where IPSec determines the priority for a new policy by adding n to the current highest priority for that policy category, where n is the automatic priority increment value. When a policy is added with this mechanism, it becomes the last policy evaluated before the default policy in the category; you might have to modify the priority value for your policies.

Using IPSec with IPFilter

HP-UX IPFilter is located below HP-UX IPSec in the networking stack. HP-UX IPFilter processes inbound IP packets before HP-UX IPSec and processes outbound packets after HP-UX IPSec.

To use IPSec with IPFilter, you must configure IPFilter to pass the following packets:

- IP packets with protocol 50 (IPsec Encapsulating Security Payload protocol, ESP)
- IP packets with protocol 51 (IPsec Authentication Header protocol, AH)
- UDP packets with port 500 (IPsec Internet Key Exchange protocol, IKE)

If HP-UX IPSec secures a packet (the packet has an AH or ESP header), HP-UX IPFilter cannot filter the packet based on upper layer information, such as TCP port numbers and connection states, and ICMP message types. The only upper-layer protocol information that HP-UX IPFilter processes is the IP protocol number. IPSec packets do not match any IPFilter rules based on the TCP, UDP, or ICMP protocol type or based on field values for these protocols (such as port numbers).

Technology for better business outcomes

© Copyright 2011 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

5900-1837, July 2011



Get connected

www.hp.com/go/getconnected

Current HP drivers, support & security alerts
delivered directly to your desktop

