Managing System Services in Oracle® Solaris 11.4

Part No: E60998
February 2022

Managing System Services in Oracle Solaris 11.4

Part No: E60998

Contents

Figures

Tables

Examples

Using This Documentation

- **Overview** – Describes how to use the Oracle Solaris Service Management Facility (SMF) feature. SMF is one of the components of the wider Oracle Solaris Predictive Self Healing capability.
- **Audience** – System administrators who manage system services
- **Required knowledge** – Experience administering Oracle Solaris systems

Product Documentation Library

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

Feedback

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

♦ ♦ ♦ C H A P T E R 1

Introduction to the Service Management Facility

The Oracle Solaris Service Management Facility (SMF) framework manages system and application services. SMF manages critical system services essential to the working operation of the system and manages application services such as a database or Web server. SMF improves the availability of a system by ensuring that essential system and application services run continuously even in the event of hardware or software failures.

SMF replaces the use of configuration files for managing services and is the recommended mechanism to use to start applications. SMF replaces the `init` scripting start-up mechanism, `inetd.conf` configurations, and most `rc?.d` scripts. SMF preserves compatibility with existing administrative practices wherever possible. For example, most customer and ISV-supplied `rc` scripts still work the same way they worked without SMF.

This chapter describes:

- Capabilities of SMF
- New features in this release
- Concepts and components of SMF
- Configuration files versus SMF services
- How to gain privileges you need to use some SMF commands

For information about developing custom SMF services, see *Developing System Services in Oracle Solaris 11.4*.

SMF Capabilities

The SMF framework is always active on an Oracle Solaris 11 system. SMF provides the following capabilities:

- Boot faster. SMF speeds booting of large systems by starting independent services in parallel.

- Restart failed services. SMF services have well defined dependency relationships with other services. If a service fails, SMF reports any affected dependent services. SMF automatically attempts to restart failed services in dependency order.
- Inspect services. View the relationships between services and processes. View the values of service properties.
- Manage services. Enable, disable, and restart services. These changes can persist through upgrades and reboots, or you can specify temporary changes.
- Configure services.
  - Change the values of service properties.
  - Add and delete custom properties.
- Audit service changes. SMF writes Solaris audit records for every administrative change to a service or its properties. SMF can show whether a property value or service state was set by an administrator.
- Securely delegate tasks to non-root users, including the ability to modify properties and enable, disable, or restart services.
- Configure how you will be notified of particular software events or hardware faults.
- Debug service problems. Easily display an explanation for why an enabled service is not running or why a service is preventing another service from running.
- Create a new instance of an existing service or modify an existing service instance.
- Create new services. See *Developing System Services in Oracle Solaris 11.4* for more information about the following capabilities:
  - Using the service creation tool.
  - Converting `inetd.conf` configurations to SMF services.
  - Converting SMF service properties to configuration files. This mechanism provides a bridge for services that are managed by SMF but interact with applications that still require configuration files.
  - Creating a service that runs periodically rather than continuously, similar to a `cron` job.

New Features in This Release

The following SMF features are new in this release:

Additional repository layers

Three new repository layers enable finer grained specification of configuration. See "Repository Layers" on page 30 and "Creating SMF Profiles" on page 108 for more information.

1. `enterprise-profile` layer – Configuration that applies across all of the Oracle Solaris systems for an enterprise.

2. `node-profile` layer – Configuration that is specific to a particular Oracle Solaris instance.

   The existing `site-profile` layer had been used for any configuration that was not delivered by Oracle Solaris, such as configuration delivered by `sysconfig` or by user profiles. In Oracle Solaris 11.4, the `site-profile` layer is for configuration that is common to many systems at the same location or site. Most configuration that previously belonged to the `site-profile` layer now belongs to the `node-profile` layer.

   When a system is initially updated to Oracle Solaris 11.4, all profiles in the `/etc/svc/profile/site/` directory and the `/etc/svc/profile/site.xml` profile, if it exists, are moved to the `/etc/svc/profile/node/` directory. The configuration these profiles describe is then part of the `node-profile` layer. Moved profiles are named to indicate that they were moved and where they were previously located. For example, the `/etc/svc/profile/site/sc_profile.xml` profile might be moved to `/etc/svc/profile/node/migrated_etc_svc_profile_site_sc_profile.xml`, and `/etc/svc/profile/site.xml` might be moved to `/etc/svc/profile/node/migrated_etc_svc_profile_site.xml`.

   Any file in the `/etc/svc/profile/site/` directory that was delivered by an IPS package is not moved, and that configuration remains part of the `site-profile` layer.

3. `sysconfig-profile` layer – Configuration that is specified by using the `sysconfig` command interactively, and any configuration that is specified in a profile that is installed by an AI client or by the `zoneadm` (`install`, `attach`, or `clone`) command.

   When a system is initially updated to Oracle Solaris 11.4, all `.xml` files in the `/etc/svc/profile/sysconfig/` directory are backed up into the `/etc/svc/profile/backup/`*timestamp*`/profiles.tar` file. Some of the profiles in this directory might contain configuration performed by `sysconfig` or the Oracle Solaris installer and might contain active configuration at the `admin` layer. Profiles that contain active configuration are left in place, effectively migrating the configuration from the `admin` layer to the `sysconfig-profile` layer. Any profile in the `/etc/svc/profile/sysconfig` directory that is not referenced by the SMF repository is removed after being backed up into the `/etc/svc/profile/backup/`*timestamp*`/profiles.tar` file.

Nested property groups

The parent of a property group can be a service or service instance or another property group. Another way to describe this feature is a property group can have properties and can have any number of child property groups. Nesting property groups enables finer definition of relationships among configuration data. To identify properties in a nested property group, name all the parent property groups as well, as shown in "Inspecting Service Configuration" on page 49.

Property group name and property name options

The `-G` option can be used with the `svcprop` and `svccfg` commands to specify a property group. The `-P` option can be used to specify a property. The `-T` option can be used with the `svccfg` command to specify a property type. See "Inspecting Service Configuration" on page 49.

Special characters in property group names and property names

Property group names and property names can contain any character defined in Uniform Resource Identifier (URI) Generic Syntax RFC 3986. See "Naming Property Groups and Properties" in *Developing System Services in Oracle Solaris 11.4*.

Stencil defines

If your application requires multiple configuration files that have the same syntax, you can use the stencil defines feature to use a single service to define all configuration files.

Goal services

A goal service provides a single point of monitoring for a configurable set of dependent services. Most services that cannot reach the `online` state remain silently in the `offline` state. A goal service transitions to the `maintenance` state and generates an FMA alert if any of its dependencies cannot be satisfied without administrative intervention.

`svcadm goals` command

The `svcadm goals` command configures goal dependencies for a goal service. See "Changing the Goals of a Goal Service" on page 75. As with other subcommands of the `svcadm` command, the `svcadm goals` command can take the `-s` option to request synchronous behavior.

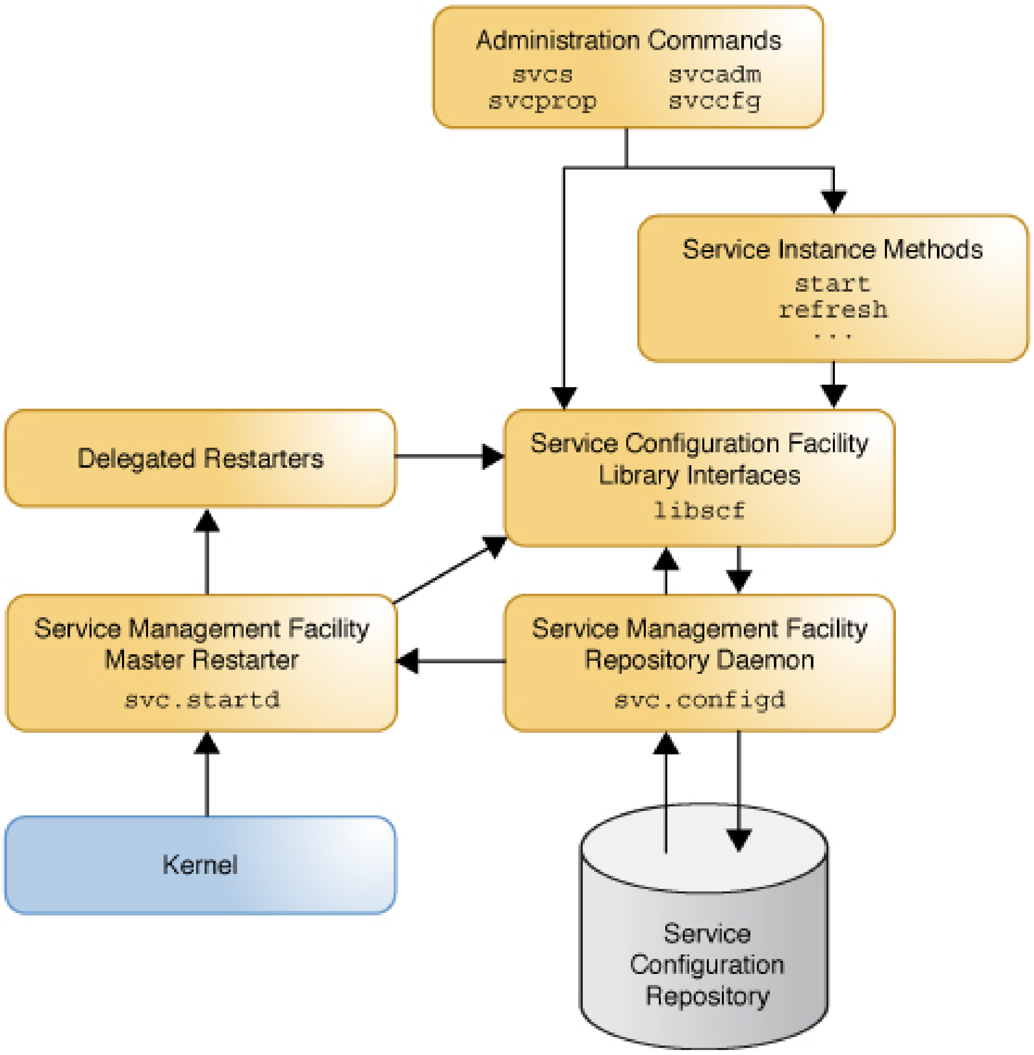
SMF Concepts and Components

This section defines terms that are used in the remainder of this guide.

The following figure shows the primary components of the SMF framework. When you boot an image, SMF updates the service configuration repository if necessary, reads the repository data, and starts enabled service instances in correct dependency order. Independent services are started in parallel. When you shut down an image, services are shut down in reverse dependency order.

In the following figure, `libscf` is the library interface that the restarters use to interact with the service configuration repository. Interaction between the service configuration repository and `libscf` library interfaces is managed by the `svc.configd` daemon. The `svcs`, `svcprop`, `svcadm`, and `svccfg` commands are the interface that administrators use to interact with the service configuration repository.

FIGURE 1 Service Management Facility Framework


Administration Commands
svcs
svcadm
svcprop
svccfg
Service Instance Methods
start
refresh
...
Delegated Restarters
Service Configuration Facility Library Interfaces
libscf
Service Management Facility Master Restarter
svc.startd
Service Management Facility Repository Daemon
svc.configd
Kernel
Service Configuration Repository

SMF Service

An SMF *service* is a persistently running application that represents a system entity such as the following:

- Application services such as a database or a Web server
- Essential system services
- The software state of a device
- Kernel configuration information
- Milestones that correspond to a level of system readiness

A *service instance* is a child of a service and provides capabilities and dependency relationships to applications and other service instances. Only instances have a state and can be started and stopped. If an instance fails for any reason, such as a hardware or software fault, SMF automatically detects the failure and restarts the instance and any dependent instances.

Instances of a service allow multiple configurations of a service to run simultaneously. Service instances inherit and customize common service configuration. For example, you can define a Web server service with one instance configured to listen on port 80 and another instance configured to listen on port 1008. Most services have a `default` instance. A few services do not have instances, such as some services that use SMF to store configurations but not to run programs. For example, the `x11/x11-server` service does not have any instances.

An SMF service is described in a file called a service *manifest*. The manifest describes service instances, dependencies, configuration properties, and methods. Service *methods* start, stop, and refresh service instances. A method can be a daemon, other binary executable, or an executable script. A service *profile* file enables you to customize an existing service, primarily by adding properties and adding and overriding property values. The new properties and values are layered over the values assigned in the manifest, as described in “Repository Layers” on page 30. See “Service Bundles” on page 29 for more information about manifests and profiles. A profile is also an excellent tool for applying the same custom configuration to multiple systems, as described in “Creating SMF Profiles” on page 108.

Service information is stored in the *service configuration repository*, which is also called the *SMF database*. The service configuration repository stores the current state of each service instance on the system and the configuration data for each service and service instance. The data is stored in *layers* according to how values were modified, as described in “Repository Layers” on page 30.

SMF provides *actions* that you can invoke on a service instance, including enable, disable, refresh, and restart. Each service instance is managed by a *restarter*, which performs these administrative actions. In general, restarters perform actions by executing *methods* to move the service instance from one state to another state. For more information about restarters, see "Service Restarters" on page 26.

A *milestone service* is a special type of service that represents a level of system readiness such as a system `init` state. A milestone is a service that other service instances depend on to start. For example, run levels are represented by milestone services such as `svc:/milestone/multi-user-server`. Milestones also can be used to indicate the readiness of a group of services, such as `svc:/milestone/devices`, `svc:/milestone/network`, or `svc:/milestone/name-services`. The `svc:/milestone/goals` service defines the set of services that need to be running in order for the system to function as intended. See Chapter 7, "Creating a Service that Notifies if Conditions are not Satisfied" in *Developing System Services in Oracle Solaris 11.4*.

Service Models

SMF services are one of the following models:

Transient service

The service does some work and then exits without starting any long running processes.

Child or wait service

The service is restarted whenever its child process exits cleanly. A child process that exits cleanly is not treated as an error.

Contract or daemon service

The service starts a long running daemon or starts several related processes that are tied together as part of a *service contract*. The contract service manages processes that it starts and any dependent services and their start order. You only need to manage the high-level service.

Goal service

The service notifies administrators if any of its dependent services is not running. See Chapter 7, "Creating a Service that Notifies if Conditions are not Satisfied" in *Developing System Services in Oracle Solaris 11.4*.

Periodic services are a special case that do not fit any of these models. A *periodic* or *scheduled* service starts short-running processes at regular or scheduled intervals, staying online between runs when no associated contracted processes exist. See Chapter 3, "Creating a Service to Run Periodically" in *Developing System Services in Oracle Solaris 11.4* and Chapter 4, "Creating a Service to Run on a Specific Schedule" in *Developing System Services in Oracle Solaris 11.4* for more information.

Service Names

Each service and service instance is represented by a Fault Management Resource Identifier (FMRI). The full FMRI for a service instance has the following format:

`svc:/`*service_name*`:`*instance_name*

The *service_name* is a hierarchical name such as `network/dns/client` or `application/pkg/server`. Components of the *service_name* that precede the final forward slash character (/) are the *category* of the service. Categories such as `application`, `device`, `milestone`, `network`, and `system` help identify the purpose of the service.

The `site` category is reserved to help you avoid name conflicts when you create your own SMF services. For example, a site-specific service named `svc:/site/`*tool* will not conflict with an Oracle Solaris service named `svc:/`*tool*.

Service instance names are appended to the parent service name after a colon character. For example, `svc:/system/identity:node` and `svc:/system/identity:domain` are instances of the `svc:/system/identity` service.

In scripts, best practice is to use the full service instance name. Interactively, names can be shortened to the rightmost portions of the name that result in a unique name. For example, `svc:/system/identity` can be shortened to `identity`, and `svc:/system/identity:domain` can be shortened to `identity:domain`. Instance names must be preceded by some portion of the service name, followed by a colon character.

Service States

At any particular time, an SMF service instance is in one of the following states:

- `degraded` – The instance is running or available to run, but is functioning at a limited capacity.
- `disabled` – The instance is not enabled and is not running or available to run.
- `incomplete` – The instance is not completely installed on the system and has not been evaluated by its restarter. The instance is mentioned solely in a profile (and not explicitly marked complete in the profile), or is mentioned only as a dependent of another service.
- `maintenance` – The instance is enabled but not able to run. The instance might be transitioning through the `maintenance` state because an administrative action has not yet completed. Otherwise, administrative action is required to resolve the problem.

- `offline` – The instance is enabled but not running or available to run. For example, if the dependencies of an enabled service are not satisfied, the service is kept in the `offline` state. Failure of the start method can leave the instance in the degraded or maintenance state. Clearing the maintenance state (svcadm clear) places the instance in the uninitialized state so that the restarter can re-evaluate the instance.
- `online` – The instance is enabled and running or available to run. The `online` state is the expected operating state for a correctly configured service instance with all dependencies satisfied.
- `uninitialized` – This state is the initial state for all service instances, including newly created instances (through executing `svccfg add`, importing a manifest, or applying a profile). When an instance is cleared from the `maintenance` state (`svcadm clear`), it is placed in the `uninitialized` state so that its restarter can re-evaluate its configuration. Instances are moved to `maintenance`, `offline`, or `disabled` state after evaluation by the appropriate restarter. Note that evaluation of an instance can only occur if its restarter service is `online`.

A service instance transitions between states depending on conditions such as administrative actions or the state of its dependent services. For example, when you enable an instance that was in the `disabled` state, the newly-enabled instance first transitions into the `offline` state, and transitions into the `online` state when all of its dependencies are satisfied.

In addition to the current state, administrators can show the auxiliary state. Restarters (see “Service Restarters” on page 26) use the auxiliary state to store information about the state. The master restarter uses the auxiliary state to store the reason the instance transitioned to its current state. For example, after a transition to the `online` state, the value of the auxiliary state is typically `dependencies_satisfied`. The periodic restarter uses the auxiliary state to store whether the periodic task is currently running, as described in “Creating a Periodic Service” in *Developing System Services in Oracle Solaris 11.4*.

See the `smf`(7) man page for more information about these service states and about how service instances transition through these states.

Service Dependencies

A service can have a *dependency* on a service, a service instance, or a file. Service dependencies define relationships between services.

Dependency relationships determine when a service starts and automatically stops. When dependencies of an enabled service are not satisfied, the service is in the `offline` state. When dependencies of an enabled service are satisfied, the service is started. If the service start is successful, the service transitions to the `online` state.

Service dependencies are reevaluated as services transition through states. Service dependencies that are satisfied can later become not satisfied. File dependencies are evaluated only one time.

Dependencies can be required or optional. Service dependencies can be required to be running or disabled. A dependent service can be configured to restart or not when one of its service dependencies is stopped or refreshed.

Dependency relationships allow the following capabilities:

- Scalable and reproducible initialization processes
- Faster system startup on systems that have parallel capabilities by starting independent services in parallel
- Precise fault containment and fault recovery by restarting only services that are directly affected by a fault, and restarting those services in correct dependency order

Service Restarters

Each SMF service instance is managed by a *restarter*. The restarter retrieves instance configuration and provides an execution environment. See the `smf_restarter`(7) man page for information common to all restarters.

Master Restarter Daemon

The `svc.startd` daemon is the *master restarter* daemon for SMF and the *default restarter* for all service instances. The `svc.startd` daemon manages states for all service instances and their dependencies. As dependencies are satisfied when instances move to the online state, the master restarter invokes start methods of other instances or directs the delegated restarter to invoke the start method. The master restarter stops a service instance when the dependencies of the instance are no longer satisfied. The restarter attempts to restart an instance if the instance fails. Because an instance cannot be online until all of its dependencies are satisfied, the dependencies of an instance help determine the restart behavior of the instance. Properties set on each dependency declaration define whether that dependency is required and in what cases the instance will be restarted if the dependency is restarted.

Among other tasks, the `svc.startd` daemon starts the appropriate `/etc/rc*.d` scripts at the appropriate run levels, which is work that was previously done by `init`.

The following example shows that `svc.startd` is the restarter for the `network/ipmp:default` service instance. Other output has been omitted from this example.

```
$ svcs -l ipmp:default
restarter    svc:/system/svc/restarter:default
```

If the `restarter` property is empty or set to `svc:/system/svc/restarter:default`, the service instance is managed by `svc.startd`. For more information about the `svc.startd` daemon, see the `svc.startd`(8) man page.

Delegated Restarters

Some services have a set of common behaviors on startup. A *delegated restarter* can provide a specific execution environment and application-specific restarting behavior for these services. The delegated restarter specified by the `restarter` property is responsible for managing the service instance once that restarter is available.

Oracle Solaris includes the following delegated restarters:

`inetd`

The `inetd` delegated restarter can start Internet services on demand, rather than having the services always running. The `inetd` restarter provides its service instances with an environment composed of a network connection as input and output file descriptors. For more information about the `inetd` daemon, see the `inetd`(8) man page. The following example shows that `inetd` is the restarter for the `cups/in-lpd:default` service instance. Other output has been omitted from this example.

```
$ svcs -l cups/in-lpd:default
restarter    svc:/network/inetd:default
```

`svc.periodicd`

The periodic restarter daemon, `svc.periodicd`, is invoked automatically at system startup as part of the `svc:/system/svc/periodic-restarter` service and is automatically restarted if any failures occur. Services started by the periodic restarter remain online persistently but run their start method tasks only periodically or at scheduled times. Start method tasks of a periodic service are expected to run for a relatively brief period of time and then terminate. For more information about the periodic restarter, see the `svc.periodicd`(8) man page. For more information about periodic services, see Chapter 3, "Creating a Service to Run Periodically" in *Developing System Services in Oracle Solaris 11.4*.

`svc.zones`

Non-global zones are managed by the `svc:/system/zones:default` delegated restarter service. The following command shows that `svc:/system/zones:default` is the restarter for the `z1` non-global zone:

```
$ svcs -R svc:/system/zones:default
STATE          STIME    FMRI
online         12:11:12 svc:/system/zones/zone:z1
```

For more information about the zones restarter, see the `svc.zones`(8) man page. For more information about zones services, see *Creating and Using Oracle Solaris Zones*.

Service Properties and Property Groups

Information about services, including dependencies, methods, state, and application data, is stored in the service configuration repository as a set of *properties*. Properties can be defined on either the service or an instance of the service. Properties that are set on the service are inherited by all instances of that service. Properties that are set on an instance are used only by that instance. Service instances can customize the values of inherited properties and can define additional properties that are not defined for the parent service.

Properties are organized into *property groups*. Some common property groups include:

- `general` – Contains information such as whether the instance is enabled
- `restarter` – Contains runtime information that is stored by the restarter for the service, including the current state of the instance
- `start`, `refresh`, `stop` – Contains information such as which program to execute to start, refresh, or stop the service
- `config` – Used by service developers to hold application data.

A property group can be parented by a service or instance, or by another property group. A property group that is the child of another property group is also called a *nested property group*. Nested property groups more fully express interrelationships among configuration data.

See the `smf`(7) man page for more information about properties and property groups.

Service Configuration Repository

Information about each service is stored in the *service configuration repository*, which is also called the *SMF database*. The service configuration repository stores information as services, instances, property groups, and properties. In addition to information defined by service developers, the service configuration repository stores information such as the start time and current state of each service instance on the system.

The repository stores persistent configuration information as well as SMF runtime data for services.

- Persistent configuration information is stored in layers according to the source of the data. See "Repository Layers" on page 30.
- Runtime data, or non-persistent configuration information, is not preserved across reboot, and the repository does not store layer information for non-persistent data. Non-persistent data generally hold an active program state.

The repository also stores service template data, such as types, value constraints, and descriptions of properties. Template data is defined in the service manifest. See the `smf_template`(7) man page for more information about template data.

The service configuration repository can only be manipulated or queried by using SMF interfaces. Use the `svcs`, `svcprop`, `svcadm`, and `svccfg` commands or the Service Configuration Facility library functions listed in the `libscf`(3LIB) man page. You can read and write property values and show property values in specified layers and snapshots. For information about layers, see "Repository Layers" on page 30. For information about snapshots, see "Repository Snapshots" on page 32. You can show only the properties of the selected service instance or parent service, or you can show a *composed* view of properties. In a composed view, both properties set on the parent service and properties set on the service instance are shown; values shown are the values set on the service instance.

Service Bundles

A service bundle is an XML file that contains the information that is stored in the service configuration repository for a service or service instance. Information provided in service bundles is stored in the service configuration repository and can be exported from the repository. Service bundles in standard locations are imported into the repository during system boot.

The two types of service bundles are manifests and profiles.

Manifests	Manifests contain the complete set of properties associated with a specific set of services or service instances.
Profiles	Profiles typically provide customization of a service or service instance that augments or overrides information provided in the manifest. Examples of customizations include additional properties and changed property values.

The *standard location* for manifests is `/lib/svc/manifest`. The standard location for profiles is `/etc/svc/profile`.

When the system is booted or the manifest import service is restarted, manifests are imported and profiles are applied if they are new or changed. An IPS package that delivers a service bundle can specify that the manifest import service should be restarted when the package is installed.

Local customizations can be provided in profile files with an `.xml` suffix in the `/etc/svc/profile/site` directory. If the same property in the same repository layer for the same service or instance is defined by multiple manifests or profiles, SMF cannot determine which value to use. When this type of conflict is detected, the instance is placed in the maintenance state. See "Repository Layers" on page 30 for more information about layers.

In addition to delivering services into Oracle Solaris, service bundles can also deliver custom configuration across a variety of systems.

A system profile, `/etc/svc/profile/generic.xml`, is applied during installation. Do not change this profile. Any changes made to this system profile will be overwritten on upgrade. See the `smf_bootstrap`(7) man page for more information.

Repository Layers

The service configuration repository can store different values for a single property. The repository stores data in *layers* according to the source of the data. The source can be manifests, profiles, and customizations made by using SMF commands and library interfaces. You can view values in different layers to understand the source of the value that is in use in the running configuration: whether a value was assigned in the manifest, in a profile, or was changed by an administrator. See "Showing the Layer Where a Value Is Set" on page 59.

Configuration changes made by using SMF commands and library interfaces appear only in the `admin` layer. Configuration in other layers is defined in profile and manifest files in standard locations. When a property is added to the repository from a file or a property value is changed in a file, you can show the name of the file that provided that configuration by using the `-f` or `-o file` options of the `svccfg listprop` command. See "Showing the File that Contributed the Configuration" on page 60.

If a property has different values assigned at different layers, the value that is used by the service instance is the value in the highest layer of the layer hierarchy. The following table shows the order of layers in the hierarchy. For example, if a property has a value in the `node-profile` layer, that value overrides the value in the `manifest` layer or any other lower layer. If a property has a value in the `admin` layer, that value overrides all other values set in any other layer.

Layer	Content
`admin`	Any changes that are made by an administrator using the SMF commands or by an application using SMF library interfaces. The `admin` layer also includes any changes that

Layer	Content
	are made by importing a manifest or applying a profile from a non-standard location. See "Importing and Applying Manifests and Profiles" on page 100 for caution about the use of non-standard locations.
`sysconfig-profile`	Changes that are made by an administrator by using the `sysconfig` command interactively, and configuration that is specified in a profile that is installed by an Automated Installer (AI) client or by a zone. You are using the `sysconfig` command interactively when you use System Configuration Interactive (SCI) Tool. See "Creating Configuration Profiles" in *Customizing Automated Installations With Manifests and Profiles* for information about using SCI Tool. AI client profiles (created by `installadm create-profile` or `installadm update-profile`), zone profiles (created by `zoneadm install`, `zoneadm attach`, or `zoneadm clone`), and profiles created by using the `sysconfig` command interactively are stored in the `/etc/svc/profile/sysconfig` directory unless otherwise specified. If you specify the `-c` *dir* option with the `install`, `attach`, or `clone` subcommand of the `zoneadm` command or with the `sysconfig configure` command, and *dir* has subdirectories `enterprise`, `site`, or `node`, configuration from profiles in those directories will be set in the appropriate `enterprise-profile`, `site-profile`, or `node-profile` layer. When the `sysconfig` command is used interactively to configure the system, if property values are defined that are also defined in the `admin` layer, those property values are removed in the `admin` layer so that the new property values defined by using the `sysconfig` command are effective.
`node-profile`	Configuration that is specific to an Oracle Solaris instance. This configuration comes from SMF profiles in the `/etc/svc/profile/node` directory and from the legacy `/etc/svc/profile/site.xml` file. Do not add new configuration to the `/etc/svc/profile/site.xml` file.
`site-profile`	Configuration that is common to systems at the same location or site. This configuration comes from SMF profiles in the `/etc/svc/profile/site` directory. Note that `/var/svc/profile` is deprecated as a standard location. Do not put new profiles in the `/var/svc/profile` directory.
`enterprise-profile`	Configuration that applies to all of your Oracle Solaris systems. This configuration comes from SMF profiles in `/etc/svc/profile/enterprise`.
`system-profile`	Any values from the `/etc/svc/profile/generic.xml` and `/etc/svc/profile/platform.xml` system profiles as well as any profiles in the `/etc/svc/profile/system` directory.
`manifest`	Values from manifests in the `/lib/svc/manifest` and `/var/svc/manifest` directories. Do not put new service manifests in the `/var/svc/manifest` directory.

Configuration conflicts are not permitted within any layer. Configuration that is set by using SMF commands, the `sysconfig` command, or SMF library interfaces overwrites the previous setting. If conflicting configuration is delivered by multiple files in any single layer, and is not set at a higher layer, the `manifest-import` service log reports the conflict, and the service with the conflicting configuration is not started. See "Conflicting Configuration" on page 108 for more information.

You can specify the layer of configuration data to view and therefore identify which data are administrative customizations and which data were delivered with the software. When a client does not specify the layer from which to retrieve configuration data, the topmost layer data is provided. The topmost layer is determined by the order shown in the above table, where the `admin` layer is the topmost layer and the `manifest` layer is the lowest priority layer. If a property has a value in the `admin` layer, that is the value that the repository delivers. In this way, local customizations are preferred over the values that were provided when the system was installed.

Repository Snapshots

The repository captures a read-only *snapshot* of each service each time the service is successfully started. These snapshots enable you to easily return to a previous working state if necessary. The following snapshots might be available for any given instance:

`initial`	Initial configuration when the service and its instances were imported for the first time. An `initial` snapshot is not created if a profile starts the service or instance before manifest import.
`previous`	Current configuration captured when a manifest import is performed for a service that has already been delivered. The service could have already been delivered by the manifest being imported or by another manifest.
`running`	The running configuration of the service instance. When you change configuration data, use the `svcadm refresh` or `svccfg refresh` command to promote the new values to the running snapshot.
`start`	Configuration captured during a successful transition to the `online` state.

Repository Backups

SMF automatically takes the following *backups* of the service configuration repository:

- The `boot` backup is taken immediately before the first change to the repository is made during each system startup.
- The `manifest_import` backups occur before `svc:/system/early-manifest-import:default` or `svc:/system/manifest-import:default` completes, if the service imported any new manifests or ran any upgrade scripts.

Four backups of each type are maintained by the system, with the oldest backups deleted as necessary.

You can restore the repository from one of these backups. See "How to Restore a Repository From Backup" on page 123.

Configuration Files and SMF Services

SMF is the recommended mechanism to use to start applications. In most cases, SMF replaces the use of configuration files for managing services. This section describes how some common legacy configuration scripts and files are handled.

`/etc/rc?.d` scripts

The `/etc/rc?.d` directories, where `?` represents a run level, contain legacy initialization and termination scripts for managing services that execute on run level transitions. Most services that were formerly implemented by `/etc/rc?.d` scripts are managed by SMF. Some `/etc/rc?.d` scripts are retained to enable you to use third-party applications that expect these services as `/etc/rc*.d` scripts. These scripts are hard linked to files in the `/etc/init.d` directory. For information about `/etc/rc?.d` scripts and about run levels, see the `/etc/init.d/README` file, the `README` files in the `/etc/rc?.d` directories, and the `inittab`(5) man page. For instructions to convert a run control script, see "How to Convert a Run Control Script to an SMF Service" in *Developing System Services in Oracle Solaris 11.4*. After you convert an `rc?d` script, rename the script from `S`*script* to `s`*script* to effectively remove the script.

`/etc/init.d` scripts

The `/etc/init.d` directory contains initialization and termination scripts for changing `init` states. Some of these scripts are hard linked to scripts in the `/etc/rc?.d` directories. For information about `/etc/init.d` scripts, see `/etc/init.d/README` and the `init.d`(5) man page.

Legacy `init.d` run control scripts are represented with SMF FMRIs that begin with `lrc` instead of `svc`. The state of these `lrc` services is `legacy_run`. As shown in the following example, you can list names and start times of legacy services, but you cannot administer these services by using SMF.

```
$ svcs lrc:\*
STATE          STIME     FMRI
legacy_run     13:50:15  lrc:/etc/rc2_d/S89PRESERVE
$ svcs -l lrc:/etc/rc2_d/S89PRESERVE
svcs: Operation not supported for legacy service 'lrc:/etc/rc2_d/S89PRESERVE'
$ svccfg -s lrc:/etc/rc2_d/S89PRESERVE listprop
svccfg: Operation not supported for legacy service 'lrc:/etc/rc2_d/S89PRESERVE'
```

`/etc/inittab` entries

Entries in the `/etc/inittab` file control process dispatching by `init`. Do not edit the `/etc/inittab` file directly. Instead, modify SMF services. See "How to Modify a `ttymon` Property Value" on page 90 for an example of how to modify a parameter passed to `ttymon`.

For information about the format of `/etc/inittab` file entries, see the `inittab`(5) man page. For information about run levels, see the `inittab`(5) man page and `/etc/init.d/README`.

`/etc/inetd.conf` file

Services that were formerly configured by using the `inetd.conf` file are now configured by using SMF. Configurations in the `inetd.conf` file must be converted to SMF services to be available for use. See "Converting `inetd` Services to SMF Services" on page 130. For `inetd` services that are already converted to SMF services, see "Modifying Services that are Controlled by `inetd`" on page 101.

`/etc/nscd.conf` file
`/etc/nsswitch.conf` file
`/etc/resolv.conf` file

Do not edit these files. Edits will be lost. These files are automatically generated from SMF data for backward compatibility with applications that might parse the file. Use the `svccfg setprop` command to modify property values as shown in "Setting Property Values" on page 84.

The function of the `nscd.conf` file is replaced by the `svc:/system/name-service/cache` SMF service. See the `nscd.conf`(5) man page to see which `name-service-cache` properties to configure instead of editing the `nscd.conf` file.

The function of the `nsswitch.conf` file is replaced by the `svc:/system/name-service/switch` SMF service. See the `nsswitch.conf`(5) man page to see which `name-service/switch` properties to configure instead of editing the `nsswitch.conf` file.

The function of the `resolv.conf` file is replaced by the `svc:/network/dns/client` SMF service. See the `resolv.conf`(5) man page to see which `dns/client` properties to configure instead of editing the `resolv.conf` file.

These files are examples of configuration files that you should not edit. Other such files exist. In a few cases, editing a configuration file is the correct way to modify configuration, as described in "Modifying Services that are Configured by a File" on page 104. Before editing any configuration file, read any comments in the file and any associated man page to ensure that editing the file is the correct way to modify the configuration for the related service.

Service Management Privileges

Modifying service state and configuration requires increased privilege. Use one of the following methods to gain the privilege you need. See *Securing Users and Processes in Oracle Solaris 11.4* for more information about roles, profiles, and authorizations, including how to determine which role or profile you need and how to assign privileges.

Roles

Use the `roles` command to list the roles that are assigned to you. Use the `su` command with the name of the role to assume that role. As this role, you can execute any commands that are permitted by the rights profiles that are assigned to that role. For example, if the role is assigned the Service Configuration rights profile, you can execute the `svccfg` and `svcadm` commands modify service properties and change service state.

Rights profiles

Use the `profiles` command to list the rights profiles that are assigned to you. Use one of the following methods to execute commands that your rights profiles permit you to execute:

- Use a profile shell such as `pfbash` or `pfksh`.
- Use the `pfexec` command in front of the command that you want to execute. In general, you must specify the `pfexec` command with each privileged command that you execute.

Authorizations

See the `smf_security`(7) man page for detailed information about authorizations required for SMF operations. If the Service Configuration rights profile is not sufficient to manage a particular service, inspect the service for the following properties:

- The `action_authorization`, `modify_authorization`, `read_authorization`, and `value_authorization` properties specify required authorizations. Individual services can require their own particular authorizations.
- Properties of the `method` property group can specify requirements to run the method such as the user and privilege set.

sudo command

Depending on the security policy at your site, you might be able to use the `sudo` command with your user password to execute a privileged command.

♦ ♦ ♦ C H A P T E R 2

Getting Information About Services

This chapter shows how to get information about services such as the following:

- Service state, dependencies, and other property values
- Processes started by a contract service
- Log file location for troubleshooting
- FMA event and service transition event notification settings

Listing Services on the System

The `svcs` command is the primary command for listing service instance states and status.

Showing Service State

See “Service States” on page 24 for descriptions of the states shown in these examples.

EXAMPLE 1 Listing All Enabled Services

With no options or arguments, the `svcs` command lists all service instances that are enabled on this system, as well as instances that are temporarily disabled.

Service instances in the `disabled` state in this listing will be enabled on the next boot of the system. Instances in the `legacy_run` state are not managed by SMF. See “Configuration Files and SMF Services” on page 33 for more information about these legacy services. See “Getting More Information About Service States” on page 46 if you have services in the `maintenance`, `degraded`, or `offline` states.

The STIME column shows the time the instance entered the listed state. If the instance entered this state more than 24 hours ago, the STIME column shows the date. The following example is a partial listing.

```
$ svcs
STATE          STIME       FMRI
legacy_run     13:50:15    lrc:/etc/rc2_d/S89PRESERVE
disabled       13:49:11    svc:/platform/i86pc/acpihpd:default
online         13:48:41    svc:/system/early-manifest-import:default
online         13:48:41    svc:/system/svc/restarter:default
```

EXAMPLE 2 Listing All Installed Services

To list all service instances that are installed on this system, including `disabled` instances that will not be enabled automatically on next boot, use the `svcs -a` command.

```
$ svcs -a
```

An asterisk (`*`) is appended to the state for service instances that are transitioning from the listed state to another state. For example, `offline*` probably means the instance is still executing its start method.

A question mark (`?`) is displayed if the state is absent or unrecognized.

EXAMPLE 3 Listing All Instances of a Service

With a service name specified, the `svcs` command lists all instances of a service. See "Showing Selected Service Information" on page 40 for information about the `-o` option.

```
$ svcs -Ho inst identity
cert-expiry
domain
version
node
cert
```

Showing More Information About Services

The `svcs -l` command shows a long listing for each specified service instance including more detailed information about the instance state, paths to the log file and configuration files for the instance, dependency types, dependency restart attribute values, and dependency state.

The following example shows that the specified service instance is enabled, is online, and the service is a contract type service. See "Service Models" on page 23 for definitions of service types. If the state value has a trailing asterisk, for example `offline*`, then the instance is in transition, and the `next_state` field shows a state value instead of `none`. The `state_time` is the time the instance entered the listed state.

All of the required dependencies of this service instance are online. The one dependency that is disabled is an optional dependency. This example shows that the `net-snmp` service will be restarted if the `snmpd.conf` file is refreshed. For information about dependency types and restart attribute values, see "Showing Service Dependencies" on page 41. In `svcs -l` output, states other than those described in "Service States" on page 24 are possible for dependencies. See the `svcs`(1) man page for descriptions.

```
$ svcs -l net-snmp
fmri         svc:/application/management/net-snmp:default
name         net-snmp SNMP daemon
enabled      true
state        online
next_state   none
state_time   July  8, 2020 at  6:18:35 PM PDT
logfile      /var/svc/log/application-management-net-snmp:default.log
restarter    svc:/system/svc/restarter:default
contract_id  183
manifest     /lib/svc/manifest/application/management/net-snmp.xml
dependency   require_all/refresh file://localhost/etc/net-snmp/snmp/snmpd.conf (online)
dependency   require_all/restart svc:/system/cryptosvc (online)
dependency   require_all/none svc:/system/filesystem/local (online)
dependency   require_all/none svc:/milestone/multi-user (online)
dependency   optional_all/none svc:/milestone/name-services (online)
dependency   require_all/restart svc:/milestone/network (online)
dependency   optional_all/none svc:/network/rpc/rstat (disabled)
dependency   optional_all/none svc:/system/system-log (multiple)
```

EXAMPLE 4 Showing Processes Started by a Contract Service

Use the `svcs -p` command to show the process IDs and command names of processes started by a contract service instance. The `net-snmp` service manages the `/usr/sbin/snmpd` SNMP agent that collects information about a system through a set of Management Information Bases (MIBs).

```
$ svcs -p net-snmp
STATE          STIME          FMRI
online         18:18:35       svc:/application/management/net-snmp:default
               18:18:35            1507 snmpd
```

EXAMPLE 5 Showing a Contract Service Restarting Automatically After Process Stop

Contract service instances are automatically restarted if the contract empties. SMF also attempts to restart processes associated with a contract service instance as part of automatic recovery from hardware or software failure events. The following example shows that after the `/usr/`

sbin/snmpd process is killed, it is automatically restarted with a new process ID. The net-snmp:default instance is still online and has a new start time.

```
$ kill 1507
$ svcs -p net-snmp
STATE          STIME          FMRI
online         18:50:31       svc:/application/management/net-snmp:default
               18:50:31            1519 snmpd
```

Showing Selected Service Information

Output from the svcs command can be very useful for piping to other commands or using in scripts. The -o option of the svcs command enables you to specify the columns of information you want and the order of the columns. You can output the service name and instance name in separate columns, the current state and next state of the service, and the contract ID, for example. With the -s and -S options, you can specify the sort order of the output for one or more columns. See the COLUMNS section of the svcs(1) man page for a list of available columns. Multiple -s options behave additively.

Three additional pieces of information can be shown for periodic services. For information about periodic services, see Chapter 3, “Creating a Service to Run Periodically” in *Developing System Services in Oracle Solaris 11.4* and the svc.periodicd(8) man page.

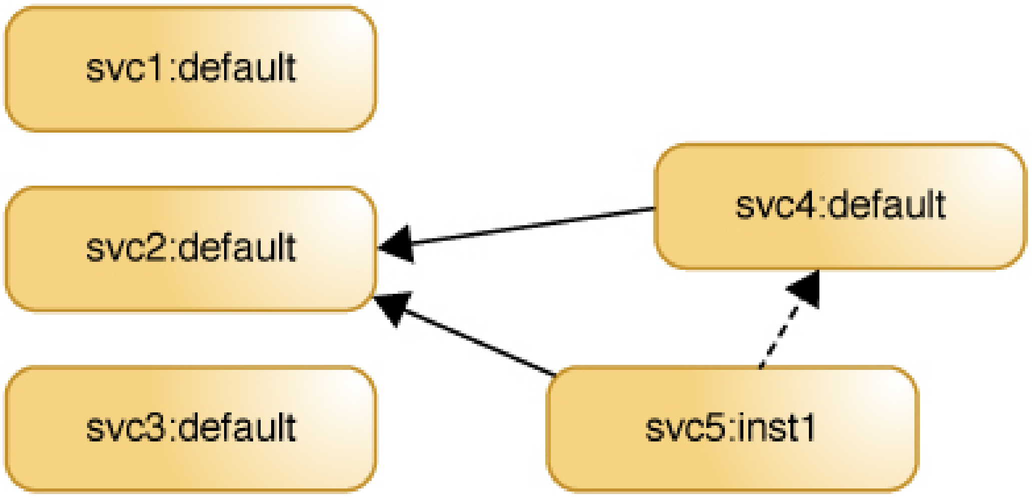
LRUN	The last time the start method of this periodic service instance ran. Service instances that are not periodic display a hyphen character (-) in this column.
NRUN	The next time the start method of this periodic service instance is scheduled to run. Service instances that are not periodic display a hyphen character (-) in this column.
ASTATE	The auxiliary state of the service instance. Service instances that are not periodic instances use this property to show the reason that the instance made its most recent transition. Instances that are not periodic instances almost always show dependencies_satisfied as the value of this property, since this is usually the reason the instance transitioned from the offline to the online state. Periodic service instances can display either running or scheduled in this column to indicate whether the instance is waiting between start method executions. See “Specifying the periodic_method Element” in *Developing System Services in Oracle Solaris 11.4* for more information.

Showing Service Dependencies

Dependency relationships govern service instance state transitions. See "Service Dependencies" on page 25 for a high-level description of dependencies. See *Developing System Services in Oracle Solaris 11.4* for detailed descriptions and how to specify different kinds of dependencies.

In the following figure, the `svc1:default`, `svc2:default`, and `svc3:default` service instances do not require any other services or any files or other resources to start. These instances can start in parallel, execute their start methods, and move to the online state without waiting on any other resources. The `svc4:default` instance cannot execute its start method until the `svc2:default` instance is online. The `svc5:inst1` instance needs both `svc2:default` resources and `svc4:default` resources. The dependency that `svc5:inst1` has on `svc4:default` is an optional dependency and is satisfied if `svc4:default` is in one of the following states: enabled and online, disabled, or not present. The `svc5:inst1` instance must wait until `svc2:default` is online, and if `svc4:default` is present and enabled, `svc5:inst1` must also wait until `svc4:default` is online. If `svc4:default` is present and disabled or is not present, `svc5:inst1` does not need to wait for `svc4:default`.

FIGURE 2 Service Dependency Relationships



Dependency Groupings

Each dependency is assigned to one of the following groupings. The grouping defines how dependencies in that grouping are satisfied.

`require_all`
: This dependency is satisfied when both of the following conditions are met:

  - All of the service dependencies in this grouping are running, either `online` or `degraded`.
  - All of the file dependencies in this grouping are present.

`require_any`
: This dependency is satisfied when either of the following conditions is met:

  - At least one of the service dependencies in this grouping is running, either `online` or `degraded`.
  - At least one of the file dependencies in this grouping is present.

`optional_all`
: This dependency is satisfied when all of the service dependencies in this group meet either of the following conditions:

  - The service is running, either `online` or `degraded`.
  - The service requires administrative action to run. The service is not present, is `incomplete`, is `disabled`, is in `maintenance`, or is `offline` waiting for dependencies that require administrative action to start.

  File dependencies in this group can be present or not present.

  This dependency is not satisfied if the service instance is in transition and does not require administrative intervention to start. In this case, the dependent service waits for this dependency to start or waits for the determination that the dependency cannot start without administrative action.

`exclude_all`
: This dependency is satisfied when both of the following conditions are met:

  - All of the service dependencies in this grouping are `disabled`, in `maintenance`, or not present.
  - All of the file dependencies in this grouping are not present.

Listing Instances That a Service Depends On

The `svcs -d` command lists the service instances that a given service depends on.

This example shows the service instances that the `system-repository` service depends on:

```
$ svcs -d system-repository
STATE          STIME          FMRI
online         13:49:26       svc:/milestone/network:default
online         13:49:58       svc:/system/filesystem/autofs:default
online         13:49:58       svc:/system/filesystem/local:default
```

The `svcs -l` command also lists the services that a given service depends on. In addition to the name and state of the dependency, the `-l` option output shows the type, or grouping, of the dependency and the value of the `restart_on` property of the dependency. In this example, two of the dependencies are required and one is optional. See "Dependency Groupings" on page 42 for descriptions of how dependencies in these groupings affect the dependent service. See "Showing Whether a Service Will Automatically Restart" on page 45 for descriptions of how different values of the `restart_on` property of the dependency affect the dependent service.

```
$ svcs -l system-repository
fmri         svc:/application/pkg/system-repository:default
name         IPS System Repository
enabled      true
state        online
next_state   none
state_time   July  8, 2020 at  3:51:34 PM PDT
logfile      /var/svc/log/application-pkg-system-repository:default.log
restarter    svc:/system/svc/restarter:default
contract_id  181
manifest     /lib/svc/manifest/application/pkg/pkg-system-repository.xml
dependency   optional_all/error svc:/system/filesystem/autofs:default (online)
dependency   require_all/none svc:/system/filesystem/local:default (online)
dependency   require_all/error svc:/milestone/network:default (online)
```

You can also use the `svcprop` command to list these dependencies. This form shows the grouping and `restart_on` values of the dependency on separate lines, and does not show the state of the dependency.

```
$ svcprop -g dependency system-repository:default
autofs/entities fmri svc:/system/filesystem/autofs:default
autofs/grouping astring optional_all
autofs/restart_on astring error
autofs/type astring service
```

```
filesystem-local/entities fmri svc:/system/filesystem/local:default
filesystem-local/grouping astring require_all
filesystem-local/restart_on astring none
filesystem-local/type astring service
network/entities fmri svc:/milestone/network:default
network/grouping astring require_all
network/restart_on astring error
network/type astring service
```

Listing Instances That Depend on a Service

The `svcs -D` command lists the service instances that depend on a given service.

This example shows the service instances that depend on the `system-repository` service:

```
$ svcs -D system-repository
STATE          STIME           FMRI
disabled       13:48:46        svc:/application/pkg/zones-proxyd:default
```

The following command confirms that `zones-proxyd` depends on `system-repository`.

```
$ svcs -do svc,desc zones-proxyd
SVC                                DESC
application/pkg/system-repository IPS System Repository
system/filesystem/minimal         minimal file system mounts
milestone/network                 Network milestone
```

The following command shows more information about how `zones-proxyd` depends on `system-repository`. The last line of this output shows that the `zones-proxyd` service requires the `system-repository` service to be running and shows that `system-repository` is currently running.

```
$ svcs -l zones-proxyd
fmri         svc:/application/pkg/zones-proxyd:default
name         Zones Proxy Daemon
enabled      true
state        online
next_state   none
state_time   July  8, 2020 at  4:13:02 PM PDT
logfile      /var/svc/log/application-pkg-zones-proxyd:default.log
restarter    svc:/system/svc/restarter:default
contract_id  182
manifest     /lib/svc/manifest/application/pkg/zoneproxyd.xml
dependency   require_any/none svc:/system/filesystem/minimal (online)
dependency   require_any/error svc:/milestone/network (online)
dependency   require_all/none svc:/application/pkg/system-repository (online)
```

This output also shows that the `zones-proxyd` service will be restarted if the `milestone/network` service stops due to error.

Showing Whether a Service Will Automatically Restart

A running service can be configured to restart when one of its dependencies is stopped or refreshed. If dependencies of a running service (`online` or `degraded` state) are not satisfied, the service transitions to the `offline` state. If a service restarts after a dependency stop or refresh, dependencies might again be satisfied and the dependent service transitioned back to a running state.

The following factors determine whether a service is restarted after a `require_all`, `require_any`, or `optional_all` dependency is stopped or refreshed:

- Whether the dependency was stopped or refreshed. If stopped, whether the dependency was stopped because of an error such as a hardware error or a core dump or for some other reason such as an administrative action.
- The value of the `restart_on` attribute of the dependency. Possible values are `none`, `error`, `restart`, and `refresh`.

As shown in the following table, if the value of the `restart_on` attribute of the dependency is `none`, the dependent service is not restarted when the dependency is stopped or refreshed. If the value of the `restart_on` attribute of the dependency is `refresh`, the dependent service is always restarted when the dependency is stopped or refreshed. If the value of `restart_on` is `error`, the dependent service is only restarted if the dependency stopped because of an error. If the value of `restart_on` is `restart`, the dependent service is only restarted if the dependency was refreshed.

TABLE 1 Automatically Restarting a Service After a Dependency Stop

`require_all`, `require_any`, or `optional_all` **Dependency**
Stop or Refresh Event
	`restart_on=`
	`none`
		`restart_on=`
		`error`
			`restart_on=`
			`restart`
				`restart_on=`
				`refresh`

Stop due to error	No restart	Restart	No restart	Restart
Other stop	No restart	No restart	No restart	Restart
Refresh	No restart	No restart	Restart	Restart

"Listing Instances That a Service Depends On" on page 43 shows that the system-repository service has two require_all dependencies and one optional_all dependency. The following command shows that the system-repository service will be restarted if the milestone/network service or the system/filesystem/autofs service stops due to an error but not if they stop for any other reason or are refreshed. The system-repository service will not be restarted if the system/filesystem/local service is refreshed or stopped for any reason.

```
$ svcprop -p restart_on -g dependency system-repository
autofs/restart_on astring error
filesystem-local/restart_on astring none
network/restart_on astring error
```

Getting More Information About Service States

With no arguments, the svcs -x command gives explanatory information about the following service instances:

- Instances that are enabled but not running.
- Instances that are preventing other enabled services from running.

If all enabled services are running, the svcs -x command produces no output.

In the following example, the pkg/depot service is in the maintenance state because its start method failed.

```
$ svcs -x
svc:/application/pkg/depot:default (IPS Depot)
 State: maintenance since September 11, 2013 01:30:42 PM PDT
Reason: Start method exited with $SMF_EXIT_ERR_FATAL.
   See: http://support.oracle.com/msg/SMF-8000-KS
   See: pkg.depot-config(8)
   See: /var/svc/log/application-pkg-depot:default.log
Impact: This service is not running.
```

The output suggests a Predictive Self-Healing knowledge article from My Oracle Support, a man page, and a log file to reference to determine why the start method failed. See "Viewing Service Log Files" on page 47 for information about different ways to view log files. See "Repairing an Instance That Is Degraded, Offline, or in Maintenance" on page 117 for information about how to fix a service that is in the maintenance state.

In the following example, the print/server service has dependent services that are not running. The dependent services cannot run because the print/server service has been disabled.

```
$ svcs -x
svc:/application/print/server:default (LP print server)
 State: disabled since Fri Mar 08 14:42:32 2013
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: lpsched(8)
Impact: 2 dependent services are not running.  (Use -v for list.)
$ svcs -xv
svc:/application/print/server:default (LP print server)
 State: disabled since Fri Mar 08 14:42:32 2013
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: man -M /usr/share/man -s 8 lpsched
Impact: 2 dependent services are not running:
        svc:/application/print/rfc1179:default
        svc:/application/print/ipp-listener:default
$ svcs -D print/server
STATE          STIME    FMRI
online         Mar_08   svc:/milestone/multi-user:default
offline        Mar_08   svc:/application/print/ipp-listener:default
offline        Mar_08   svc:/application/print/rfc1179:default
```

If an argument given to the `svcs -x` command does not meet the criteria stated at the beginning of this section, the output does not show any reason for the instance state but still shows resources for more information.

```
$ svcs -x smb
svc:/network/smb:default (SMB properties)
 State: online since July  8, 2020 at  1:48:45 PM PDT
   See: smb(5)
   See: /var/svc/log/network-smb:default.log
Impact: None.
```

Viewing Service Log Files

SMF records information about significant restarter actions, method standard output, and standard error output to `/var/svc/log/`*service*`:`*instance*`.log` for each service instance. Hyphens are substituted for forward slashes in the *service* name in the log file name. The `svcs` command with the `-L`, `-l`, or `-x` option shows the full path name of the log file for the specified service instance. The `svcs -xL` command shows the last few lines of the log file and tells you to use the `svcs -Lv` command to view the complete log file. The `svcs -Lv` command displays the complete file, which could be quite long. If you prefer to view the log file in an editor or view just the last *n* entries, for example, operate on the output of the `svcs -L` command.

The following example shows how to use the log file to investigate why the service shown in the `svcs` listing is temporarily disabled.

```
$ svcs
STATE          STIME        FMRI
legacy_run     13:50:15     lrc:/etc/rc2_d/S89PRESERVE
disabled       13:49:11     svc:/platform/i86pc/acpihpd:default
online         13:48:41     svc:/system/early-manifest-import:default
online         13:48:41     svc:/system/svc/restarter:default
$ svcs -x acpihpd
svc:/platform/i86pc/acpihpd:default (Intel ACPI hot-plug daemon)
 State: disabled since July  8, 2020 at  1:49:11 PM PDT
Reason: Temporarily disabled by service method: "no acpidr device was found on this
 system."
   See: http://support.oracle.com/msg/SMF-8000-1S
   See: acpihpd(8)
   See: /var/svc/log/platform-i86pc-acpihpd:default.log
Impact: This service is not running.
$ svcs -xL acpihpd
svc:/platform/i86pc/acpihpd:default (Intel ACPI hot-plug daemon)
 State: disabled since July  8, 2020 at  1:49:11 PM PDT
Reason: Temporarily disabled by service method: "no acpidr device was found on this
 system."
   See: http://support.oracle.com/msg/SMF-8000-1S
   See: acpihpd(8)
   See: /var/svc/log/platform-i86pc-acpihpd:default.log
Impact: This service is not running.
   Log:
[ 2020 Mar  3 09:43:45 Enabled. ]
[ 2020 Mar  3 09:44:09 Executing start method ("/lib/svc/method/svc-acpihpd"). ]
[ 2020 Mar  3 09:44:09 Method "start" exited with status 101. ]
[ 2020 Mar  3 09:44:09 "start" method requested temporary disable: "no acpidr device was
 found on this system"
 ]
[ 2020 Jul  8 13:48:47 Enabled. ]
[ 2020 Jul  8 13:49:11 Executing start method ("/lib/svc/method/svc-acpihpd"). ]
[ 2020 Jul  8 13:49:11 Method "start" exited with status 101. ]
[ 2020 Jul  8 13:49:11 "start" method requested temporary disable: "no acpidr device was
 found on this system"
 ]

   Use: 'svcs -Lv svc:/platform/i86pc/acpihpd:default' to view the complete log.
$ svcs -L acpihpd
/var/svc/log/platform-i86pc-acpihpd:default.log
$ view `svcs -L acpihpd`
```

Other log files that you might find useful include the log for the master restarter daemon and the system log. To see the log file name and view the log file for the `svc.startd` restarter daemon, use the service name `restarter` with the `svcs` command. To see the log file name and view the log file for the `syslogd` system log daemon, use the service name `system-log`.

```
$ svcs -L restarter
/var/svc/log/svc.startd.log
$ svcs -L system-log
/var/svc/log/system-system-log:default.log
/var/svc/log/system-system-log:rsyslog.log
```

See "Specifying the Amount of Startup Messaging" on page 126 for instructions to change the amount of messaging you see on system boot. See "Configuring Notification of State Transition and FMA Events" on page 77 for instructions to configure services to notify you when they transition into or out of a service state or when an FMA event occurs.

Inspecting Service Configuration

Service configuration is expressed in properties that are set on services and service instances and stored in layers in the service configuration repository. Properties that are set on a service are inherited by all instances of that service. Properties that are set on an instance are used only by that instance. Service instances can customize the values of inherited properties and can define additional properties that are not defined for the parent service.

This section shows how to retrieve property values and how to identify whether the value is global for the service, is specific to an instance, was delivered with the software, or is an administrative customization.

Showing Descriptions of Properties and Property Groups

The `svccfg describe` command displays a description of the property groups and properties of a service, including the current values of properties. With no operands, `describe` shows descriptions of all property groups and properties of the selected service or service instance. Use the `-v` option to show more information, including a description of the current value and a list of possible values. Use the `-t` option to show template information.

```
$ svccfg -s pkg/server describe network/restart_on
network/restart_on astring     none
    Determines whether to restart the service due to a dependency refresh, restart, or
 failure.
```

```
$ svccfg -s pkg/server describe -v network/restart_on
network/restart_on astring     none
    type: astring
    required: true
    Determines whether to restart the service due to a dependency refresh, restart, or
 failure.
    visibility: readwrite
    minimum number of values: 1
    maximum number of values: 1
  value: none
    value description: Never restart due to dependency refresh, restart, or failure.
  value constraints:
    value name: none
    value name: error
    value name: restart
    value name: refresh
  value choices:
    value name: none
    value name: error
    value name: restart
    value name: refresh
```

Showing Service and Instance Property Values

The examples in this section describe how to view service and instance properties and property groups in different views, layers, and snapshots. Note the following definitions. For descriptions of layers and snapshots, see "Service Configuration Repository" on page 28.

committed properties	Property values that are currently effective. The `svcprop` command shows committed properties by default. Use the `svcprop` command with the `-c` option to show any property values that are not yet committed.
composed view	Properties that are inherited from the parent service and properties that are defined on the instance. If a property that is defined in the parent service is customized in the instance, the customized value is shown. The `svcprop` command shows the composed view by default.
directly attached properties	Property values that are not committed. The property has been set on the instance or on the parent service but the instance has not been refreshed. When you refresh the instance, a running snapshot is taken and the property becomes committed. Use the `svcprop` command with the `-c` option or the `-C` option to view directly attached properties.
editing view	The most recent property changes that are not committed.

EXAMPLE 6 Listing Instance and Inherited Properties Currently in Use

By default, the `svcprop` command shows the values assigned to properties in the running snapshot, which are the values currently being used. By default, the `svcprop` command shows properties in the *composed view* of the running snapshot, which means that both instance-specific properties and inherited properties are shown. If the value of an inherited property is customized in the instance, the value set in the instance is shown. The output lists one line for each property, showing the property group and property name separated by a forward slash character, the data type of the property value, and the property value. If no property or group name is specified, all property values in the running snapshot are shown.

If the FMRI or pattern operand specifies the full service name but does not specify an instance, properties set only on the service are shown. Properties set only on an instance are not shown. The following command shows properties such as service dependencies, the type of the service, and the paths of the profile and manifest files.

```
$ svcprop svc:/system/identity
```

When you specify an instance, you see the composed view of properties customized for that instance and properties inherited from the parent service. The following command lists all the properties in the running snapshot for the specified instance, including properties inherited from the parent service and properties specific to this instance. For inherited properties whose value is customized for this instance, the customized value is shown. This example shows properties such as additional dependencies, the path to the executable that starts this instance, the path to the log file for this instance, and information about the state of this instance.

```
$ svcprop svc:/system/identity:domain
```

See also the use of the `-c` option in Example 8, “Listing Service and Instance Values in the Editing View,” on page 52.

EXAMPLE 7 Listing Specified Properties or Property Groups Currently in Use

Use the `-p` option to show specific properties or all properties in a specific property group. The following example shows the value of a specific property:

```
$ svcprop -p config/prop1 example
svc:/site/example:default/:properties/config/prop1 count 1
svc:/site/example:newinst/:properties/config/prop1 count 1
```

You can specify the `-p` option multiple times:

```
$ svcprop -p config/prop1 -p config/prop2 example
svc:/site/example:default/:properties/config/prop1 count 1
svc:/site/example:default/:properties/config/prop2 count 2
```

```
svc:/site/example:newinst/:properties/config/prop1 count 1
svc:/site/example:newinst/:properties/config/prop2 count 2
```

The following example shows the values of all properties in a specific property group:

```
$ svcprop -p config example
svc:/site/example:default/:properties/config/prop1 count 1
svc:/site/example:default/:properties/config/prop2 count 2
svc:/site/example:newinst/:properties/config/prop3 count 3
svc:/site/example:newinst/:properties/config/prop1 count 1
svc:/site/example:newinst/:properties/config/prop2 count 2
```

The following commands show that `config/prop1` and `config/prop2` are defined for the `example` parent service and not for either of the service instances. The preceding `svcprop` command shows all three properties because `svcprop` shows the composed view: properties defined for an instance plus inherited properties. The `svccfg` command shows only properties for the specified service instance or parent service.

```
$ svccfg -s example listprop config
config       application
config/prop1 count       1
config/prop2 count       2
$ svccfg -s example:default listprop config
$ svccfg -s example:newinst listprop config
config       application
config/prop3 count       3
```

EXAMPLE 8 Listing Service and Instance Values in the Editing View

With the `-C` and `-c` options, the `svcprop` command shows the *editing* view instead of the running snapshot. The editing view shows the most recent changes. The changes in the editing view might or might not have been committed into the running snapshot by refreshing the instance. The following commands illustrate the difference between the running snapshot and the editing view.

The following command changes the value of a property of the `example:default` service instance:

```
$ svccfg -s example:default setprop config/prop1 = 11
```

The change does not show because the instance has not been refreshed:

```
$ svcprop -p config example
svc:/site/example:default/:properties/config/prop1 count 1
svc:/site/example:default/:properties/config/prop2 count 2
svc:/site/example:newinst/:properties/config/prop3 count 3
svc:/site/example:newinst/:properties/config/prop1 count 1
```

```
svc:/site/example:newinst/:properties/config/prop2 count 2
```

The `-C` option shows the new value in the editing view:

```
$ svcprop -C -p config example
svc:/site/example:default/:properties/config/prop1 count 11
svc:/site/example:newinst/:properties/config/prop3 count 3
```

The `-C` option shows properties that are directly attached to an instance, without composition. The newest values of instance properties are shown, and inherited properties are not shown.

The `-c` option shows the composed view of directly attached properties. The newest values of both instance properties and inherited properties are shown.

```
$ svcprop -c -p config example
svc:/site/example:default/:properties/config/prop1 count 11
svc:/site/example/:properties/config/prop2 count 2
svc:/site/example:newinst/:properties/config/prop3 count 3
svc:/site/example/:properties/config/prop1 count 1
svc:/site/example/:properties/config/prop2 count 2
```

The following command changes the value of a property of the `example` parent service:

```
$ svccfg -s example setprop config/prop2 = 22
```

This change to the parent service property is not shown by the `-C` option because the `-C` option shows properties that are directly attached to an instance:

```
$ svcprop -C -p config example
svc:/site/example:default/:properties/config/prop1 count 11
svc:/site/example:newinst/:properties/config/prop3 count 3
```

This change to the parent service property is shown by the `-c` option because the `-c` option shows the composed view:

```
$ svcprop -c -p config example
svc:/site/example:default/:properties/config/prop1 count 11
svc:/site/example/:properties/config/prop2 count 22
svc:/site/example:newinst/:properties/config/prop3 count 3
svc:/site/example/:properties/config/prop1 count 1
svc:/site/example/:properties/config/prop2 count 22
```

In the following example, new values are shown for the `example:default` instance because that instance is refreshed. The `example:newinst` instance must also be refreshed to make the new `config/prop2` value the effective value for that instance.

```
$ svcadm refresh example:default
$ svcprop -p config example
svc:/site/example:default/:properties/config/prop1 count 11
svc:/site/example:default/:properties/config/prop2 count 22
svc:/site/example:newinst/:properties/config/prop3 count 3
svc:/site/example:newinst/:properties/config/prop1 count 1
svc:/site/example:newinst/:properties/config/prop2 count 2
```

The `svccfg` command displays the editing property values by default, not the values in the running snapshot. You can force `svccfg` to display values in the running snapshot by using the `selectsnap` subcommand as shown in "Showing Values in a Specified Snapshot" on page 61.

The `svccfg` command only shows values for the parent service when you specify a parent service and only shows values for an instance when you specify an instance. If you receive no output from the `svccfg listprop` command, the property you specified might not be set on the parent service or the instance that you specified. If the property was deleted, use `listcust -M` to view the masked value, as shown in "Showing Configuration Customizations" on page 61.

The following command lists all editing property values for the specified service because no property group or property name is specified. In addition to the output shown by the `svcprop svc:/system/identity` command, this output includes property group names and types and template data.

```
$ svccfg -s svc:/system/identity listprop
```

The following command lists all editing property values for the specified service instance. Because this command does not show the composed view, this output does not show the paths to the profile and manifest files, for example.

```
$ svccfg -s svc:/system/identity:domain listprop
```

EXAMPLE 9 Listing Specified Properties or Property Groups in the Editing View

The following command lists all editing property values in the specified property group for the specified service instance. The `-o` option enables you to select the columns to display. See the `svccfg`(8) man page for the list of valid column names.

```
$ svccfg -s pkg/server:s11 listprop pkg
pkg                  application
pkg/inst_root       astring      /var/share/pkg/repositories/solaris
pkg/port            count        81
$ svccfg -s pkg/server:s11 listprop -o propname,value pkg
inst_root      /var/share/pkg/repositories/solaris
port           81
```

EXAMPLE 10 Listing Properties Whose Names Include Special Characters

"Naming Property Groups and Properties" in *Developing System Services in Oracle Solaris 11.4* provides a list of reserved characters that can be used in property group and property names. These reserved characters appear encoded in FMRIs:

```
$ svcprop -p config enchars_example:default
config/%25%20increase count 10
config/maximum%20%23 count 9
config/start%3Aend count 10
config/students%2Fteachers count 20
```

To query property groups or properties whose names contain reserved characters, copy the encoded names or use the `-G` and `-P` options:

```
$ svcprop -p config/%25%20increase enchars_example:default
10
$ svcprop -G config -P students/teachers enchars_example:default
20
$ svccfg -s enchars_example:default listprop -G config -P "maximum #"
config/maximum%20%23 count       9
```

Showing Properties in a Nested Property Group

To list properties or property values in a property group that is the child of another property group, specify the full ancestry of the property.

In the following example, `http` and `https` are child property groups of the `config` property group, and `ssl` is a child property group of the `https` property group:

```
$ svccfg -s npg_example listprop config
config                    application
config/port               count       80
config/http               application
config/http/port          count       80
config/https              application
config/https/port         count       443
config/https/ssl          application
config/https/ssl/certfile astring     cert.crt
config/https/ssl/keyfile  astring     key.crt
```

As in preceding examples, specifying only a top-level property group to the `svcprop` command shows only properties directly attached to that property group. Properties for child property groups are not shown.

```
$ svcprop -p config npg_example
config/port count 80
```

EXAMPLE 11 Listing Properties in a Child Property Group

If you use the `-p` option of the `svcprop` command to view properties in a nested property group, you must include all the parent property groups. The following command lists all properties of the `config/https/ssl` property group:

```
$ svcprop -p config/https/ssl svc:/site/npg_example
config/https/ssl/certfile astring cert.crt
config/https/ssl/keyfile astring key.crt
```

If you use the `-G` option of the `svcprop` command to view properties in a nested property group, specify a separate `-G` option for each property group.

```
$ svcprop -G config npg_example
config/port count 80
$ svcprop -G config -G https npg_example
config/https/port count 443
$ svcprop -G config -G https -G ssl npg_example
config/https/ssl/certfile astring cert.crt
config/https/ssl/keyfile astring key.crt
```

As shown at the beginning of this section, the `svccfg listprop` command shows properties in all child property groups of the specified property group.

```
$ svccfg -s npg_example listprop config/https
config/https              application
config/https/port         count       443
config/https/ssl          application
config/https/ssl/certfile astring     cert.crt
config/https/ssl/keyfile  astring     key.crt
```

EXAMPLE 12 Using a Wildcard in Place of Child Property Group Names

Wildcards can be used to replace child property group names in the `svccfg listprop` command.

```
$ svccfg -s npg_example listprop config/*port
config/port       count       80
config/http/port  count       80
config/https/port count       443
```

Wildcards cannot be used to replace child property group names in the `svcprop` command.

```
$ svcprop -p config/*port npg_example
```

```
svcprop: Couldn't find property group 'config/*port' for instance
'svc:/site/npg_example:default'.
```

EXAMPLE 13 Showing Property Values in a Child Property Group

If you use the `svcprop` command, specify all parent property group names.

```
$ svcprop -p config/https/ssl/certfile npg_example
cert.crt
```

If you use the `-G` option with the `svcprop` command, then you must use the `-P` option to specify the property name.

```
$ svcprop -G config -G https -G ssl -P certfile npg_example
cert.crt
```

If you use the `svccfg` command, you can specify all parent property group names or you can use a wildcard.

```
$ svccfg -s npg_example listprop config/https/ssl/certfile
config/https/ssl/certfile astring     cert.crt
$ svccfg -s npg_example listprop config/*file
config/https/ssl/certfile astring     cert.crt
config/https/ssl/keyfile  astring     key.crt
```

Showing Properties in a Property Group Type

In addition to showing property values by property name or property group name, you can also show property values by property group type.

EXAMPLE 14 Showing Property Groups and Their Types

The `listpg` subcommand of the `svccfg` command shows the name and type of each property group.

```
$ svccfg -s pkg/server listpg
autofs         dependency
fs             dependency
general        framework
manifestfiles  framework
network        dependency
ntp            dependency
```

```
pkg            application
pkg_bui        application
pkg_secure     application
start          method
stop           method
tm_common_name template
$ svccfg -s pkg/server:s11 listpg
general           framework
restarter         framework             NONPERSISTENT
restarter_actions framework             NONPERSISTENT
```

Non-persistent property groups generally hold an active program state. Values of properties in non-persistent property groups are cleared during system boot.

Specify a property group name to show the type of only that property group.

```
$ svccfg -s pkg/mirror listpg config
config  application
```

EXAMPLE 15 Listing Properties of a Property Group Type

Use the `-g` option of the `svcprop` command to show properties in a specific property group type.

```
$ svcprop -g com.sun,fw_configuration smtp
firewall_config/apply_to astring ""
firewall_config/exceptions astring ""
firewall_config/policy astring use_global
firewall_config/value_authorization astring solaris.smf.value.firewall.config
```

Multiple `-g` options show properties from property groups of all specified types.

```
$ svcprop -g application -g com.sun,fw_configuration smtp
config/db_version integer 5
config/include_info boolean false
config/local_only boolean true
config/path_to_sendmail_mc astring ""
config/value_authorization astring solaris.smf.value.sendmail
firewall_config/apply_to astring ""
firewall_config/exceptions astring ""
firewall_config/policy astring use_global
firewall_config/value_authorization astring solaris.smf.value.firewall.config
```

If you use both the `-p` and `-g` options, do not specify the name of the property group in the `-p` option value.

```
$ svcprop -g plugin -p path auditd
audit_binfile/path astring audit_binfile.so
```

```
audit_remote/path astring audit_remote.so
audit_sstore/path astring audit_sstore.so.1
audit_syslog/path astring audit_syslog.so
```

Showing the Layer Where a Value Is Set

The service configuration repository stores property data in layers according to the source of the data. Both the `svcprop` and `svccfg` commands can show the layer that is the source of a property value. The `-l` option of the `svcprop` and `svccfg` commands requires an argument to specify the layer for which you want information. Argument values are `manifest`, `system-profile`, `enterprise-profile`, `site-profile`, `node-profile`, `sysconfig-profile`, and `admin`. The output indicates whether a specific property value was set in the service manifest, in a profile, or by an administrator or application. See "Repository Layers" on page 30 for descriptions of the layers. The keyword `all` is an alias for all layers. If the layer you specify is not the source of the property values you request, no output is shown.

The following command shows that some property values come from the service manifest and some were set by an administrator. Some properties have values in more than one layer. The `pkg/readonly` property has a value set in the service manifest, and an administrator also set that same value. Values from different layers could be different.

```
$ svcprop -l all -p pkg pkg/server:s11
pkg/port count admin 81
pkg/inst_root astring admin /var/share/pkg/repositories/solaris
pkg/address net_address manifest
pkg/cfg_file astring manifest ""
...
pkg/readonly boolean manifest true
pkg/readonly boolean admin true
...
```

The `-l` option of the `svccfg listprop` command can also take the argument `current`. Using `current` as the `-l` argument shows the same property values that are shown when you do not use the `-l` option. The only difference in the output is that the name of the layer is also shown. The non-persistent data does not show a layer name (the third column displays `<none>`) because the service configuration repository does not store layer information for non-persistent data. Non-persistent property groups generally hold an active program state, and values of properties in non-persistent property groups are cleared during system boot.

```
$ svccfg -s pkg/server:s11 listprop -l current
pkg                                application admin
pkg/inst_root                     astring     admin     /var/share/pkg/repositories/
solaris
pkg/port                          count       admin     81
```

```
general                          framework   manifest
general/complete                 astring     manifest
general/enabled                  boolean     admin    true
restarter                        framework   <none>       NONPERSISTENT
restarter/auxiliary_state        astring      <none>   dependencies_satisfied
restarter/contract               count        <none>   121
restarter/logfile                astring      <none>   /var/svc/log/application-pkg-
server:default.log
restarter/next_state             astring      <none>   none
restarter/start_method_timestamp  time        <none>   1594325294.273843000
restarter/start_method_waitstatus integer     <none>   0
restarter/start_pid              count        <none>   1055
restarter/state                  astring      <none>   online
restarter/state_timestamp        time         <none>   1594325294.286321000
restarter_actions                framework   <none>       NONPERSISTENT
restarter_actions/auxiliary_fmri  astring     <none>   svc:/network/ssh:default
restarter_actions/auxiliary_tty   boolean     <none>   true
restarter_actions/enable_complete time        <none>   1594325294.291642000
```

Showing the File that Contributed the Configuration

The following commands show that the `localtime` property is set to UTC in the service manifest and set to US/Pacific in a profile in the `/etc/svc/profile/node` directory. The value set at the `node-profile` layer overrides the value set at the `manifest` layer.

```
$ svcprop -l all -p timezone/localtime system/timezone:default
timezone/localtime astring manifest UTC
timezone/localtime astring node-profile US/Pacific
$ svccfg -s system/timezone:default listprop -l all -o propname,layer,value \
> timezone/localtime
localtime          node-profile   US/Pacific
localtime          manifest       UTC
```

Use either the `-f` option or the `-o file` option of the `svccfg listprop` command to show the name of the file that contributed the configuration:

```
$ svccfg -s system/timezone:default listprop -l all -f timezone/localtime
localtime          /etc/svc/profile/node/migrated_etc_svc_profile_site_sc_profile.xml
 US/Pacific
localtime          /lib/svc/manifest/system/timezone.xml UTC
$ svccfg -s system/timezone:default listprop -l all -o propname,value,file \
> timezone/localtime
localtime          US/Pacific /etc/svc/profile/node/
migrated_etc_svc_profile_site_sc_profile.xml
```

```
localtime          UTC /lib/svc/manifest/system/timezone.xml
```

Showing Values in a Specified Snapshot

The following command lists the snapshots that are available for this service instance. Use these snapshot names with either `svcprop` or `svccfg` to show the values of properties that were set in that snapshot. Only instances have snapshots. Services do not have snapshots. See "Repository Snapshots" on page 32 for information about snapshots of the service configuration repository.

```
$ svccfg -s pkg/server:default listsnap
initial
previous
running
start
$ svccfg -s pkg/server:s11 listsnap
previous
running
start
```

The following commands show that the value of the `pkg/inst_root` property was different in the previous snapshot.

```
$ svcprop -s previous -p pkg/inst_root pkg/server:s11
/var/share/pkg/repositories/solaris
$ svccfg -s pkg/server:s11
svc:/application/pkg/server:s11> selectsnap previous
[previous]svc:/application/pkg/server:s11> listprop pkg/inst_root
pkg/inst_root astring     /var/share/pkg/repositories/solaris
[previous]svc:/application/pkg/server:s11> exit
```

Showing Configuration Customizations

The `svccfg listcust` command displays customizations at the `admin` layer for the specified service. Use the `-L` option to also show customizations in the `enterprise-profile`, `site-profile`, `node-profile`, and `sysconfig-profile` layers.

The following command shows all customizations at the `admin` layer of the `pkg/server:solaris` service:

```
$ svccfg -s pkg/server:solaris listcust
general                           framework   admin
```

```
general/complete                  astring     admin
general/enabled                   boolean     admin                 true
pkg                               application admin
pkg/inst_root                     astring     admin                 /var/share/pkgrepos/
solaris
pkg/port                          count       admin                 83
pkg/readonly                      boolean     admin                 true
pkg/standalone                    boolean     admin                 false
```

The following command shows that the definition of the property `config/nodename` is provided at the `manifest` layer, but the value `solaris` is set at the `node-profile` layer.

```
$ svccfg -s identity:node listprop -l all -o propname,layer,value config/nodename
nodename        node-profile   solaris
nodename        manifest
```

The following command shows only `admin` layer customizations of the `identity:node` service:

```
$ svccfg -s identity:node listcust
config/loopback   astring     admin                 solaris
```

The following command shows all customizations of the `identity:node` service:

```
$ svccfg -s identity:node listcust -L
config            application node-profile
config/loopback   astring     admin                 solaris
config/nodename   astring     node-profile          solaris
general           framework   node-profile
general/enabled   boolean     node-profile          true
```

The `svccfg listcust` command also displays all *masked* entities. Use the `-M` option to list only masked entities. Before you use the `svccfg delcust` command, use the `svccfg listcust` command to verify what will be deleted. See "Deleting Property Groups, Properties, and Property Values" on page 95 and the `smf`(7) man page for a description of masked entities.

Showing Event Notification Parameters

The `svcs -n` command displays the FMA events notification parameters, system wide SMF state transition notification parameters, and service instance state transition notification parameters. See "Notification Parameters" in the `smf`(7) man page for information about these parameters.

```
$ svcs -n
Notification parameters for FMA Events
```

```
    Event: problem-diagnosed
        Notification Type: smtp
            Active: true
            reply-to: root@localhost
            to: root@localhost

        Notification Type: snmp
            Active: true

        Notification Type: syslog
            Active: true

    Event: problem-repaired
        Notification Type: snmp
            Active: true

    Event: problem-resolved
        Notification Type: snmp
            Active: true

System wide notification parameters:
svc:/system/svc/global:default:
    Event: to-maintenance
        Notification Type: smtp
            Active: true
            to: sysadmins@example.com

svc:/application/pkg/mirror:default:
    Event: to-maintenance
        Notification Type: smtp
            Active: true
            to: installteam@example.com
```

Three FMA events are shown: `problem-diagnosed`, `problem-repaired`, and `problem-resolved`. Notification parameters can also be configured for a fourth event: `problem-updated`.

For the system wide state transition notification setting, the service that stores these global settings is also listed. This system wide setting is a custom setting. System wide, or global, values apply to all service instances that do not have custom values set.

The last setting shown is a custom setting for a particular service instance.

Use the `svccfg listnotify` command to show notification parameters for only the specified event. For state transition events, use the `-g` option to show global settings. The output also shows the source of the notification parameter values.

```
$ svccfg listnotify problem-resolved
    Event: problem-resolved (source: svc:/system/fm/notify-params:default)
```

```
        Notification Type: snmp
            Active: true
$ svccfg listnotify -g to-maintenance
    Event: to-maintenance (source: svc:/system/svc/global:default)
        Notification Type: smtp
            Active: true
            to: sysadmins@example.com
$ svccfg -s pkg/mirror listnotify to-maintenance
    Event: to-maintenance (source: svc:/application/pkg/mirror)
        Notification Type: smtp
            Active: true
            to: installteam@example.com
```

See "Configuring Notification of State Transition and FMA Events" on page 77 for information about configuring event notification.

♦ ♦ ♦ CHAPTER 3

Administering Services

This chapter describes:

- How to start, stop, and restart a service
- How to reread service configuration
- How to configure the system to notify you of FMA events or service state transitions

The command that changes service state is `svcadm`. The `svcadm` command operates on a service instance. If you provide a service name with no instance specified, and that service has only a single instance, `svcadm` operates on that instance. If you provide a service name with no instance specified, and that service has multiple instances, or if you specify any other pattern that matches multiple instances, `svcadm` issues an error message.

Managing SMF Service Instances

A service instance is always in one of the states described in "Service States" on page 24. This section discusses how to cause an instance to transition to a different state, how to commit updated property values to the running snapshot, and how to delete instances from normal view.

Starting a Service

A service instance that is in any of the following states is already enabled and does not need to be started: `degraded`, `maintenance`, `offline`, `online`. If the instance you want to start is in the `degraded`, `maintenance`, or `offline` state, see "Repairing an Instance That Is Degraded, Offline, or in Maintenance" on page 117. If the instance you want to start is in the `disabled` state, enable the instance as shown in the following procedure. When you enable an instance, the restarter for that instance attempts to transition the instance to the `online` state.

▼ How to Enable a Service Instance

1. **Check the instance state and dependencies.**

   Check that the instance is currently disabled and that all of its required dependencies are running (in the `online` or `degraded` state).

   ```
   $ svcs -l FMRI
   ```

2. **Enable the instance.**

   The restarter for the service attempts to bring the specified instance to the `online` state.

   An instance can be permanently or temporarily enabled. Permanent enable is persistent across system reboot and is the default. Temporary enable lasts only until reboot.

   - **Permanently enable the instance.**

     ```
     $ svcadm enable FMRI
     ```

   - **Temporarily enable the instance.**

     Use the `-t` option to specify temporary enable.

     ```
     $ svcadm enable -t FMRI
     ```

     If you want an instance to run now but not run on next reboot, make sure the instance is disabled, and then temporarily enable the instance. To verify that the instance is temporarily enabled, use the `svcs -l` command and check the `enabled` row:

     ```
     enabled      true (temporary)
     ```

   - **Synchronously enable the instance.**

     If you specify the `-s` option, `svcadm` enables the instance and waits for the instance to enter the `online` or `degraded` state before returning. The `svcadm` command returns when the instance reaches an online state or when it determines that the instance requires administrator intervention to reach an online state.

     Use the `-T` option with the `-s` option to specify an upper bound in seconds to make the transition or determine that the transition cannot be made.

     ```
     $ svcadm enable -sT 10 FMRI
     ```

3. **Verify that the instance is online.**

   ```
   $ svcs FMRI
   ```

If the instance is in the `degraded`, `maintenance`, or `offline` state, see "Repairing an Instance That Is Degraded, Offline, or in Maintenance" on page 117.

Example 16 Enabling a Service Instance Permanently

The following command shows that the `pkg/mirror:default` service instance is currently disabled, and all of its required dependencies are online.

```
$ svcs -l pkg/mirror
fmri         svc:/application/pkg/mirror:default
name         IPS Repository Mirror
enabled      false
state        disabled
next_state   none
state_time   September 17, 2013 07:16:52 AM PDT
restarter    svc:/system/svc/restarter:default
manifest     /lib/svc/manifest/application/pkg/pkg-mirror.xml
dependency   require_all/error svc:/milestone/network:default (online)
dependency   require_all/none svc:/system/filesystem/local:default (online)
dependency   optional_all/error svc:/system/filesystem/autofs:default (online)
dependency   require_all/none svc:/application/pkg/repositories-setup (online)
```

The following command enables the `pkg/mirror:default` instance. In this case, the `svcadm` command returns because the `pkg/mirror:default` instance is successfully enabled.

```
$ svcadm enable -sT 10 pkg/mirror:default
$ svcs pkg/mirror
STATE          STIME    FMRI
online         22:03:53 svc:/application/pkg/mirror:default
```

Example 17 Enabling a Service Instance Temporarily

The following command shows that the `net-snmp:default` service instance is currently disabled, and all of its required dependencies are online. The one dependency that is disabled is an optional dependency.

```
$ svcs -l net-snmp
fmri         svc:/application/management/net-snmp:default
name         net-snmp SNMP daemon
enabled      false
state        disabled
next_state   none
state_time   September 17, 2013 05:56:39 PM PDT
logfile      /var/svc/log/application-management-net-snmp:default.log
restarter    svc:/system/svc/restarter:default
contract_id
```

```
manifest     /etc/svc/profile/generic.xml
manifest     /lib/svc/manifest/application/management/net-snmp.xml
dependency   require_all/none svc:/system/filesystem/local (online)
dependency   optional_all/none svc:/milestone/name-services (online)
dependency   optional_all/none svc:/system/system-log (online)
dependency   optional_all/none svc:/network/rpc/rstat (disabled)
dependency   require_all/restart svc:/system/cryptosvc (online)
dependency   require_all/restart svc:/milestone/network (online)
dependency   require_all/refresh file://localhost/etc/net-snmp/snmp/snmpd.conf (online)
dependency   require_all/none svc:/milestone/multi-user (online)
```

After enabling the instance using the `-t` option as shown in the following example, the instance is temporarily enabled, is online, and has a contract ID because it has started the `snmpd` process, as shown by the `svcs -p` command.

```
$ svcadm enable -t net-snmp:default
$ svcs -l net-snmp
fmri         svc:/application/management/net-snmp:default
name         net-snmp SNMP daemon
enabled      true (temporary)
state        online
next_state   none
state_time   September 17, 2013 05:57:26 PM PDT
logfile      /var/svc/log/application-management-net-snmp:default.log
restarter    svc:/system/svc/restarter:default
contract_id  160
manifest     /etc/svc/profile/generic.xml
manifest     /lib/svc/manifest/application/management/net-snmp.xml
dependency   require_all/none svc:/system/filesystem/local (online)
dependency   optional_all/none svc:/milestone/name-services (online)
dependency   optional_all/none svc:/system/system-log (online)
dependency   optional_all/none svc:/network/rpc/rstat (disabled)
dependency   require_all/restart svc:/system/cryptosvc (online)
dependency   require_all/restart svc:/milestone/network (online)
dependency   require_all/refresh file://localhost/etc/net-snmp/snmp/snmpd.conf (online)
dependency   require_all/none svc:/milestone/multi-user (online)
$ svcs -p net-snmp
STATE          STIME    FMRI
online         17:57:26 svc:/application/management/net-snmp:default
               17:57:26     5022 snmpd
```

Stopping a Service

Use the `svcadm disable` command to disable an enabled or temporarily disabled service instance. A disabled instance cannot be restarted. You must first enable the instance.

Note - Security best practice recommends that you periodically review your `online` services to assess whether any of them are no longer used and no longer need to be running.

▼ How to Disable a Service Instance

1. **Check whether other services depend on this instance.**

   a. **List services that depend on this instance.**

   ```
   $ svcs -D FMRI
   ```

   b. **Check whether the dependent service requires this instance.**

   For each result from the `svcs -D` command, use the `svcs -l` command to check whether the dependency is a required dependency.

   You should not disable this instance if this instance is a required dependency of another service. See "Showing Service Dependencies" on page 41 for information about dependency groupings and `restart_on` values.

2. **Disable the instance.**

   The restarter for the service attempts to bring the specified instance to the `disabled` state. In general, the restarter for the service attempts to run the stop method if a stop method exists. The periodic restarter does not attempt to run any stop method because processes contracted by a periodic instance do not run persistently. Periodic instances run short-lived processes and then wait until the next scheduled time to run. See Chapter 3, "Creating a Service to Run Periodically" in *Developing System Services in Oracle Solaris 11.4* for more information.

   An instance can be permanently or temporarily disabled. Permanent disable is persistent across system reboot and is the default. Temporary disable lasts only until reboot.

   - **Permanently disable the instance.**

   ```
   $ svcadm disable FMRI
   ```

   - **Temporarily disable the instance.**

   Use the `-t` option to specify temporary disable.

   ```
   $ svcadm disable -t FMRI
   ```

   If you want an instance to be disabled now but run on next reboot, make sure the instance is running (in the `online` or `degraded` state), and then temporarily disable the instance. To verify that the instance is temporarily disabled, use the `svcs -l` command and check the `enabled` row:

```
enabled      false (temporary)
```

- **Synchronously disable the instance.**

  If you specify the `-s` option, `svcadm` disables the instance and waits for the instance to enter the `disabled` state before returning. The `svcadm` command returns when the instance reaches the `disabled` state or when it determines that the instance requires administrator intervention to reach the `disabled` state.

  Use the `-T` option with the `-s` option to specify an upper bound in seconds to make the transition or determine that the transition cannot be made.

  ```
  $ svcadm disable -sT 10 FMRI
  ```

3. **Verify that the instance is disabled.**

   ```
   $ svcs FMRI
   ```

Example 18 Disabling a Service Instance

The following commands show that the Generic Security Service, `rpc/gss`, is online, and services that depend on the `rpc/gss` service are disabled:

```
$ svcs rpc/gss
STATE          STIME    FMRI
online         Oct_20   svc:/network/rpc/gss:default
$ svcs -D rpc/gss
STATE          STIME    FMRI
disabled       Oct_20   svc:/network/nfs/client:default
disabled       Oct_20   svc:/network/smb/client:default
disabled       Oct_21   svc:/network/nfs/server:default
```

The following command shows that even if the dependents were online, the `rpc/gss` service is an optional dependency and no attempt will be made to start the `rpc/gss` service if any of these three dependent services is refreshed or stopped for any reason:

```
$ svcs -l nfs/client smb/client nfs/server | grep rpc/gss
dependency   optional_all/none svc:/network/rpc/gss (online)
dependency   optional_all/none svc:/network/rpc/gss (online)
dependency   optional_all/none svc:/network/rpc/gss (online)
```

The `svcadm disable` command is successful, the instance is currently in the `disabled` state, and the restart attempt fails.

```
$ svcadm disable rpc/gss
```

```
$ svcs rpc/gss
STATE          STIME    FMRI
disabled       12:45:55 svc:/network/rpc/gss:default
$ svcadm restart pkg/update:default
$ svcs rpc/gss
STATE          STIME    FMRI
disabled       12:45:55 svc:/network/rpc/gss:default
```

Restarting a Service

The restart operation only restarts instances that are currently running (in the `online` or `degraded` state). You might need to restart a running instance because you have made a configuration change that cannot be effected while the instance is running, for example.

Restarting a service instance does not refresh configuration. In general, the `svcadm restart` command runs the stop method of the instance and then runs the start method of the instance. The periodic restarter runs only the start method. The `svcadm restart` command does not commit property changes into the running snapshot and does not run the refresh method of the instance. See "Rereading Service Configuration" on page 72 for information about committing configuration changes into the running snapshot.

Restarting the `manifest-import` service is a special case. Restarting the `manifest-import` service imports any changed manifests or profiles in standard locations, commits the changes into the service configuration repository, takes a new running snapshot, and runs the refresh method of changed instances if a refresh method exists.

▼ How to Restart a Service Instance

1. **Check the instance state.**

   The instance must be in the `online` or `degraded` state.

   ```
   $ svcs FMRI
   ```

2. **Restart the instance.**

   The restarter for the service attempts to bring the specified instance to the `online` state. Most restarters implement the restart operation as a stop operation followed by a start operation.

   - **Restart the instance.**

     ```
     $ svcadm restart FMRI
     ```

- **Synchronously restart the instance.**

  If you specify the `-s` option, `svcadm` restarts the instance and waits for the instance to enter the `online`, `degraded`, or `maintenance` state before returning. The `svcadm` command returns when the instance reaches one of these states or when it determines that the instance requires administrator intervention to reach one of these states.

  Use the `-T` option with the `-s` option to specify an upper bound in seconds to make the transition or determine that the transition cannot be made.

  ```
  $ svcadm restart -sT 10 FMRI
  ```

3. **Verify that the instance is started.**

   If the restart is successful, the instance is in the `online`, `degraded`, or `maintenance` state. If the instance is in the `degraded` or `maintenance` state, see "Repairing an Instance That Is Degraded, Offline, or in Maintenance" on page 117.

   ```
   $ svcs FMRI
   ```

Rereading Service Configuration

You can use the following methods to change service configuration:

- Use the `svccfg setprop` command.
- Edit the service manifest file.
- Edit a profile or provide a new profile associated with the service.

When you change service configuration or provide new configuration, those changes do not immediately appear in the running snapshot. To apply configuration changes, refresh the service instances. This implementation enables you to make multiple changes to multiple services and then apply all the changes at once.

To apply changes you made by using the `svccfg setprop` command, you must use either the `svcadm refresh` or `svccfg refresh` command. The service instance must have a `refresh` method and must be in the `online` or `degraded` state.

To apply changes you made by modifying or adding manifest files or profiles, you can use either the `svcadm refresh` or `svccfg refresh` command, or you can restart the `manifest-import` service. The service instance must be in the `online` or `degraded` state. The `manifest-import` service updates the configuration of all running instances whose manifests or profiles have changed. The `manifest-import` service does not apply configuration changes that you made by using the `svccfg setprop` command. In addition to manually restarting the `manifest-`

`import` service, booting the system and many `pkg` operations also restart the `manifest-import` service.

The `svcadm refresh` and `svccfg refresh` commands both perform the following steps:

1. Create a new running snapshot to commit the editing properties into the running snapshot.
2. Run the refresh method of the instance, if a refresh method exists and the instance is in the `online` or `degraded` state. The refresh method should notify the application that changes have been made. The refresh method might reread property values from the running snapshot. Even if no refresh method exists, the configuration in the running snapshot is updated.

   The periodic restarter does not attempt to run any refresh method. When a periodic instance is refreshed, the periodic restarter rereads the values of the properties in the `periodic` property group described in "Storing Periodic Service Data in the Service Configuration Repository" in *Developing System Services in Oracle Solaris 11.4*.

The `svcadm refresh` command operates on a service instance. The `svccfg refresh` command operates on a service instance or on a parent service. If a service is specified, the `svccfg refresh` command refreshes all instances of that service. While snapshots are taken only for service instances and not for parent services, parent service properties are inherited by service instances. Changed parent service properties appear in a service instance snapshot if the instance does not override those changes.

Some changes, such as dependency changes, take effect immediately. Other changes do not become effective until the service is restarted as described in "Restarting a Service" on page 71. Changes that cannot be made while the application is running require a refresh followed by a restart. Examples of changes that cannot be made while the application is running include closing or opening a socket or resetting an environment variable.

If you specify the `-s` option with the `svcadm refresh` command, `svcadm` refreshes the instance and waits for the instance to enter the `online`, `degraded`, or `maintenance` state before returning. The `svcadm` command returns when the instance reaches one of these states or when it determines that the instance requires administrator intervention to reach one of these states. Use the `-T` option with the `-s` option to specify an upper bound in seconds to make the transition or determine that the transition cannot be made.

Deleting a Service

The `svccfg delete` command does not remove a service instance from the system. Instead, the `svccfg delete` command masks the instance. After you run the `svccfg delete` command, the service manifest still exists in `/lib/svc/manifest`. SMF keeps the service configuration repository in sync with file system content. Since the manifest still exists on the file system in a standard location, that service information is still stored in the repository and is only masked from normal view. Any administrative customizations are deleted from a masked instance. See the `smf`(7) man page for a description of masked entities.

Files that support a service instance are updated when you use `pkg` commands, even if that service instance is masked. When files that support a service instance are updated by `pkg` commands, the SMF data store is updated even though the service is still masked from view. If the service instance is unmasked, that service instance is already updated from the files delivered by `pkg` with no further intervention needed. To unmask a service instance, see "How to Undo Deletion of a Service Instance" on page 74.

▼ How to Delete a Service Instance

1. **Check the dependents of the instance to be deleted.**

   Use the `svcs -D` command to show instances that depend on this instance. After you delete this instance, dependent instances might not be able to run. Use the `svcs -l` command to check whether this instance is a required dependency of the dependent instance.

2. **Mask the instance.**

   Use the `svccfg delete` command to mask the instance from normal view. Use the `svcs` command to show the state of the instance. If the instance is running (is in the `online` or `degraded` state), use the `svccfg delete -f` command to mask the instance from normal view.

```
$ svcs -H my-svc
disabled        7:25:37 svc:/site/my-svc:default
$ svccfg delete svc:/site/my-svc:default
```

3. **Verify that the instance is masked.**

   Use the `svccfg listcust -M` command to confirm that the instance is masked. Commands such as `svcs` should display an error message that no matching instance is found.

```
$ svccfg listcust -M
svc:/site/my-svc:default manifest MASKED
  general                    admin    MASKED
  general/complete astring   admin    MASKED
  general/enabled  boolean   admin    MASKED true
$ svcs -H my-svc
svcs: Pattern 'my-svc' doesn't match any instances
```

▼ How to Undo Deletion of a Service Instance

1. **Confirm that the instance is masked.**

Use the `svccfg listcust -M` command as shown in the previous procedure.

2. **Unmask the instance.**

```
$ svccfg -s svc:/site/my-svc:default delcust
 Deleting customizations for instance: default
```

Reimporting the manifest does not remove a mask.

3. **Verify that the instance is unmasked.**

Use the `svccfg listcust -M` command to confirm that the instance is not masked. The `svcs` command should display the state of the instance.

Changing the Goals of a Goal Service

A goal service defines a set of services that must be running to achieve certain functionality. The services in this critical set are specified as dependencies of the goal service.

Use one of the following methods to set the dependencies of a goal service:

- Use the `svcadm goals` command.
- Modify a profile file.

See Chapter 7, "Creating a Service that Notifies if Conditions are not Satisfied" in *Developing System Services in Oracle Solaris 11.4* for information about recommended types of services for the dependency set of a goal service.

Using the `svcadm goals` Command

The `svcadm goals` command takes a list of services on which the goal service depends. The set of dependencies for the goal service becomes the set of services named as operands of the `svcadm goals` command. To add a dependency, you must list all current dependencies as well as the new dependency.

Services specified as operands of the `svcadm goals` command are administrative customizations of the goal service, or `admin` layer changes. See "Repository Layers" on page 30 for information about layers. The `svcadm goals` command resets all administrative customizations on the dependency set of the goal service.

Dependencies set by using the `svcadm goals` command are in the `require_all` dependency grouping, and the value of the `restart_on` attribute is `restart`. See "Dependency Groupings" on page 42 and "Showing Whether a Service Will Automatically Restart" on page 45 for more information. If you want a dependency of a goal service to have a different grouping or different `restart_on` setting, use the `svccfg` command to change the attributes of that dependency. For example, if you must include a dynamically-enabled service as a dependency of a goal service, use the `svccfg` command to change the value of the `grouping` attribute of the dependency to `optional_all`.

EXAMPLE 19 Resetting the Dependency Set for a Goal Service

The following command replaces the dependency set for the `milestone/goals` goal service with the specified `database/oracle`, `database/listener`, and `milestone/multi-user-server` services. The `milestone/multi-user-server` service is the default goal specified at the `manifest` layer. If you do not include `multi-user-server` in the `svcadm goals` command, then `multi-user-server` will be masked and not used as a goal. The `svcadm goals` command replaces all currently set goals. By default, the `svcadm goals` command operates on the `milestone/goals` service. To replace the dependency set for a goal service that is not the `milestone/goals` service, use the `-g` option to specify the goal service.

```
$ svcadm goals site/application/database/oracle:default \
> site/application/database/listener:default \
> svc:/milestone/multi-user-server:default
```

Both of the following commands show that the `milestone/goals` service now has three goal dependencies:

```
$ svcprop -p goals-dependencies/entities milestone/goals
$ svcs -d milestone/goals
```

Both of the following commands show that resetting the dependency set of a goal service is an administrative customization. See "Showing Configuration Customizations" on page 61 for more information about customizations.

```
$ svccfg -s milestone/goals listcust
$ svccfg -s milestone/goals listprop -l all goals-dependencies/entities
```

EXAMPLE 20 Removing Dependencies From a Goal Service

The `-c` option clears administrative customizations of the dependency set. The set of dependencies for the goal service reverts to the set of dependencies delivered at the manifest and profile layers. See "Repository Layers" on page 30 for information about layer content and hierarchy.

The following command removes all dependencies except the default `milestone/multi-user-server` dependency from the `milestone/goals` service. Use the `-g` option to clear all `admin` layer dependencies for a goal service that is not the `milestone/goals` service.

```
$ svcadm goals -c
```

Both of the following commands show that only goal dependencies set at the manifest layer or a profile layer remain:

```
$ svcprop -p goals-dependencies/entities -l all milestone/goals
$ svccfg -s milestone/goals listprop -l all goals-dependencies/entities
```

The `svccfg delcust` command can remove all `admin` layer customizations for a service instance. To remove only the dependencies customizations for a goal service instance, use the `-c` option of the `svcadm goals` command, See Example 40, "Deleting Customizations," on page 97 for more information about the `delcust` subcommand.

Setting Goal Services in a Profile File

To set goals as file-backed configuration, also called configuration with bundle support, create an SMF profile as described in Chapter 5, "Configuring Multiple Systems". File-backed configuration cannot be removed by using the `svcadm goals -c` command.

For an existing goal service, the easiest way to create a profile is to use the `svccfg extract` command as described in "How to Create a Profile by Using `svccfg`" on page 110. Modify the `goals-dependencies` section.

The following command shows customizations made in profile layers as well as any customizations made in the `admin` layer:

```
$ svccfg -s milestone/goals listcust -L goals-dependencies
```

Configuring Notification of State Transition and FMA Events

You can configure the system to notify you when a service changes state or when an FMA event occurs. You can specify either Simple Mail Transfer Protocol (SMTP) or Simple Network Management Protocol (SNMP) notification.

SMF uses the system SNMP configuration. No SNMP configuration is required in SMF unless you want to change the default notification. By default, SNMP traps are sent on `maintenance` transitions. If you use SNMP for transition notification, you can configure additional traps for other state transitions. See the `snmp-notify`(8) man page for examples.

The following examples show how to set notification parameters for SMF and FMA events and how to delete notification parameters. If you receive a notification of a service in the `maintenance`, `offline`, or `degraded` state, use the `svcs` command to investigate, as described in "Repairing an Instance That Is Degraded, Offline, or in Maintenance" on page 117.

EXAMPLE 21 Configuring a Global Notification for a Service State Event

The following command creates a notification that sends email when services go into the `maintenance` state.

```
$ svccfg setnotify -g to-maintenance mailto:sysadmins@example.com
```

`-g`	The `-g` option sets this notification parameter for all service instances that do not have custom values set. All modified service instances are refreshed. The `-g` option can only be used when setting notification for service state transitions, not with FMA events.
`to-maintenance`	The `to-maintenance` argument is a state transition event as described in "Notification Parameters" in the `smf`(7) man page. Specifying only the state name includes both `to-`*state* and `from-`*state* transitions. This event can also be a comma separated list of transitions.
`mailto:`	The `mailto` argument specifies the notification you want to receive for the specified event. This argument could also specify `snmp`. An `snmp` notification value must be either `snmp:active` or `snmp:inactive`. A `mailto` notification value can be either `mailto:active` or `mailto:inactive`, in addition to the form shown in this example. Setting a notification parameter overwrites any existing value for that event. The `active` and `inactive` settings do not overwrite existing values but toggle whether the existing notification is in effect for the specified event.

EXAMPLE 22 Configuring Notification for Multiple State Events

This example notifies whenever the system shuts down, successfully boots, or fails to boot. The following command creates a notification that sends email when services transition into the `online` state, out of the `online` state, or into the `maintenance` state. Specifying `online` is equivalent to specifying both `to-online` and `from-online`.

```
$ svccfg setnotify -g online,to-maintenance mailto:sysadmins@example.com
```

EXAMPLE 23 Configuring a Notification for a Specified Service Instance

The following command creates a notification that sends email when the `pkg/mirror` service transitions into the `maintenance` state.

```
$ svccfg -s pkg/mirror setnotify to-maintenance mailto:installteam@example.com
```

The following command creates a notification that sends email when the `http:apache24` service transitions out of the `online` state.

```
$ svccfg -s http:apache24 setnotify from-online mailto:webservices@example.com
```

EXAMPLE 24 Configuring a Notification for an FMA Event

The `problem-diagnosed` argument is an FMA event. This argument can be a comma separated list of FMA events. See the list of FMA events in “Notification Parameters” in the `smf`(7) man page.

```
$ svccfg setnotify problem-diagnosed mailto:IT@example.com
```

EXAMPLE 25 Deleting Notification Settings

The following commands delete the notification settings set in the previous examples.

```
$ svccfg delnotify -g to-maintenance
$ svccfg -s pkg/mirror delnotify to-maintenance
$ svccfg setnotify problem-diagnosed mailto:root@localhost
```

♦ ♦ ♦ C H A P T E R 4

Configuring Services

SMF stores configuration data in the service configuration repository. Configuring SMF services means modifying the data in the configuration repository and then committing the modifications into the running snapshot. This chapter describes how to modify the data in the configuration repository. For viewing data in the configuration repository, see “Inspecting Service Configuration” on page 49. For committing configuration modifications into the running snapshot, see “Rereading Service Configuration” on page 72.

Each service and service instance stores configuration data in properties, which are organized into property groups. Modifying the data in the configuration repository includes modifying service property values, creating custom property groups and properties, creating new instances of a service, and applying a profile. Modifying configuration also includes deleting customizations and reverting repository snapshots.

This chapter describes:

- Adding and modifying property values
- Adding and deleting properties and property groups
- Adding service instances
- Reverting snapshots
- Importing and applying service manifests and profiles
- Modifying services that are controlled by `inetd`
- Modifying services that are configured by a file

SMF configuration changes can be logged by using the Oracle Solaris auditing framework. Refer to “Configuring the Audit Service” in *Managing Auditing in Oracle Solaris 11.4* for more information.

Using the Service Configuration Command

The `svccfg` command manipulates data in the service configuration repository. Changes made with the `svccfg` command are recorded in the `admin` layer. See “Repository Layers” on page 30 for information about layers. Changes made with the `svccfg` command are stored in the service configuration repository as current, or editing, property values, and do not immediately appear in the running snapshot. When you change configuration data, use the `svcadm refresh` or `svccfg refresh` command to commit the new values into the running snapshot.

Keeping newly changed data separate from the running snapshot enables you to make multiple changes, and then commit all the changes to the running snapshot together. While you are in the process of making multiple changes, some property values might be incompatible or inconsistent, but the running snapshot is unmodified. When you are finished making changes, perform a refresh.

You can use the `svccfg` command in any of the following ways:

- Use the `svccfg editprop` command to invoke an editor on the property groups and properties of the currently selected entity.
- Enter a full `svccfg` command on the command line, specifying subcommands such as `setprop`.
- Enter only `svccfg` or `svccfg -s` *FMRI* on the command line to start an interactive session.
- Specify the `-f` option to read `svccfg` commands from a file.

Invoking a Property Editor

Invoking the `svccfg` command as shown in the following example opens an editor on the properties of the selected entity. This form of the `svccfg` command can be very fast and convenient for modifying several property values. For the `editprop` subcommand, you must specify an entity with the `-s` option.

```
$ svccfg -s pkg/server:s11 editprop
```

A file of `setprop` commands for the current values of each property of the specified entity opens in the editor specified by the `VISUAL` environment variable. If `VISUAL` is not defined, the editor specified by `EDITOR` is opened. If neither `VISUAL` nor `EDITOR` is defined, the property file is opened in `vi`.

Each line of the file is preceded by a comment character. To change the value of a property in the `svccfg` editing configuration, remove the comment character, change the value, and save the file. To change the value of a property in the running snapshot, remove the comment character from the last line of the file, which is the `refresh` subcommand.

The following listing shows a partial example of a file created by the `editprop` subcommand:

```
##
## Change property values by removing the leading '#' from the
## appropriate lines and editing the values. svccfg subcommands
## such as delprop can also be added to the script.
##
## Property group "pkg"
## The following properties are defined in the selected instance
## (svc:/application/pkg/server:s11)

# setprop pkg/port = count: 81
# setprop pkg/inst_root = astring: /export/ipsrepos/Solaris11

## The following properties inherit from the parent service
## (svc:/application/pkg/server)

# ...

## Property group "pkg_bui"

# ...

## Property group "pkg_secure"

# ...

## Uncomment to apply these changes to this instance.
# refresh
```

As the instructions in the file state, you can add subcommands other than `setprop`. For example, you could add a `delprop` command. Some property groups, such as `framework` and `dependency`, are not displayed by default. Specify `editprop -a` to show all properties.

The uncommented commands in this temporary file are executed when you save and quit the editing session.

Invoking `svccfg` Interactively or With a File

Invoking the `svccfg` command interactively as shown in the following example can be convenient when you want to perform several configuration operations.

```
$ svccfg
svc:> select pkg/server
svc:/application/pkg/server> list
:properties
```

```
default
svc:/application/pkg/server> add s11
svc:/application/pkg/server> select s11
svc:/application/pkg/server:s11> setprop pkg/inst_root=/export/ipsrepos/Solaris11
svc:/application/pkg/server:s11> setprop pkg/port=81
svc:/application/pkg/server:s11> unselect
svc:/application/pkg/server> add oss
svc:/application/pkg/server> select oss
svc:/application/pkg/server:oss> setprop pkg/inst_root=/export/ipsrepos/SolarisStudio
svc:/application/pkg/server:oss> setprop pkg/port=82
svc:/application/pkg/server:oss> unselect
svc:/application/pkg/server> list
:properties
default
s11
oss
svc:/application/pkg/server> refresh
svc:/application/pkg/server> select pkg/mirror:default
svc:/application/pkg/mirror:default> listprop config/crontab_period
config/crontab_period astring     "30 2 25 * *"
svc:/application/pkg/mirror:default> setprop config/crontab_period="00 3 25 * *"
svc:/application/pkg/mirror:default> refresh
svc:/application/pkg/mirror:default> exit
$
```

The same commands given at the interactive prompts in the preceding example could also be provided in a file and executed with a command such as the following command.

```
$ svccfg -f cfgpkgrepos
```

Setting Property Values

The following commands set property values:

`svccfg setprop`

Changes the value of a property.

`svccfg addpropvalue`

Adds a value to a multi-value property.

`svccfg setenv`

Changes the value of an environment variable for a service process execution environment.

Remember to use the `svccfg refresh` command or `svcadm refresh` command to commit configuration changes into the running snapshot.

EXAMPLE 26 Setting a Simple Value

In the simplest use of `setprop`, specify a *pg*/*name* for the selected service or instance, where *pg* is the name of the property group and *name* is the name of the property, and specify the new value after an equals symbol. If the property already exists or is templated, you do not need to specify the property type.

```
$ svccfg -s pkg/server:s11 setprop pkg/port=81
```

EXAMPLE 27 Setting a Value that Contains Embedded Spaces

Use double quotation marks to set a value that contains embedded spaces. Depending on your shell, you might need to enclose the double-quoted string in single quotation marks.

```
$ svccfg -s pkg/mirror setprop config/crontab_period = "00 3 25 * *"
$ svccfg -s pkg/mirror setprop config/crontab_period = '"00 3 25 * *"'
```

Use quotation marks to set a value that contains double quotation marks or backslash characters, and use a backslash character to escape any double quotation marks or backslash characters.

EXAMPLE 28 Setting a Value that Is a Set of Values

Use parentheses to specify a set of values as a single value. Depending on your shell, you might need to escape the parentheses.

```
$ svccfg -s dns/client setprop config/nameserver = (10.0.0.1 192.168.0.1)
$ svccfg -s dns/client setprop config/nameserver = \(10.0.0.1 192.168.0.1\)
$ svccfg -s dns/client listprop config/nameserver
config/nameserver net_address 10.0.0.1 192.168.0.1
```

Use the `describe` subcommand to find the number of values allowed in the set of values.

```
$ svccfg -s dns/client describe -v config/nameserver
config/nameserver net_address 10.0.0.1 192.168.0.1
    type: net_address
    required: false
    The IP address of a DNS nameserver to be used by the resolver.
    visibility: readwrite
    minimum number of values: 1
    maximum number of values: 3
  value: 10.0.0.1
  value: 192.168.0.1
```

EXAMPLE 29 Setting a Property Whose Name Contains Special Characters

"Naming Property Groups and Properties" in *Developing System Services in Oracle Solaris 11.4* provides a list of reserved characters that can be used in property group and property names. These reserved characters appear encoded in FMRIs:

```
$ svccfg -s enchars_example:default
svc:/site/enchars_example:default> listprop config
config                     application
config/%25%20increase      count       10
config/maximum%20%23       count       9
config/start%3Aend         count       10
config/students%2Fteachers count       20
```

To set these properties, you can copy the encoded names:

```
svc:/site/enchars_example:default> setprop config/students%2Fteachers=21
svc:/site/enchars_example:default> listprop config/students%2Fteachers
config/students%2Fteachers count       21
```

Another way to set these properties is to use the `-G` and `-P` options:

```
svc:/site/enchars_example:default> setprop -G config -P students/teachers 20
svc:/site/enchars_example:default> listprop -G config -P students/teachers
config/students%2Fteachers count       20
```

Use quotation marks for property group names or property names that contain spaces:

```
svc:/site/enchars_example:default> setprop -G config -P "% increase" 12
svc:/site/enchars_example:default> listprop -G config -P "% increase"
config/%25%20increase count       12
```

The `editprop` tool automatically uses the `-G` and `-P` options:

```
$ svccfg -s enchars_example:default editprop
...
$ setprop -G "config" -P "% increase" -T count 10
$ setprop -G "config" -P "maximum #" -T count 9
$ setprop -G "config" -P "start:end" -T count 10
$ setprop -G "config" -P "students/teachers" -T count 20
```

EXAMPLE 30 Setting the Value of a Property in a Nested Property Group

To set the value of a property in a property group that is the child of another property group, specify the full ancestry of the property.

In the following example, `http` and `https` are child property groups of the `config` property group, and `ssl` is a child property group of the `https` property group:

```
$ svccfg -s npg_example listprop config
config                    application
config/port               count       80
config/http               application
config/http/port          count       80
config/https              application
config/https/port         count       443
config/https/ssl          application
config/https/ssl/certfile astring     cert.crt
config/https/ssl/keyfile  astring     key.crt
```

To specify different values for properties of nested property groups, specify the full FMRI of the property or use multiple `-G` options. The following example changes the values of the certificate and key files for the default instance:

```
$ svccfg -s npg_example:default
svc:/site/npg_example:default> setprop config/https/ssl/certfile=cert-1.pem
svc:/site/npg_example:default> setprop -G config -G https -G ssl -P keyfile key-1.pem
svc:/site/npg_example:default> listprop config/https/ssl
config/https/ssl          application
config/https/ssl/certfile astring     cert-1.pem
config/https/ssl/keyfile  astring     key-1.pem
```

EXAMPLE 31 Adding a Value

Use the `addpropvalue` subcommand to add the given value to the specified property of the selected service or service instance. The new value is appended to the end of the existing list of property values for the property.

```
$ svcprop -p keymap/layout keymap:default
US-English
$ svccfg -s keymap:default addpropvalue keymap/layout UK-English
$ svccfg -s keymap:default listprop keymap/layout
keymap/layout astring     "US-English" "UK-English"
```

In the previous `setprop` example, all values in the set of values must be specified at once. If only one value is specified, that value becomes the new set of one value. In this `addpropvalue` example, the added values are distinct. To access these added values, you must use the `libscf` function `scf_iter_property_values()` to iterate over the values. While `listprop` lists both values, `describe` lists only the first value and reports that the maximum allowed number of values for this property is one.

```
$ svccfg -s keymap:default describe -v keymap/layout
keymap/layout astring     US-English
```

```
    type: astring
    required: true
    The keyboard layout
    visibility: readwrite
    minimum number of values: 1
    maximum number of values: 1
  value: US-English
```

▼ How to Schedule a Periodic or Scheduled Service

A periodic or scheduled service runs its start method on a cadence or at a time that you specify by setting property values of the service.

1. **Confirm that the service you want to schedule is a periodic or scheduled service.**
   The following command lists all periodic and scheduled services on the system because it lists all instances whose restarter is the `periodic-restarter` service:

   ```
   $ svcs -R svc:/system/svc/periodic-restarter:default
   ```

   You could also use `svcs -l`, `svcprop`, or `svccfg` to show the value of the `general/restarter` property.

2. **Show the current schedule for the service.**
   The following command shows whether the instance is a periodic service or a scheduled service. A periodic service has a `periodic` property group, and a scheduled service has a `scheduled` property group:

   ```
   $ svcprop -p periodic -p scheduled FMRI
   ```

   If *FMRI* is a periodic service, the values of the `periodic/period` property and any other `periodic` properties that are set are shown. An error message is shown for the `scheduled` property group.
   If *FMRI* is a scheduled service, the values of the `scheduled/interval` and `scheduled/frequency` properties and any other `scheduled` properties that are set are shown. An error message is shown for the `periodic` property group.

3. **Change the schedule.**
   Properties that have no value are not shown by the `svcprop` or `svccfg listprop` commands. Use the following references to decide which properties to set to achieve the schedule you want:

   - For periodic services, see "Specifying the periodic_method Element" in *Developing System Services in Oracle Solaris 11.4* and "Scheduling Executions of a Periodic Service Start

Method" in *Developing System Services in Oracle Solaris 11.4*. Only the `periodic/period` property is required to schedule a periodic service. Other properties are optional.

- For scheduled services, see "Specifying the scheduled_method Element" in *Developing System Services in Oracle Solaris 11.4* and "Scheduling Executions of a Scheduled Service Start Method" in *Developing System Services in Oracle Solaris 11.4*. The `scheduled/interval` and `scheduled/frequency` properties are required to schedule a scheduled service.

If you set a property that has not been set before, you must specify the property type as shown in Example 32, "Scheduling the Compliance Assessment Service," on page 89. If you reset that property value, you do not need to specify the property type again. Also, if you set a property that has not been set before, you will see the following message:

```
Type required for new properties.
```

4. **Read the modified property values.**

   Use the `svcadm` or `svccfg` command to refresh the service instance. Use the `svcprop` command again to verify that the `periodic` or `scheduled` property values are updated.

5. **Verify that the service instance is online.**

   The instance might have been disabled while you were modifying property values, or the instance might have gone into maintenance when you performed the refresh. If the instance is in maintenance, view the log file as described in "Viewing Service Log Files" on page 47. Possible problems include the following:

   - Conflicting properties are set. For example, you cannot set both the `scheduled/day_of_month` and `scheduled/day` property values.
   - Not all required properties are set. For example, you cannot set just the `day` and `minute`, you must also set the `hour`.

   For more information about moving a service out of the maintenance state, see "How to Repair an Instance That Is in Maintenance" on page 118.

6. **Verify the schedule.**

   The following command shows the next time the service will run:

   ```
   $ svcs -o nrun FMRI
   ```

   If no schedule is shown, make sure the instance is online.

Example 32 Scheduling the Compliance Assessment Service

The following command shows that the compliance assessment service is currently scheduled to run once each week:

```
$ svcprop -p scheduled compliance:default
scheduled/frequency integer 1
scheduled/interval astring week
```

With this schedule, the instance can run at any time on any day the first week. Subsequent runs will then occur some time that same day in future weeks.

The following commands set the instance to invoke its start method some time between 3am and 4am every Wednesday:

```
$ svccfg -s compliance:default
svc:/application/security/compliance:default> setprop scheduled/day=Wednesday
Type required for new properties.
svc:/application/security/compliance:default> setprop scheduled/day = astring: Wednesday
svc:/application/security/compliance:default> setprop scheduled/hour = integer: 3
svc:/application/security/compliance:default> refresh
svc:/application/security/compliance:default> exit
$ svcs compliance:default
STATE          STIME    FMRI
online         13:16:57 svc:/application/security/compliance:default
$ svcs -o nrun compliance:default
NRUN
 3:27:16
```

▼ How to Modify a `ttymon` Property Value

This procedure shows how to modify parameters passed to `ttymon`.

1. **Identify the service to modify.**

   The `ttymon`(8) man page states that the service to modify is `svc:/system/console-login`. The `ttymon`(8) man page also contains descriptions of the properties in the `ttymon` property group.

   The following command shows multiple instances of the `console-login` service in this image and shows that the `default` instance is the only instance currently online:

```
$ svcs console-login
STATE          STIME    FMRI
disabled       10:49:43 svc:/system/console-login:terma
disabled       10:49:43 svc:/system/console-login:termb
online         10:50:54 svc:/system/console-login:default
```

2. **Identify the property to modify.**

   The following command shows the name, data type, value, and a brief description of each property in the `ttymon` property group in the `default` instance:

```
$ svccfg -s console-login:default describe ttymon
ttymon                application
ttymon/device        astring     /dev/console
    The terminal device to be used for the console login prompt.
ttymon/terminal_type astring
    Sets the initial value of the TERM environment variable
```

The preceding output shows no value for the `terminal_type` property. The following command confirms that the value of the `ttymon/terminal_type` property of the `console-login:default` instance is currently null:

```
$ svcprop -p ttymon/terminal_type console-login:default
""
```

3. **Modify the property value.**

   Enter the following command to change the value of the `ttymon/terminal_type` property of the `console-login:default` instance to `xterm`:

```
$ svccfg -s system/console-login:default setprop ttymon/terminal_type=xterm
```

4. **Commit the new value into the running snapshot.**

   The following output shows that the value of the `terminal_type` property is changed in the editing configuration but not in the running snapshot:

```
$ svccfg -s console-login:default listprop ttymon/terminal_type
ttymon/terminal_type astring     xterm
$ svcprop -p ttymon/terminal_type console-login:default
""
```

   After you refresh the service instance, the property value is changed in the running snapshot:

```
$ svcadm refresh console-login:default
$ svcprop -p ttymon/terminal_type console-login:default
xterm
```

▼ How to Modify an Environment Variable for a Service Process Environment

This procedure shows how to set a value for an environment variable in the environment where processes started by the service will run.

The example in this procedure shows how to modify `cron` environment variables to help with debugging.

1. **Verify that the service is running.**

   The following output shows that the `cron` service is online and a `cron` process is running.

   ```
   $ svcs -p cron
   STATE          STIME    FMRI
   online         10:24:05 svc:/system/cron:default
                  10:24:05     1089 cron
   ```

2. **Set environment variables.**

   The `setenv` subcommand sets an environment variable for the environment where a process started by a service or service instance will run.

   Use the following command to check the current values of the environment variables you want to set:

   ```
   $ pargs -e `pgrep -f /usr/sbin/cron`
   ```

   The environment variables that are set in this example do not have any current values.

   The following commands set the `UMEM_DEBUG` and `LD_PRELOAD` environment variables for the `/usr/sbin/cron` process started by the `svc:/system/cron:default` service instance:

   ```
   $ svccfg -s system/cron:default setenv UMEM_DEBUG default
   $ svccfg -s system/cron:default setenv LD_PRELOAD libumem.so
   ```

3. **Refresh and restart the service.**

   Changing an environment variable value requires a restart as well as a refresh to take effect.

   ```
   $ svcadm refresh system/cron:default
   $ svcadm restart system/cron:default
   ```

4. **Verify that the change has been made.**

   The following output shows that the service has been restarted, the process has a new process ID, and the two environment variables are set for that process environment.

   ```
   $ svcs -p cron
   STATE          STIME    FMRI
   online          9:24:39 svc:/system/cron:default
                   9:24:39     5601 cron
   $ svcprop -g method -p environment system/cron:default
   start/environment astring LD_PRELOAD=libumem.so UMEM_DEBUG=default
   $ pargs -e `pgrep -f /usr/sbin/cron`
   5601: /usr/sbin/cron
   ```

```
envp[0]: LOGNAME=root
envp[1]: LD_PRELOAD=libumem.so
envp[2]: PATH=/usr/sbin:/usr/bin
envp[3]: SMF_FMRI=svc:/system/cron:default
envp[4]: SMF_METHOD=start
envp[5]: SMF_RESTARTER=svc:/system/svc/restarter:default
envp[6]: SMF_ZONENAME=global
envp[7]: UMEM_DEBUG=default
```

See Also The `unsetenv` subcommand unsets an environment variable for a process started by a service or service instance.

Adding Property Groups, Properties, and Property Values

The following commands add properties and property groups:

`svccfg setprop`
`svccfg addpropvalue`

Adds the property whose value is being set if the property does not already exist.

`svccfg addpg`

Adds a new property group to a service or service instance.

Remember to use the `svccfg refresh` command or `svcadm refresh` command to commit configuration changes into the running snapshot.

EXAMPLE 33 Using `addpg` to Create a New Property Group

Use the `addpg` subcommand to add a property group to the selected service or service instance.

`svccfg -s` *FMRI* `addpg` *name type* `[`*flags*`]`

type	By convention, the value of *type* is usually `application`. See *Developing System Services in Oracle Solaris 11.4* for more information about property group types.
flags	Specify `P` for the value of *flags* to store the property group and any added properties as non-persistent. If `P` is specified, this property group and contained properties will be automatically removed on reboot, The value `P` is an alias for `SCF_PG_FLAG_NONPERSISTENT`. See the `scf_service_add_pg`(3SCF) man page.

```
$ svccfg -s svc:/site/my-svc addpg config application
$ svccfg -s my-svc listprop config
config  application
$ svccfg -s my-svc:default listprop config
$
```

In this example, the administrator added the `config` property group to the parent service, `my-svc`, but not to the instance, `my-svc:default`. The `listprop` command shows that the `config` property group does not exist in the service instance.

The following example adds a nested property group. Nested property groups must all be the same type.

```
$ svccfg -s my-svc addpg config/security application
$ svccfg -s my-svc listprop config
config          application
config/security application
```

The following example adds a property group whose name includes a reserved character:

```
$ svccfg -s my-svc addpg -G config -G security -G certs+keys application
$ svccfg -s my-svc listprop config
config                        application
config/security               application
config/security/certs%2Bkeys application
```

EXAMPLE 34 Using `setprop` to Create a New Property

Use the `setprop` subcommand to set a property value as described in "Setting Property Values" on page 84. If the property group does not already exist in the selected instance or service, the property group is created if the type and flags are found in the template definitions. If the property does not already exist in the selected instance or service, you must specify the property *type*.

```
$ svccfg -s my-svc:default setprop config/source = uri: http://example1.com/
$ svccfg -s my-svc:default listprop config/source
config/source uri        http://example1.com/
```

To add a new property that has the same name as a property group, use the `-G` and `-P` options:

```
$ svccfg -s my-svc setprop -G config -P security -T astring yes
$ svccfg -s my-svc listprop config/security
config/security               astring     yes
config/security               application
config/security/certs%2Bkeys application
```

The following example adds a property whose name includes a reserved character. Use the `-T` option to specify the type of the new property.

```
$ svccfg -s my-svc setprop -G config -P "maximum %" -T integer 100
$ svccfg -s my-svc listprop -G config -P "maximum %"
config/maximum%20%25 integer     100
```

EXAMPLE 35 Using `addpropvalue` to Create a New Property

Use the `addpropvalue` subcommand to add a property value as described in "Setting Property Values" on page 84. If the property group does not already exist in the selected instance or service, the property group is created if the type and flags are found in the template definitions. If the property does not already exist in the selected instance or service, you must specify the property *type*.

```
$ svccfg -s my-svc:default addpropvalue config/source http://example2.com/
$ svccfg -s my-svc:default addpropvalue config/target hostname: example3
$ svccfg -s my-svc:default listprop config
config                       application
config/source                uri         http://example1.com/ http://example2.com/
config/target                hostname    example3
```

Deleting Property Groups, Properties, and Property Values

The following commands delete property values, properties, and property groups:

`svccfg setprop`

Delete all values of a property.

`svccfg delpropvalue`

Delete all values of the specified property that match the specified pattern.

`svccfg delprop`

Delete a property.

`svccfg delpg`

Delete a property group.

`svccfg delcust`

Delete administrative customizations.

Remember to use the `svccfg refresh` command or `svcadm refresh` command to commit configuration changes into the running snapshot.

Deleting Administrative Configuration

Configuration modifications made by using `svccfg` commands or `libscf` calls modify only the `admin` layer of the service configuration repository. See "Repository Layers" on page 30 for information about layers. When you delete configuration that is only defined in the `admin` layer and does not exist in any other layer, that configuration is gone. Commands that display configuration no longer show the deleted configuration, even when you use the `-l` option to show all layers of the service configuration repository. See "Deleting Non-Administrative Configuration" on page 98 for information about deleting configuration that exists in other layers.

EXAMPLE 36 Deleting All Values of a Property

Use the `setprop` subcommand as described in "Setting Property Values" on page 84. To delete all values of a property, do not specify any type or value. The values are deleted, but the property still exists.

```
$ svccfg -s my-svc:default setprop config/vendor =
$ svccfg -s my-svc:default listprop config/vendor
config/vendor   astring
```

EXAMPLE 37 Deleting All Matching Values of a Property

Use the `delpropvalue` subcommand to delete all values of the named property that match the given pattern.

```
$ svccfg -s my-svc:default setprop config/tool = astring: \(hammer tongs wrench\)
$ svccfg -s my-svc:default listprop config
config           application
config/customer astring     acustomer
config/vendor   astring     "vendora" "vendorb"
config/tool     astring     "hammer tongs wrench"
$ svccfg -s my-svc:default delpropvalue config/vendor '*b'
$ svccfg -s my-svc:default delpropvalue config/tool 'tong*'
$ svccfg -s my-svc:default listprop config
config           application
config/customer astring     acustomer
config/vendor   astring     vendora
config/tool     astring     "hammer tongs wrench"
$ # config/tool is a single value that is a value set
$ svccfg -s my-svc:default delpropvalue config/tool '*tong*'
$ svccfg -s my-svc:default listprop config
config           application
config/customer astring     acustomer
config/vendor   astring     vendora
```

```
config/tool     astring
```

EXAMPLE 38 Deleting a Property

Use the `delprop` subcommand to delete the named property of the selected service or service instance.

```
$ svccfg -s my-svc:default delprop config/tool
$ svccfg -s my-svc:default listprop config
config           application
config/customer astring     acustomer
config/vendor   astring     vendora
```

EXAMPLE 39 Deleting a Property Group

The `delpg` and `delprop` subcommands both can delete a property group. The `delpg` subcommand deletes the named property group of the selected service or service instance. The `delprop` subcommand deletes the named property group if no property is named.

```
$ svccfg -s my-svc:default delpg config
$ svccfg -s my-svc:default listprop config
$
```

EXAMPLE 40 Deleting Customizations

The `delcust` subcommand deletes administrative customizations on the selected service or service instance. Before you use the `delcust` subcommand, use the `listcust` subcommand with the same pattern or option to see what will be deleted. If a pattern is given, the pattern must match a property or property group.

```
$ svccfg -s my-svc:default listcust
config                            application admin
config/customer                  astring     admin                acustomer
config/vendor                    astring     admin                "vendora" "vendorb"
config/tool                      astring     admin                "hammer tongs
 wrench"
$ svccfg -s my-svc:default listcust '*tool'
config/tool                      astring     admin                "hammer tongs
 wrench"
$ svccfg -s my-svc:default delcust '*tool'
 Deleting customizations for property: config/tool
$ svccfg -s my-svc:default listcust '*tool'
$ svccfg -s my-svc:default listcust
config                            application admin
config/customer                  astring     admin                acustomer
```

```
config/vendor                      astring     admin                 "vendora" "vendorb"
```

Deleting Non-Administrative Configuration

Configuration that exists in the `sysconfig-profile`, `node-profile`, `site-profile`, `enterprise-profile`, `system-profile`, and `manifest` layers of the service configuration repository is defined in service manifests and profile files. See "Repository Layers" on page 30 for information about layers. SMF keeps the service configuration repository in sync with file system content. Any configuration that is defined in a manifest or profile file in a standard location still exists on the file system after administrative customization, including after being deleted, and is still stored in the service configuration repository. Configuration that is defined in a manifest or profile is said to have bundle support. When you delete configuration that has bundle support, the information is not deleted from the file system but is *masked* so that it is not seen in the normal view. See the `smf`(7) man page for a description of masked entities.

Deleting configuration that has bundle support is an administrative customization. In this case, the `delcust` subcommand *unmasks* the configuration, rather than deleting anything. Use the `listcust -M` subcommand to view masked configuration. Use the `delcust -M` subcommand to unmask configuration, or undo the deletion or masking of the configuration.

EXAMPLE 41 Deleting Configuration that has Bundle Support

In "Deleting Administrative Configuration" on page 96, the `config` property group of the `my-svc` service only existed in the `admin` layer. The `config` property group did not exist in any manifest or profile. When those properties were deleted, they were gone from the system. This example shows the different result when you delete configuration that has bundle support.

The property is defined in the service manifest:

```
$ svccfg -s pkg/server listprop -l all pkg/inst_root
pkg/inst_root astring     admin               /export/ipsrepos/Solaris11
pkg/inst_root astring     manifest            /var/pkgrepo
$ svccfg -s pkg/server delprop pkg/inst_root
```

After deletion, the property is not displayed by using `listprop` with no options. Because the property has bundle support, the property still exists in the service configuration repository and can be displayed by using the `-l` or `-M` options with the `listprop` subcommand.

```
$ svccfg -s pkg/server listprop pkg/inst_root
$ svccfg -s pkg/server listprop -l all pkg/inst_root
pkg/inst_root astring     admin        MASKED /export/ipsrepos/Solaris11
pkg/inst_root astring     manifest     MASKED /var/pkgrepo
```

```
$ svccfg -s pkg/server listcust -M
pkg/inst_root astring     admin       MASKED /export/ipsrepos/Solaris11
```

EXAMPLE 42 Unmasking Configuration

When you unmask the property, both customizations are gone:

- The property is no longer masked or hidden.
- The property no longer has its customized value.

```
$ svccfg -s pkg/server delcust -M
 Deleting customizations for property: pkg/inst_root
$ svccfg -s pkg/server listprop -l all pkg/inst_root
pkg/inst_root astring     manifest            /var/pkgrepo
$ svccfg -s pkg/server listprop pkg/inst_root
pkg/inst_root astring     /var/pkgrepo
```

Adding Service Instances

Instances of a service allow multiple configurations of a service to run simultaneously. Service instances inherit and customize common service configuration.

Use the `add` subcommand to create a new entity with the given name as a child of the selected service.

```
$ svcs -Ho inst pkg/server
default
$ svccfg -s pkg/server add s11
$ svcs -Ho inst pkg/server
default
s11
```

Reverting Snapshots

Each of the following operations creates a new running snapshot:

- `svcadm restart manifest-import`
- `svcadm refresh`

- `svccfg refresh`

The `revert` subcommand reverts the administrative customizations (`admin` layer) of the instance specified by the `-s` option and its service to the values recorded in the named snapshot or the currently selected snapshot. Use the `listsnap` subcommand to view a list of possible snapshots for this service instance. Use the `selectsnap` subcommand to select a snapshot in interactive mode.

```
$ svcprop -p pkg/inst_root pkg/server:default
pkg/inst_root astring /export/ipsrepos/Solaris11
$ svccfg -s pkg/server:default listsnap
initial
previous
running
start
$ svcprop -s previous -p pkg/inst_root pkg/server:default
pkg/inst_root astring /var/pkgrepo
```

Because the `revert` subcommand reverts all administrative customizations, list all administrative customizations and examine their values before you revert.

```
$ svcprop -s previous -l admin pkg/server:default
pkg/inst_root astring /var/pkgrepo
$ svccfg -s pkg/server:default revert previous
$ svcadm refresh pkg/server:default
$ svcprop -p pkg/inst_root pkg/server:default
pkg/inst_root astring /var/pkgrepo
```

Importing and Applying Manifests and Profiles

When you restart the `manifest-import` service, manifests in standard locations are imported and profiles in standard locations are applied if they are new or changed. See "Service Bundles" on page 29 for manifest and profile standard locations. If importing a manifest or applying a profile results in the service being started or stopped, the appropriate method is executed if one exists.

Specifying a file in a standard location to the `svccfg import` command restarts the `manifest-import` service.

Recommended best practice is to put your manifest and profile files in the standard locations and restart the `manifest-import` service rather than use the `svccfg import` or `svccfg apply` commands.

```
$ svcadm restart manifest-import
```

When you restart the manifest-import service, the configuration in profiles and manifests in standard locations is applied to the appropriate layer (sysconfig-profile, node-profile, site-profile, enterprise-profile, system-profile, or manifest) of affected instances, affected instances are refreshed and validated, and a new snapshot is created. See "Repository Layers" on page 30 for a description of each layer.

When you import or apply profiles and manifests in non-standard locations, configuration is applied to the admin layer of affected instances. Using non-standard locations is strongly discouraged for default or initial configuration delivery. For making a large number of configuration changes, importing or applying from a non-standard location might be easier than issuing many commands, but you lose the benefit of the automated management mechanisms of the manifest-import service. To manage service delivery, the manifest-import service requires known locations and expected states.

The svccfg apply command applies all configuration to the admin layer, even for manifests and profiles in standard locations.

Modifying Services that are Controlled by inetd

A service that is controlled by inetd is an SMF service that was converted from a configuration in the inetd.conf file. The inetd command is the delegated restarter for these services.

The following procedure shows how to change property values of services that are controlled by inetd.

To confirm that the service you want to modify is controlled by inetd, use either the inetadm command or the svcs -R command as shown in the following examples to list all inetd controlled services. The following examples show only a partial list:

```
$ inetadm
ENABLED   STATE          FMRI
enabled   online         svc:/application/cups/in-lpd:default
...
disabled  disabled       svc:/application/x11/xvnc-inetd:default
$ svcs -R network/inetd:default
STATE     STIME          FMRI
online    8:11:10        svc:/application/cups/in-lpd:default
...
online    8:11:11        svc:/network/rpc/smserver:default
```

The -l option of the inetadm command lists all the properties of the inetd controlled service. In the following example, the error message indicates that the specified service is not an `inetd` controlled service. “No restarter property” means that the master restarter, `svc.startd`, manages the service instance.

```
$ inetadm -l ssh
Error: Specified service instance "svc:/network/ssh:default" has no restarter
property.  inetd is not the delegated restarter of this instance.
```

Similarly, in the following example, the message “Couldn't find property 'general/restarter'” indicates that the default restarter, `svc.startd`, manages the service instance.

```
$ svcprop -p general/restarter ssh
svcprop: Couldn't find property 'general/restarter' for instance
'svc:/network/ssh:default'.
```

If a service is controlled by `inetd`, its restarter is `inetd`, as shown in the following example:

```
$ svcprop -p general/restarter cups/in-lpd
svc:/network/inetd:default
```

The `svcs -l` command also shows the restarter. The following example shows only partial output:

```
$ svcs -l cups/in-lpd
...
restarter    svc:/network/inetd:default
...
```

▼ How to Change a Property Value for an `inetd` Controlled Service

1. **List the properties for the service.**

   Use the `-l` option of the `inetadm` command to list all the properties of the specified service. Inspect the current values of the properties.

   ```
   $ inetadm -l FMRI
   ```

2. **Change a property value.**

   Use the `-m` option of the `inetadm` command to change the value of a specified property. Specific information about the properties for a service should be covered in the man page associated with the service.

   ```
   $ inetadm -m FMRI property-name=value
   ```

   To delete a property value, specify an empty value.

```
$ inetadm -m svc property=""
```

3. **Verify that the property value is changed.**

   List the properties again to make sure that the appropriate change has occurred.

   ```
   $ inetadm -l FMRI
   ```

4. **Confirm that the change has taken effect.**

   Confirm that the property change has the expected effect.

Example 43 Limiting the Number of Concurrent Processes that are Allowed to Run

This example shows how to limit the number of `finger` processes that are allowed to run concurrently.

Security best practice recommends limiting the number of processes that are allowed to run concurrently for each system service that is controlled by `inetd`. In addition, if a service that is controlled by `inetd` is not needed, disable that service as described in "Stopping a Service" on page 68.

The property to configure to limit the number of processes that are allowed to run concurrently is the `max_copies` property. The `inetadm -p` command lists the properties common to all services managed by `inetd` and their default values. The following example shows only partial output:

```
$ inetadm -p
NAME=VALUE
...
max_copies=-1
...
```

The `inetadm -l` command lists all the properties of the specified service so that you can inspect the current value of the `max_copies` property for that service.

```
$ inetadm -l finger | grep copies
default  max_copies=-1
```

Use the `-m` option of the `inetadm` command to change the value of the property:

```
$ inetadm -m finger max_copies=5
```

Verify that the property value is changed.

```
$ inetadm -l finger | grep copies
         max_copies=5
```

Notice that the scope of “default” is no longer displayed for this property.

▼ How to Add a New Instance of a Service that is Controlled by `inetd`

1. **Determine the location of the manifest for the service.**

```
$ svcs -l FMRI
...
manifest     /lib/svc/manifest/path/file.xml
```

2. **Edit the manifest file to add a new instance with appropriate property values.**

   a. **Copy and paste an existing instance element.**

   b. **Give the new instance element a unique instance name.**

   c. **Change the property values as necessary.**

3. **Add the new instance to the profile file.**

   Edit the file `/etc/svc/profile/generic.xml` to add a new `instance` element inside the `service` element for the appropriate service.

4. **Restart the manifest import service.**

```
$ svcadm restart manifest-import
```

5. **Verify that the new instance has been added.**

   Search for the *FMRI* in the output of the `svcs -a` or `inetadm` command.

6. **Verify the property values of the new instance.**

```
$ inetadm -l FMRI
```

Modifying Services that are Configured by a File

A few SMF services that are not managed by `inetd` get some of their configuration from a file rather than from service properties. To modify this configuration, edit the configuration file and use SMF commands to restart the service. These configuration files can be changed while the service is running, but the content of the files is only read when the service is started.

Before you edit a configuration file directly, check the following conditions:

- Make sure the configuration file does not contain a message telling you not to directly edit it.
- Make sure the service does not have a property group of type `configfile`.

  ```
  $ svcprop -g configfile network/ntp
  ```

  If the service has a property group of type `configfile`, modify the properties in those property groups and not the configuration file. See Chapter 6, "Using a Stencil to Create a Configuration File" in *Developing System Services in Oracle Solaris 11.4*.

For example, to add a new NTP server to support your NTP clients, add a new entry for the server to the `/etc/inet/ntp.conf` file and then restart the NTP service as shown in the following command:

```
$ svcadm restart svc:/network/ntp:default
```

To enable IKEv2, modify the `/etc/inet/ike/ikev2.config` file to configure the IKEv2 daemon, and then enable the IKEv2 service as shown in the following command. To edit the `ikev2.config` file, use the `pfedit` command as a user who is assigned the Network IPsec Management profile. Editing the file in this way preserves the correct file ownership. See the `pfedit`(8) man page for information about using `pfedit`.

```
$ svcadm enable svc:/network/ipsec/ike:ikev2
```

♦ ♦ ♦ CHAPTER 5

Configuring Multiple Systems

This chapter describes the recommended way to manage configuration change across multiple systems.

The most accurate, efficient way to deliver consistent system configuration to multiple systems across system updates is to deliver SMF profiles in IPS packages.

This chapter describes:

- How to manage the configuration of multiple systems
- How to create SMF profiles
- How to deliver configuration to multiple systems

Managing Configuration of Multiple Systems

The following steps summarize how to manage configuration of multiple systems:

1. Create SMF profiles that specify the services you want enabled and the values of their properties. Profiles can add and set properties for existing services and instances and specify new service instances. Profiles can specify almost anything that a manifest can specify.
2. Create IPS packages to install those profiles in the `/etc/svc/profile/enterprise` or `/etc/svc/profile/site` directories on each system.
    - A profile that describes configuration for all systems in your enterprise goes in the `/etc/svc/profile/enterprise` directory.
    - A profile that describes configuration for systems in a particular location or site goes in the `/etc/svc/profile/site` directory.
    - A profile that describes configuration for a single Oracle Solaris system goes in the `/etc/svc/profile/node` directory.

    See "Repository Layers" on page 30 for descriptions of these profile directories and configuration layers.

Creating SMF Profiles

The following steps summarize how to create SMF profile files:

1. Use any of the following methods to create SMF profile files:
   - Use the `svccfg extract` command to capture profile information from an existing service as described in "How to Create a Profile by Using `svccfg`" on page 110.
   - Use the `sysconfig create-profile` command to create a new profile file as described in "How to Create a Profile by Using `sysconfig`" on page 111.
   - Use the `svcbundle` command with `bundle-type=profile` to create a new profile file as described in "How to Create a Profile by Using `svcbundle`" on page 112.
2. Customize property values in the profile file, and include comments about the reason for each customization. Ensure that you do not specify conflicting configuration. See "Conflicting Configuration" on page 108 for more information.
3. To test the profile, copy the profile file to the appropriate `/etc/svc/profile/` layer subdirectory, and restart the `manifest-import` service.

   Use one of the following commands to verify that the configuration described in profile files has been applied:
   - `svccfg listcust -L`
   - `svccfg listprop -l all`
   - `svcprop -l all`

Conflicting Configuration

When you create a profile, ensure that the configuration defined does not conflict with configuration defined in the same layer in any other profile for the same service or service instance.

Examples of conflicting configuration include different values for the same property and different types for the same property or property group.

If conflicting configuration is delivered by multiple files in any single layer, and is not set at a higher layer, the `manifest-import` service log reports the conflict and the service with the conflicting configuration is not started and is placed in the `maintenance` state.

If you create two profile files, each of which specify different values for the same property, place both of these profile files in the same `/etc/svc/profile/` layer subdirectory, and reboot the system, you get results similar to the following:

- If you have notifications configured, the notification reports the conflict. The following example is a service state transition notification:

  ```
  HOSTNAME: host
  TIMESTAMP: June 27, 2016 at 04:15:43 PM PDT
  FMRI: svc:/site/example:default
  FROM-STATE: uninitialized
  TO-STATE: maintenance
  DESCRIPTION: The indicated service has transitioned to the maintenance state
  REASON: the instance is in conflict
  ```

  The following example is an FMA event notification. Note the information in the Impact and Recommended Action sections.

  ```
  SUNW-MSG-ID: SMF-8001-02, TYPE: Defect, VER: 1, SEVERITY: Major
  EVENT-TIME: Mon Jun 27 16:14:07 PDT 2016
  PLATFORM: Ultra 24, CSN: 0817FMB003, HOSTNAME: host
  SOURCE: software-diagnosis, REV: 0.2
  EVENT-ID: c46b48bb-484d-4c9f-a82b-e1349a0ddce6
  DESC: The Solaris Service manager tried to import a manifest or apply a profile
   defining
  the service, but detected one or more entities with conflicting definitions.
  AUTO-RESPONSE: The service may have been placed into the maintenance state.
  IMPACT: The service is not running.  It will not be started until the conflict is
   resolved
  and the maintenance state is cleared.  Services with require-type dependencies on the
  service will not be started.  (Use 'svcs -xv svc:/site/nm_example:default' to see a
   list
  of services waiting for the service to start.)
  REC-ACTION: Edit the problematic manifest or profile to resolve the conflict and
   reimport
  or reapply it, or use svccfg to administratively override the conflicting
   definitions.
  Then clear the maintenance state. Please refer to the associated reference document
   at
  http://support.oracle.com/msg/SMF-8001-02 for the latest service procedures and
   policies
  regarding this diagnosis.
  ```

- The `svcs -x` command reports the conflicts. All locations where conflicts are found are listed, so that you can more easily find the problem. The following output lines are broken for readability.

  ```
  $  svcs -x
  svc:/site/example:default (site/example)
  ```

```
 State: maintenance since Wed Nov 16 15:29:51 2016
Reason: Instance has conflicts.
Conflicting value: FMRI="svc:/site/example"; Name of conflicting property="config/
prop1";
from file="/etc/svc/profile/node/example_prof1.xml"; from file="/etc/svc/profile/
site/example_prof2.xml";
Conflicting value: FMRI="svc:/site/example"; Name of conflicting property="config/
prop2";
from file="/etc/svc/profile/node/example_prof1.xml";
   See: http://support.oracle.com/msg/SMF-8001-02
   See: /var/svc/log/site-example:default.log
Impact: This service is not running.
```

- The `manifest-import` service log file also reports the conflicts. The following output lines are broken for readability.

```
$ svcs -xL manifest-import
svc:/system/manifest-import:default (service manifest import)
 State: online since Wed Nov 16 15:29:52 2016
   See: smf_bootstrap(7)
   See: /var/svc/log/system-manifest-import:default.log
Impact: None.
   Log:
svccfg: svc:/site/example: property group "config" has a conflict.
Conflicting value: FMRI="svc:/site/example"; Name of conflicting property="config/
prop1";
from file="/etc/svc/profile/node/example_prof1.xml"; from file="/etc/svc/profile/
site/example_prof2.xml";
Conflicting value: FMRI="svc:/site/example"; Name of conflicting property="config/
prop2";
from file="/etc/svc/profile/node/example_prof1.xml";
svccfg: Multiple definitions of property group reg in entity default.
[ 2016 Nov 16 15:29:52 Method "start" exited with status 0. ]
```

You can also use the `-l`, `-f`, and `-o` options of the `svccfg listprop` command to investigate the cause of a conflict. See "Showing the Layer Where a Value Is Set" on page 59 and "Showing the File that Contributed the Configuration" on page 60.

▼ How to Create a Profile by Using `svccfg`

1. **Create the profile.**

The `svccfg extract` command displays a service profile for the specified service or instance. To extract values from specific layers, use the `-l` option. The `-l` option argument can be a layer name or a comma-separated list of layer names. See the `svccfg`(8) man page for the complete list of layer names, and see "Repository Layers" on page 30 for descriptions of the layers. The `current` and `all` layer names both mean select the highest-layer value for each property.

The following command extracts the highest-layer setting for each property for the `network/dns/client` service into the `dnsclientprofile.xml` file:

```
$ svccfg extract -l current network/dns/client > dnsclientprofile.xml
```

2. **Make any necessary changes to the profile.**

   Change the name of the profile to a meaningful name. By default, the name is set to `extract`, as shown in the following example:

   ```
   <service_bundle type='profile' name='extract'>
   ```

   Make any changes that are necessary for the target systems.

3. **Copy the profile to the correct directory.**

   The following command indicates that this profile is for all systems at a given locale or site:

   ```
   $ cp dnsclientprofile.xml /etc/svc/profile/site/
   ```

   Change the ownership and permissions of the profile as necessary.

4. **Restart the manifest import service to apply the profile to the system.**

   ```
   $ svcadm restart manifest-import
   ```

   Check the `manifest-import` service log file for messages about property value conflicts or any other error messages.

   ```
   $ svcs -Lv manifest-import
   ```

   Use the `svcs -x` command to check that no services are in maintenance.

▼ How to Create a Profile by Using `sysconfig`

1. **Create the profile.**

   Run the SCI Tool and create a system configuration profile. The default location for the profile is `/system/volatile/profile/sc_profile.xml`. Use the `-o` option to specify a different directory for the profile file.

The following command creates a new profile in the siteAprofiles directory that includes network and naming_services configuration. See the sysconfig(8) man page for the complete list of functional groupings (-g) that can be configured.

```
$ sysconfig create-profile -g network,naming_services -o ./siteAprofiles
```

The SCI tool opens and prompts you for the configuration values.

2. **Make any necessary changes to the profile.**

   The profile is in ./siteAprofiles/sc_profile.xml. You might want to change sc_profile.xml to a more meaningful and unique name.

   ```
   $ cp ./siteAprofiles/sc_profile.xml ./siteAprofiles/netnamingSiteA.xml
   ```

   Make any changes that are necessary for the target systems.

3. **Copy the profile to the correct directory.**

   The following command indicates that this profile is for all systems at a given locale or site:

   ```
   $ cp ./siteAprofiles/netnamingSiteA.xml /etc/svc/profile/site/
   ```

   Change the ownership and permissions of the profile as necessary.

4. **Restart the manifest import service to apply the profile to the system.**

   ```
   $ svcadm restart manifest-import
   ```

   Check the manifest-import service log file for messages about property value conflicts or any other error messages.

   ```
   $ svcs -Lv manifest-import
   ```

   Use the svcs -x command to check that no services are in maintenance.

▼ How to Create a Profile by Using svcbundle

1. **Create the profile.**

   The following command creates a new profile in example.com.xml:

   ```
   $ svcbundle -o example.com.xml -s service-name=enterprise/example.com \
   > -s bundle-type=profile -s instance-property=pg_name:prop_name:prop_type:value \
   > -s service-property=pg_name:prop_name:prop_type:value
   ```

2. **Make any changes that are necessary for the target systems.**
3. **Copy the profile to the correct directory.**

   The following command indicates that this profile is for all systems:

   ```
   $ cp example.com.xml /etc/svc/profile/enterprise/
   ```
4. **Restart the manifest import service to apply the profile to the system.**

   ```
   $ svcadm restart manifest-import
   ```

   Check the `manifest-import` service log file for messages about property value conflicts or any other error messages.

   ```
   $ svcs -Lv manifest-import
   ```

   Use the `svcs -x` command to check that no services are in maintenance.

Delivering Configuration to Multiple Systems

The best way to deliver profiles to your systems depends on how configuration information is controlled in your organization. Common ways of dividing the information into different packages is by site, by network, and by system function. For example, DNS and NTP configuration might be the same for all systems in a DMZ, but would be different from the configurations used in internal development groups. In that same DMZ environment, the systems that act as web servers might all share a common configuration, which might be different from the configuration of systems serving other functions.

Dependencies can be used to build complete configurations out of smaller sets of configuration information. Using dependencies reduces duplication of information across packages.

Create at least one package for each group of systems that require the same configuration. A package can deliver multiple profiles.

- Configuration for different SMF layers must be delivered in different profile files in different directories.
- Within a single layer, you probably want to deliver configuration for different services in different profile files.
- Different groups of systems require different configuration and therefore different profiles.

You might want multiple profile packages per system group, for example to separately deliver configuration that you expect to change more frequently. Multiple profile packages for the same group of systems could be `group` dependencies in one package. A group package also is an easy way to deliver new profile packages for that group.

When a profile needs to change, rebuild the package with the updated profile and increment the package version number. If you are using a group package, update the group package as well. The updated configuration is then installed by `pkg update`.

See *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* for information about how to create your profile packages. The following instructions are specific to SMF profile packages:

- Do not package the `/etc`, `/etc/svc`, or `/etc/svc/profile` directories or any of the standard subdirectories of the `/etc/svc/profile` directory: Those directories are already delivered by system packages.
- Include a `restart_fmri` actuator on each profile or manifest `file` action.

Consider using a periodic service to check whether updates are available for installed profile packages. See *Developing System Services in Oracle Solaris 11.4* for information about creating a periodic service.

If you want to provide individual system administrators with a choice of configuration, deliver mediated links into the `/etc/svc/profile` directories and the target profiles elsewhere. See "Delivering Multiple Implementations of an Application" in *Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4* for more information.

♦ ♦ ♦ A P P E N D I X A

SMF Best Practices and Troubleshooting

This appendix describes best practices and troubleshooting, including:

- Repairing a service instance that is degraded, offline, or in maintenance
- Diagnosing and repairing SMF repository problems
- Specifying the amount of SMF startup messaging
- Specifying the SMF milestone to which to boot
- Investigating system boot problems
- Converting `inetd` services to SMF services

SMF Best Practices

Most services describe configuration policy. If the configuration you want is not implemented, modify the policy description by modifying the service. Modify the values of service properties or create new service instances with different property values. Do not disable service instances and edit configuration files that are intended to be managed by an SMF service. An increasing number of fundamental Oracle Solaris features are configured by SMF service properties, not by editing configuration files.

Do not modify manifests and system profiles that are delivered by Oracle or by third-party software vendors. These manifests and profiles might be replaced when you upgrade your system, and then your changes to these files will be lost. Instead, do one of the following:

- Add a new service instance with different property values as described in "Adding Service Instances" on page 99.
- Create a profile to customize the service. Use the `svcbundle` command or the `svccfg extract` command to create a profile file. Customize property values in that file, and include comments about the reason for each customization. Copy the profile file to the appropriate `/etc/svc/profile` subdirectory, and restart the `manifest-import` service.

  To apply the same custom configuration to multiple systems, copy the same profile file to the same `/etc/svc/profile` subdirectory on each system, and restart the `manifest-import`

service on each system. To automate delivering the profile to each system, package the profile. See Chapter 5, “Configuring Multiple Systems”.
- Use the `svccfg` command or the `inetadm` command to manipulate the properties directly. If you use the `svccfg` command to modify property values, be sure to refresh the service instance as explained in “Understanding Configuration Changes” on page 117. For information about modifying, adding, and deleting service configuration, see Chapter 4, “Configuring Services”. To see configuration that has already been modified, see “Showing Configuration Customizations” on page 61. To delete custom configuration, see Example 40, “Deleting Customizations,” on page 97 and Example 42, “Unmasking Configuration,” on page 99.

When you create a custom profile, make sure the configuration defined does not conflict with configuration defined in the same layer in another manifest or profile for the same service or service instance. Configuration conflicts are not permitted within any layer. If conflicting configuration is delivered by multiple files in any single layer, and is not set at a higher layer, the `manifest-import` service log reports the conflict and the service with the conflicting configuration is not started. See “Conflicting Configuration” on page 108 for more information.

Do not use non-standard locations for manifest and profile files. See “Service Bundles” on page 29 for manifest and profile standard locations.

When you create a service for your own use, use `site` at the beginning of the service name: `svc:/site/`*service_name*`:`*instance_name*.

Do not modify the configuration of the master restarter service, `svc:/system/svc/restarter:default`, except to configure logging levels as described in “Specifying the Amount of Startup Messaging” on page 126.

Before you use the `svccfg delcust` command, use the `svccfg listcust` command with the same options. The `delcust` subcommand can potentially remove all administrative customizations on a service. Use the `listcust` subcommand to verify which customizations will be deleted by the `delcust` subcommand.

In scripts, use the full service instance FMRI: `svc:/`*service_name*`:`*instance_name*.

Troubleshooting Services Problems

This section discusses the following topics:

- Committing configuration changes into the running snapshot
- Fixing services that are reported to have problems

- Manually transitioning an instance to the `degraded` or `maintenance` state
- Fixing a corrupt service configuration repository
- Configuring the amount of messaging to display or store on system startup
- Transitioning or booting to a specified milestone
- Using SMF to investigate booting problems
- Converting `inetd` services to SMF services

Understanding Configuration Changes

In the service configuration repository, SMF stores property changes separately from properties in the running snapshot. When you change service configuration, those changes do not immediately appear in the running snapshot.

The refresh operation updates the running snapshot of the specified service instance with the values from the editing configuration.

By default, the `svcprop` command shows properties in the running snapshot, and the `svccfg` command shows properties in the editing configuration. If you have changed property values but not performed a configuration refresh, the `svcprop` and `svccfg` commands show different property values. After you perform a configuration refresh, the `svcprop` and `svccfg` commands show the same property values.

Rebooting does not change the running snapshot. The `svcadm restart` command does not refresh configuration. Use the `svcadm refresh` or `svccfg refresh` command to commit configuration changes into the running snapshot.

Repairing an Instance That Is Degraded, Offline, or in Maintenance

Use the `svcs -x` command with no arguments to display explanatory information about any service instances that match any of the following descriptions:

- The service is enabled but is not running.
- The service is enabled but is not running at normal capacity.
- The service is preventing another enabled service from running.
- The service is disabled but is not able to complete the transition to the `disabled` state.

The following list summarizes how to approach service problems:

1. Diagnose the problem, starting with viewing the service log file.

   Log files located in `/var/svc/log` and `/system/volatile`. The service log file shows time stamps and method exit reasons.

   The location of the log file for a particular service is given by the following command:

   ```
   $ svcs -L service-name
   ```

   The following command displays the end of the log file for a particular service:

   ```
   $ svcs -Lx service-name
   ```
2. Fix the problem.
   - If multiple service failures are identified, start by looking at the first failure to occur, using the time stamps in the service log files.
   - Use the following command to show impacted dependencies of the failed service:

     ```
     $ svcs -l service-name
     ```

     Use the following command to show services on which *service-name* depends:

     ```
     $ svcs -d service-name
     ```
   - If fixing the problem involves modifying service configuration, refresh the service.
3. Move affected services to a running state.

▼ How to Repair an Instance That Is in Maintenance

A service instance that is in maintenance is enabled but not able to run or is disabled but not able to complete the transition to the `disabled` state.

1. Determine why the instance is in maintenance.

The instance might be transitioning through the `maintenance` state because an administrative action has not yet completed. If the instance is transitioning, its state should be shown as `maintenance*` with an asterisk at the end.

An instance that is configured to restart after failure might be placed in `maintenance` because it was restarting too frequently. In this case you need to determine the cause of the consistent failure.

If an instance is in `maintenance` because it has conflicts, or conflicting property values, see "Conflicting Configuration" on page 108.

The instance might be in the `maintenance` state because the instance was disabled but is unable to reach the `disabled` state because the `stop` method failed.

In the following example, the "State" and "Reason" lines show that the `pkg/depot` service is in the `maintenance` state because its start method failed.

```
$ svcs -x
svc:/application/pkg/depot:default (IPS Depot)
 State: maintenance since September 11, 2013 01:30:42 PM PDT
Reason: Start method exited with $SMF_EXIT_ERR_FATAL.
   See: http://support.oracle.com/msg/SMF-8000-KS
   See: pkg.depot-config(8)
   See: /var/svc/log/application-pkg-depot:default.log
Impact: This service is not running.
```

Log in to the Oracle support site to view the referenced Predictive Self-Healing knowledge article. In this case, the article tells you to view the log file to determine why the start method failed. The `svcs` output gives the name of the log file. See "Viewing Service Log Files" on page 47 for information about how to view the log file. In this example, the log file shows the start method invocation and the fatal error message.

```
[ Sep 11 13:30:42 Executing start method ("/lib/svc/method/svc-pkg-depot start"). ]
pkg.depot-config: Unable to get publisher information:
The path '/export/ipsrepos/Solaris11' does not contain a valid package repository.
```

2. **Fix the problem.**

   One or more of the following steps might be needed.

   - **Update service configuration.**

     If fixing the reported problem required modifying service configuration, use the `svccfg refresh` or `svcadm refresh` command for any services whose configuration changed. Verify that the configuration is updated in the running snapshot by using the `svcprop` command to check property values or by other tests specific to this service.

   - **Ensure dependencies are running.**

     Sometimes the "Impact" line in the `svcs -x` output tells you that services that depend on the service that is in the `maintenance` state are not running. Use the `svcs -l` command to check the current state of dependent services. Ensure that all required dependencies are running. Use the `svcs -x` command to verify that all enabled services are running.

   - **Ensure contract processes are stopped.**

     If the service that is in the `maintenance` state is a contract service, determine whether any processes that were started by the service have not stopped. When a contract service instance is in a maintenance state, the contract ID should be blank, as shown in the following example, and all processes associated with that contract should have stopped.

Use `svcs -l` or `svcs -o ctid` to check that no contract exists for a service instance in maintenance. Use `svcs -p` to check whether any processes associated with this service instance are still running. Any processes shown by `svcs -p` for a service instance in maintenance should be killed.

```
$ svcs -l system-repository
fmri         svc:/application/pkg/system-repository:default
name         IPS System Repository
enabled      true
state        maintenance
next_state   none
state_time   September 17, 2013 07:18:19 AM PDT
logfile      /var/svc/log/application-pkg-system-repository:default.log
restarter    svc:/system/svc/restarter:default
contract_id
manifest     /lib/svc/manifest/application/pkg/pkg-system-repository.xml
dependency   require_all/error svc:/milestone/network:default (online)
dependency   require_all/none svc:/system/filesystem/local:default (online)
dependency   optional_all/error svc:/system/filesystem/autofs:default (online)
```

3. **Notify the restarter that the instance is repaired.**

   When the reported problem is fixed, use the `svcadm clear` command to tell the restarter for that service that the instance is repaired. SMF will attempt to transition the instance to its configured state. If the instance is enabled, SMF will attempt to bring the instance online. If the instance is disabled, SMF will transition the instance to the `disabled` state.

   ```
   $ svcadm clear pkg/depot:default
   ```

   If you specify the `-s` option, the `svcadm` command waits to return until the instance reaches the `online` state or until it determines that the instance cannot reach the `online` state without administrator intervention. Use the `-T` option with the `-s` option to specify an upper bound in seconds to make the transition or determine that the transition cannot be made.

4. **Verify that the instance is repaired.**

   Use the `svcs` command to verify that the service that was in maintenance is now online. Use the `svcs -x` command to verify that all enabled services are running.

▼ How to Repair an Instance That Is Offline

A service instance that is offline is enabled but not running or available to run.

1. **Determine why the instance is offline.**

The instance might be transitioning through the `offline` state because its dependencies are not yet satisfied. If the instance is transitioning, its state should be shown as `offline*`.

2. **Fix the problem.**

   - **Enable service dependencies.**

     If required dependencies are disabled, enable them with the following command:

     ```
     $ svcadm enable -r FMRI
     ```

   - **Fix dependency file.**

     A dependency file might be missing or unreadable. You might want to use `pkg fix` or `pkg revert` to fix this type of problem. See the `pkg`(1) man page.

3. **Restart the instance if necessary.**

   If the instance was offline because a required dependency was not satisfied, fixing or enabling the dependency might cause the offline instance to restart and come online with no further administrative action needed.

   If you made some other fix to the service, then restart the instance.

   ```
   $ svcadm restart FMRI
   ```

4. **Verify that the instance is repaired.**

   Use the `svcs` command to verify that the instance that was offline is now online. Use the `svcs -x` command to verify that all enabled services are running.

▼ How to Repair an Instance That Is Degraded

A service instance that is degraded is enabled and running or available to run, but is functioning at a limited capacity.

1. **Determine why the instance is degraded.**

2. **Fix the problem.**

3. **Request the restarter to online the instance.**

   When the reported problem is fixed, use the `svcadm clear` command to return the instance to the `online` state. For instances in the `degraded` state, the `clear` subcommand requests that the restarter for that instance transition the instance to the `online` state.

   ```
   $ svcadm clear pkg/depot:default
   ```

4. **Verify that the instance is repaired.**

   Use the `svcs` command to verify that the instance that was degraded is now online. Use the `svcs -x` command to verify that all enabled services are running.

Marking an Instance as Degraded or in Maintenance

You can mark a service instance as being in either the `degraded` state or the `maintenance` state. You might want to do this if the application is stuck in a loop or is deadlocked, for example. The information about the state change propagates to the dependencies of the marked instance, which can help debug other related instances.

Specify the `-I` option to request an immediate state change.

When you mark an instance as `maintenance`, you can specify the `-t` option to request a temporary state change. Temporary requests last only until reboot.

If you specify the `-s` option with the `svcadm mark` command, `svcadm` marks the instance and waits for the instance to enter the `degraded`, or `maintenance` state before returning. Use the `-T` option with the `-s` option to specify an upper bound in seconds to make the transition or determine that the transition cannot be made.

Diagnosing and Repairing Repository Problems

On system startup, the repository daemon, `svc.configd`, performs an integrity check of the configuration repository stored in `/etc/svc/repository.db`. If the `svc.configd` integrity check fails, the `svc.configd` daemon writes a message to the console similar to the following:

```
svc.configd: smf(7) database integrity check of:

    /etc/svc/repository.db

  failed.  The database might be damaged or a media error might have
  prevented it from being verified.  Additional information useful to
  your service provider is in:

    /system/volatile/db_errors

  The system will not be able to boot until you have restored a working
  database.  svc.startd(8) will provide a sulogin(8) prompt for recovery
```

```
purposes.  The command:

  /lib/svc/bin/restore_repository

can be run to restore a backup version of your repository. See
http://support.oracle.com/msg/SMF-8000-MY for more information.
```

The `svc.configd` daemon then exits. That exit is detected by the `svc.startd` daemon, and `svc.startd` then starts `sulogin`.

At the `sulogin` prompt, enter Ctrl-D to exit `sulogin`. The `svc.startd` daemon recognizes the `sulogin` exit and restarts the `svc.configd` daemon, which checks the repository again. The problem might not reappear after this additional restart.

Caution - Do not directly invoke the `svc.configd` daemon. The `svc.startd` daemon starts the `svc.configd` daemon.

If `svc.configd` again reports a failed integrity check and you are again at the `sulogin` prompt, ensure that required file systems are not full. Using the `root` password, log in either remotely or at the `sulogin` prompt. Check that space is available on both the root and `system/volatile` file systems. If either of these file systems is full, clean up and start the system again. If neither of these file systems is full, follow the procedure "How to Restore a Repository From Backup" on page 123.

The service configuration repository can become corrupted for any of the following reasons:

- Disk failure
- Hardware bug
- Software bug
- Accidental overwrite of the file

The following procedure shows how to replace a corrupt repository with a backup copy of the repository.

How to Restore a Repository From Backup

Before You Begin

Caution - Only restore a corrupt repository. Do not use this repository restore procedure to delete unwanted configuration changes. To undo configuration changes, see "Showing Configuration Customizations" on page 61, Example 40, "Deleting Customizations," on page 97, and Example 42, "Unmasking Configuration," on page 99.

1. **Log in.**
   Using the `root` password, log in either remotely or at the `sulogin` prompt.

2. **Run the repository restore command:**

   ```
   # /lib/svc/bin/restore_repository
   ```

   Running this command takes you through the necessary steps to restore a non-corrupt backup. SMF automatically takes backups of the repository as described in "Repository Backups" on page 32.

   SMF maintains persistent and non-persistent configuration data. See "Service Configuration Repository" on page 28 for descriptions of these two repositories. The `restore_repository` command only restores the persistent repository. The `restore_repository` command also reboots the system, which destroys the non-persistent configuration data. The non-persistent data is runtime data that is not needed across system reboot.

   When started, the `/lib/svc/bin/restore_repository` command displays a message similar to the following:

   ```
   See http://support.oracle.com/msg/SMF-8000-MY for more information on the use of
   this script to restore backup copies of the smf(7) repository.

   If there are any problems which need human intervention, this script will
   give instructions and then exit back to your shell.
   ```

   After the `root` (`/`) file system is mounted with write permissions, or if the system is a local zone, you are prompted to select the repository backup to restore:

   ```
   The following backups of /etc/svc/repository.db exists, from
   oldest to newest:

   ... list of backups ...
   ```

   Backups are given names, based on type and the time the backup was taken. Backups beginning with `boot` are completed before the first change is made to the repository after system boot. Backups beginning with `manifest_import` are completed after `svc:/system/manifest-import:default` finishes its process. The time of the backup is given in *YYYYMMDD_HHMMSS* format.

3. **Enter the appropriate response.**

   Typically, the most recent backup option is selected.

   ```
   Please enter either a specific backup repository from the above list to
   restore it, or one of the following choices:

           CHOICE            ACTION
           ----------------  ----------------------------------------------
           boot              restore the most recent post-boot backup
           manifest_import   restore the most recent manifest_import backup
   ```

```
        -seed-            restore the initial starting repository  (All
                            customizations will be lost, including those
                            made by the install/upgrade process.)
        -quit-            cancel script and quit

Enter response [boot]:
```

If you press Enter without specifying a backup to restore, the default response, enclosed in `[]` is selected. Selecting `-quit-` exits the `restore_repository` script, returning you to your shell prompt.

Caution - Selecting `-seed-` restores the `seed` repository. This repository is designed for use during initial installation and upgrades. Only use the `seed` repository for recovery purposes when no other service configuration change or backup service repository will work. All configuration changes will be lost, including changes to fundamental Oracle Solaris features that were delivered by installing or updating packages. Using the `seed` repository for recovery purposes should be a last resort.

After you have selected the backup that you want to restore, that backup is validated and its integrity is checked. If any problems are discovered, the `restore_repository` command prints error messages and prompts you for another selection. Once you have selected a valid backup, the following information is printed, and you are prompted for final confirmation.

```
After confirmation, the following steps will be taken:

svc.startd(8) and svc.configd(8) will be quiesced, if running.
/etc/svc/repository.db
    -- renamed --> /etc/svc/repository.db_old_YYYYMMDD_HHMMSS
/system/volatile/db_errors
    -- copied --> /etc/svc/repository.db_old_YYYYMMDD_HHMMSS_errors
repository_to_restore
    -- copied --> /etc/svc/repository.db
and the system will be rebooted with reboot(8).

Proceed [yes/no]?
```

4. **Type `yes` to remedy the fault.**

   The system reboots after the `restore_repository` command executes all of the listed actions.

Specifying the Amount of Startup Messaging

By default, each service that starts during system boot does not display a message on the console. Use one of the following methods to change which messages appear on the console and which are recorded only in the `svc.startd` log file. The value of *logging-level* can be one of the values shown in the table below.

- When booting a SPARC system, specify the `-m` option to the `boot` command at the `ok` prompt. See "Messages options" in the `kernel`(8) man page.

  ```
  ok boot -m logging-level
  ```
- When booting an x86 system, edit the GRUB menu to specify the `-m` option. See "Adding Kernel Arguments at Boot Time" in *Booting and Shutting Down Oracle Solaris 11.4 Systems* and "Messages options" in the `kernel`(8) man page.
- Prior to rebooting a system, use the `svccfg` command to change the value of the `options/logging` property. If this property has never been changed on this system, then it will not exit and you will have to add it. The following example changes to verbose messaging. The change takes effect on the next restart of the `svc.startd` daemon.

  ```
  $ svccfg -s system/svc/restarter:default listprop options/logging
  $ svccfg -s system/svc/restarter:default addpg options application
  $ svccfg -s system/svc/restarter:default setprop options/logging=verbose
  $ svccfg -s system/svc/restarter:default listprop options/logging
  options/logging astring     verbose
  ```

TABLE 2 SMF Startup Message Logging Levels

Logging Level Keyword	Description
`quiet`	Display on the console any error messages that require administrative intervention. Also record these messages in `syslog` and in `/var/svc/log/svc.startd.log`.
`verbose`	In addition to the messaging provided at the `quiet` level, display on the console a single message for each service started, and record in `/var/svc/log/svc.startd.log` information about errors that do not require administrative intervention.
`debug`	In addition to the messaging provided at the `quiet` level, display on the console a single message for each service started, and record any `svc.startd` debug messages in `/var/svc/log/svc.startd.log`.

Specifying the SMF Milestone to Which to Boot

When you boot a system, you can specify the SMF milestone to which to boot.

By default, all services for which the value of the `general/enabled` property is `true` are started at system boot. To change the milestone to which to boot a system, use one of the following methods. The value of *milestone* can be the FMRI of a milestone service or a keyword as shown in Table 3, "SMF Boot Milestones and Corresponding Run Levels," on page 127.

- When booting a SPARC system, specify the `-m` option to the `boot` command at the `ok` prompt. See the `-m` option in the `kernel`(8) man page.

  ```
  ok boot -m milestone=milestone
  ```
- When booting an x86 system, edit the GRUB menu to specify the `-m` option. See "Adding Kernel Arguments at Boot Time" in *Booting and Shutting Down Oracle Solaris 11.4 Systems* and the `-m` option in the `kernel`(8) man page.
- Prior to rebooting a system, use the `svcadm milestone` command with the `-d` option. Note that with or without the `-d` option, this command restricts and restores running services immediately. With the `-d` option, the command also makes the specified milestone the default boot milestone. This new default is persistent across reboots.

  ```
  $ svcadm milestone -d milestone
  ```

  This command does not change the current run level of the system. To change the current run level of the system, use the `init` command.

  If you specify the `-s` option, `svcadm` changes the milestone and then waits for the transition to the specified milestone to complete before returning. The `svcadm` command returns when all instances have transitioned to the state necessary to reach the specified milestone or when it determines that administrator intervention is required to make a transition. Use the `-T` option with the `-s` option to specify an upper bound in seconds to complete the milestone change operation or return.

The following table describes SMF boot milestones, including any corresponding Oracle Solaris run level. A system's *run level* defines what services and resources are available to users. A system can be in only one run level at a time. For information about run levels,see "How Run Levels Work" in *Booting and Shutting Down Oracle Solaris 11.4 Systems*, the `inittab`(5) man page, and the `/etc/init.d/README` file. For more information about SMF boot milestones, see the `milestone` subcommand in the `svcadm`(8) man page.

TABLE 3 SMF Boot Milestones and Corresponding Run Levels

SMF Milestone FMRI or Keyword	Corresponding Run Level	Description
`none`		The `none` keyword represents a milestone where no services are running except for the master restarter. When `none` is specified, all services except for `svc:/system/svc/restarter:default` are temporarily disabled.

SMF Milestone FMRI or Keyword	Corresponding Run Level	Description
		The none milestone can be useful when for debugging startup problems. See "How to Investigate Problems Starting Services at System Boot" on page 129 for specific instructions.
`all`		The `all` keyword represents a milestone that depends on every service. When `all` is specified, temporary enable and disable requests are ignored for all services. This is the default milestone used by `svc.startd`.
`svc:/milestone/single-user`	s or S	Ignore temporary enable and disable requests for `svc:/milestone/single-user:default` and all services on which it depends either directly or indirectly. Temporarily disable all other services.
`svc:/milestone/multi-user`	2	Ignore temporary enable and disable requests for `svc:/milestone/multi-user:default` and all services on which it depends either directly or indirectly. Temporarily disable all other services.
`svc:/milestone/multi-user-server`	3	Ignore temporary enable and disable requests for `svc:/milestone/multi-user-server:default` and all services on which it depends either directly or indirectly. Temporarily disable all other services.

To determine the milestone to which a system is currently booted, use the `svcs` command. The following example shows that the system is booted to run level 3, `milestone/multi-user-server`:

```
$ svcs 'milestone/*'
STATE          STIME    FMRI
online          9:08:05 svc:/milestone/unconfig:default
online          9:08:06 svc:/milestone/config:default
online          9:08:07 svc:/milestone/devices:default
online          9:08:25 svc:/milestone/network:default
online          9:08:31 svc:/milestone/single-user:default
online          9:08:51 svc:/milestone/name-services:default
online          9:09:13 svc:/milestone/self-assembly-complete:default
online          9:09:23 svc:/milestone/multi-user:default
online          9:09:24 svc:/milestone/multi-user-server:default
```

Using SMF to Investigate System Boot Problems

This section describes actions to take if your system hangs during boot or if a key service fails to start during boot.

▼ How to Investigate Problems Starting Services at System Boot

If problems occur when starting services at system boot, sometimes the system will hang during boot. This procedure shows how to investigate services problems that occur at boot time.

1. **Boot without starting any services.**

   The following command instructs the `svc.startd` daemon to temporarily disable all services and start `sulogin` on the console.

   ```
   ok boot -m milestone=none
   ```

   See "Specifying the SMF Milestone to Which to Boot" on page 126 for a list of SMF milestones that you can use with the `boot -m` command.

2. **Log in to the system as `root`.**

3. **Enable all services.**

   ```
   # svcadm milestone all
   ```

4. **Determine where the boot process is hanging.**

   When the boot process hangs, determine which services are not running by running `svcs -a`. Look for error messages in the log files in `/var/svc/log`.

5. **After fixing the problems, verify that all services have started.**

   a. **Verify that all needed services are online.**

      ```
      # svcs -x
      ```

   b. **Verify that the `console-login` service dependencies are satisfied.**

      This command verifies that the `login` process on the console will run.

      ```
      # svcs -l system/console-login:default
      ```

6. **Continue the normal booting process.**

▼ How to Force Single-User Login if the Local File System Service Fails During Boot

Local file systems that are not required to boot the system are mounted by the `svc:/system/filesystem/local:default` service. When any of those file systems cannot be mounted, the `filesystem/local` service enters a maintenance state. System startup continues, and any services that do not depend on `filesystem/local` are started. Services that have a required dependency on the `filesystem/local` service are not started.

This procedure explains how to change the configuration of the system so that a `sulogin` prompt appears immediately after the service fails instead of allowing system startup to continue.

1. **Modify the `system/console-login` service.**

```
$ svccfg -s svc:/system/console-login
svc:/system/console-login> addpg site,filesystem-local dependency
svc:/system/console-login> setprop site,filesystem-local/entities = fmri: svc:/system/
filesystem/local
svc:/system/console-login> setprop site,filesystem-local/grouping = astring: require_all
svc:/system/console-login> setprop site,filesystem-local/restart_on = astring: none
svc:/system/console-login> setprop site,filesystem-local/type = astring: service
svc:/system/console-login> end
```

2. **Refresh the service.**

```
$ svcadm refresh console-login
```

When a failure occurs with the `system/filesystem/local:default` service, use the `svcs -vx` command to identify the failure. After the failure has been fixed, use the following command to clear the error state and allow the system boot to continue:

```
$ svcadm clear filesystem/local
```

Converting `inetd` Services to SMF Services

The `inetd.conf` file on your system should contain no entries. The `inetd.conf` file should contain only comments that this is a legacy file no longer directly used. If the `inetd.conf` file contains any entries, follow the instructions in this section to convert these configurations to SMF services. Services that are configured in the `inetd.conf` file but are not configured as an SMF service are not available for use. Services that are configured in the `inetd.conf` file are not restarted by the `inetd` command directly. Rather, the `inetd` command is the delegated restarter for the converted services.

During initial system boot, configurations in the `inetd.conf` file are automatically converted to SMF services. After initial system boot, entries might be added to the `inetd.conf` file by installing additional software that is not delivered by Image Packaging System (IPS) packages. Software that is delivered by IPS packages includes any required SMF manifest, and that SMF manifest instantiates that service instance with the correct property values.

If the `inetd.conf` file on your system contains any entries, use the `inetconv` command to convert those configurations to SMF services. The `inetconv` command converts `inetd.conf` entries into SMF service manifest files and imports those manifests into the SMF repository to instantiate the service instances. See the `inetconv`(8) man page for information about command options and to see examples of using the command.

The name of the new SMF manifest incorporates the *service_name* from the `inetd.conf` entry. The entry from the `inetd.conf` file is saved as a property of the new service instance. The new SMF manifest specifies property groups and properties to define the actions listed in the `inetd.conf` entry. After running the `inetconv` command, use the `svcs` and `svcprop` commands to ensure the new service instance was created and has the correct property values.

The `inetd` command is the delegated restarter for SMF internet services. Do not use the `inetd` command directly to manage these services. Use the `inetadm` command with no options or operands to see a list of services that are controlled by `inetd`. Use the `inetadm`, `svcadm`, and `svccfg` commands to configure and manage these converted services.

The `inetconv` command does not modify the input `inetd.conf` file. You should manually delete any entries in the `inetd.conf` file after successfully running `inetconv`.

For information about configuring `inetd` services that are already converted to SMF services, see "Modifying Services that are Controlled by `inetd`" on page 101.

Index