Managing System Services in Oracle®
Solaris 11.2
This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
Contents

Using This Documentation .............................................................................................................. 13

1 Introduction to the Service Management Facility ................................................................. 15
   SMF Capabilities .................................................................................................................. 15
   New Features in This Release ....................................................................................... 16
   SMF Concepts and Components .................................................................................. 17
      SMF Service .............................................................................................................. 18
      Service Models ........................................................................................................... 20
      Service Names ............................................................................................................ 20
      Service States .............................................................................................................. 21
      Service Dependencies ............................................................................................. 21
      Service Restarters ..................................................................................................... 22
      Service Properties and Property Groups ................................................................. 23
      Service Configuration Repository ............................................................................ 23
   Configuration Files and SMF Services ......................................................................... 27
   Service Management Privileges .................................................................................... 29

2 Getting Information About Services ..................................................................................... 31
   Listing Services on the System ..................................................................................... 31
       Showing Service State .............................................................................................. 31
       Showing More Information About Services .......................................................... 32
       Showing Selected Service Information .................................................................. 33
   Showing Service Dependencies ..................................................................................... 34
      Dependency Groupings ............................................................................................. 35
      Listing Instances That a Service Depends On .......................................................... 36
      Listing Instances That Depend on a Service .............................................................. 37
      Showing Whether a Service Will Automatically Restart ......................................... 38
   Getting More Information About Service States ............................................................. 39
   Viewing Service Log Files ............................................................................................. 40
   Inspecting Service Configuration ................................................................................... 41
Contents

Showing Descriptions of Properties and Property Groups ........................................ 42
Showing Service and Instance Property Values ..................................................... 42
Showing Properties in a Property Group Type .................................................... 45
Showing the Layer Where a Value Is Set .......................................................... 46
Showing Values in a Specified Snapshot ......................................................... 47
Showing Configuration Customizations ............................................................. 47
Showing Event Notification Parameters ............................................................ 48

3 Administering Services ................................................................................... 51
Managing SMF Service Instances ...................................................................... 51
Starting a Service ............................................................................................ 51
Stopping a Service ......................................................................................... 54
Restarting a Service ....................................................................................... 56
Rereading Service Configuration ..................................................................... 57
Deleting a Service ........................................................................................... 58
Configuring Notification of State Transition and FMA Events ......................... 59

4 Configuring Services ..................................................................................... 63
Using the Service Configuration Command ....................................................... 63
Invoking a Property Editor ............................................................................. 64
Invoking svccfg Interactively or With a File ................................................... 65
Setting Property Values ................................................................................... 66
▼  How to Modify a ttymon Property Value .................................................. 68
▼  How to Modify an Environment Variable for a Service Process
Environment .................................................................................................. 69
Adding Property Groups, Properties, and Property Values ............................... 70
Deleting Property Groups, Properties, and Property Values ............................ 72
Deleting Administrative Configuration ........................................................... 72
Deleting Non-Administrative Configuration .................................................... 74
Adding Service Instances ................................................................................ 76
Reverting Snapshots ....................................................................................... 76
Importing and Applying Manifests and Profiles ................................................. 77
Configuring Multiple Systems ......................................................................... 77
▼  How to Create a Profile by Using svcbundle ............................................. 78
▼  How to Create a Profile by Using svccfg .................................................. 79
Modifying Services that are Controlled by inetd .............................................. 79
▼  How to Change a Property Value for an inetd Controlled Service ............ 80
Modifying Services that are Configured by a File ............................................ 81
5 Using SMF to Control Your Application ............................................ 83
   Creating an SMF Service ................................................................. 83
      ▼ How to Create an SMF Service Using the Service Bundle Generator Tool ......................................................... 84
   Creating a Service to Start or Stop an Oracle Database Instance .......... 85
   Naming Services, Instances, Property Groups, and Properties ............. 91
      ▼ How to Convert a Run Control Script to an SMF Service .......... 92
   Using a Stencil to Create a Configuration File .................................. 93
      ▼ How to Create a Stencil Service .................................................. 94
   Stencil Service Examples in Oracle Solaris ........................................ 95

A SMF Best Practices and Troubleshooting ......................................... 101
   SMF Best Practices ........................................................................ 101
   Troubleshooting Services Problems .............................................. 102
      Understanding Configuration Changes ........................................ 102
      Repairing an Instance That Is Degraded, Offline, or in Maintenance ...... 103
      Marking an Instance as Degraded or in Maintenance ..................... 106
      Diagnosing and Repairing Repository Problems ............................ 107
      Specifying the Amount of Startup Messaging ............................... 110
      Specifying the SMF Milestone to Which to Boot ........................... 111
      Using SMF to Investigate System Boot Problems .......................... 112
   Converting inetd Services to SMF Services ....................................... 114

Index ....................................................................................................... 117
Figures

FIGURE 1-1  Service Management Facility Framework ............................................. 18
FIGURE 2-1  Service Dependency Relationships ...................................................... 35
# Tables

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Automatically Restarting a Service After a Dependency Stop</td>
<td>38</td>
</tr>
<tr>
<td>A-1</td>
<td>SMF Startup Message Logging Levels</td>
<td>110</td>
</tr>
<tr>
<td>A-2</td>
<td>SMF Boot Milestones and Corresponding Run Levels</td>
<td>112</td>
</tr>
</tbody>
</table>
# Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 2-1</td>
<td>Listing All Enabled Services</td>
<td>31</td>
</tr>
<tr>
<td>Example 2-2</td>
<td>Listing All Installed Services</td>
<td>32</td>
</tr>
<tr>
<td>Example 2-3</td>
<td>Listing All Instances of a Service</td>
<td>32</td>
</tr>
<tr>
<td>Example 2-4</td>
<td>Showing Processes Started by a Contract Service</td>
<td>33</td>
</tr>
<tr>
<td>Example 2-5</td>
<td>Showing a Contract Service Restarting Automatically After Process Stop</td>
<td>33</td>
</tr>
<tr>
<td>Example 2-6</td>
<td>Listing Instance and Inherited Properties Currently in Use</td>
<td>42</td>
</tr>
<tr>
<td>Example 2-7</td>
<td>Listing Specified Properties or Property Groups Currently in Use</td>
<td>43</td>
</tr>
<tr>
<td>Example 2-8</td>
<td>Listing Service and Instance Values in the Editing View</td>
<td>43</td>
</tr>
<tr>
<td>Example 2-9</td>
<td>Listing Specified Properties or Property Groups in the Editing View</td>
<td>44</td>
</tr>
<tr>
<td>Example 2-10</td>
<td>Showing Property Groups and Their Types</td>
<td>45</td>
</tr>
<tr>
<td>Example 2-11</td>
<td>Listing Properties of a Property Group Type</td>
<td>45</td>
</tr>
<tr>
<td>Example 3-1</td>
<td>Enabling a Service Instance Permanently</td>
<td>53</td>
</tr>
<tr>
<td>Example 3-2</td>
<td>Enabling a Service Instance Temporarily</td>
<td>53</td>
</tr>
<tr>
<td>Example 3-3</td>
<td>Disabling a Service Instance</td>
<td>55</td>
</tr>
<tr>
<td>Example 3-4</td>
<td>Configuring a Global Notification for a Service State Event</td>
<td>60</td>
</tr>
<tr>
<td>Example 3-5</td>
<td>Configuring a Notification for a Specified Service Instance</td>
<td>60</td>
</tr>
<tr>
<td>Example 3-6</td>
<td>Configuring a Notification for an FMA Event</td>
<td>61</td>
</tr>
<tr>
<td>Example 3-7</td>
<td>Deleting Notification Settings</td>
<td>61</td>
</tr>
<tr>
<td>Example 4-1</td>
<td>Setting a Simple Value</td>
<td>66</td>
</tr>
<tr>
<td>Example 4-2</td>
<td>Setting a Value that Contains a Colon Character</td>
<td>66</td>
</tr>
<tr>
<td>Example 4-3</td>
<td>Setting a Value that Contains Embedded Spaces</td>
<td>67</td>
</tr>
<tr>
<td>Example 4-4</td>
<td>Setting a Value that Is a Set of Values</td>
<td>67</td>
</tr>
<tr>
<td>Example 4-5</td>
<td>Adding a Value</td>
<td>67</td>
</tr>
<tr>
<td>Example 4-6</td>
<td>Using <code>addpg</code> to Create a New Property Group</td>
<td>71</td>
</tr>
<tr>
<td>Example 4-7</td>
<td>Using <code>setprop</code> to Create a New Property</td>
<td>71</td>
</tr>
<tr>
<td>Example 4-8</td>
<td>Using <code>addpropvalue</code> to Create a New Property</td>
<td>72</td>
</tr>
<tr>
<td>Example 4-9</td>
<td>Deleting All Values of a Property</td>
<td>73</td>
</tr>
<tr>
<td>Example 4-10</td>
<td>Deleting All Matching Values of a Property</td>
<td>73</td>
</tr>
<tr>
<td>Example 4-11</td>
<td>Deleting a Property</td>
<td>73</td>
</tr>
<tr>
<td>EXAMPLE 4-12</td>
<td>Deleting a Property Group .......................................................... 74</td>
<td></td>
</tr>
<tr>
<td>EXAMPLE 4-13</td>
<td>Deleting Customizations ................................................................. 74</td>
<td></td>
</tr>
<tr>
<td>EXAMPLE 4-14</td>
<td>Deleting Configuration that has Bundle Support ............................... 75</td>
<td></td>
</tr>
<tr>
<td>EXAMPLE 4-15</td>
<td>Unmasking Configuration ................................................................. 75</td>
<td></td>
</tr>
<tr>
<td>EXAMPLE 4-16</td>
<td>Automatically Installing a Profile by Using svcbundle .......................... 78</td>
<td></td>
</tr>
<tr>
<td>EXAMPLE 4-17</td>
<td>Modifying the Command to Execute When an inetd Controlled Service Starts ................................................................. 81</td>
<td></td>
</tr>
<tr>
<td>EXAMPLE 5-1</td>
<td>Automatically Installing a Generated Manifest .................................. 85</td>
<td></td>
</tr>
</tbody>
</table>
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

Late-breaking information and known issues for this product are included in the documentation library at http://www.oracle.com/pls/topic/lookup?ctx=E36784.

Access to Oracle Support

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

Feedback

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

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 services in parallel according to the dependencies of the services.
- **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.

Create new services. Easily create a new instance of an existing service, copy and modify an existing service, or use a service creation tool.

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.

Configure how you will be notified of particular software events or hardware faults.

Convert `inetd.conf` configurations to SMF services.

Convert 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.

---

**New Features in This Release**

The following SMF features are new in this release:

Synchronous restart and refresh operations

A new `-s` option for the `restart` and `refresh` subcommands of the `svcadm` command specifies synchronous operation similar to the existing `-s` option for the `enable` and `disable` subcommands.

Show log files

A new `-L` option for the `svcs` command shows the complete log file, the last few lines of the log file, or the full path name of the log file for the specified service instance.

Batch operation on a set of service instances

New `libscf(3LIB)` function calls allow common administrative operations to be grouped and executed in a batch either synchronously or asynchronously. You can enable, disable, restart, refresh, mark, clear the maintenance or degraded state, and transition between milestones for a set of service instances.

Conversion of SMF service properties to configuration files

For services that are managed by SMF but interact with applications that still require configuration files, a new stenciling capability creates and updates a configuration file from property values defined for the service in the service configuration repository. A new file called a stencil file contains the structural definition of the configuration file that will be created by SMF. A new `svcio` utility generates the configuration file from the definitions in the stencil file and properties in the SMF service. This feature enables you to take advantage of SMF configuration management with minimal change to the existing application.
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 service instances that are enabled. In the following figure, \texttt{libscf} is the library interface that the restarters use to interact with the service configuration repository. Interaction between the service configuration repository and \texttt{libscf} library interfaces is managed by the \texttt{svc.configd} daemon. The \texttt{svcs}, \texttt{svcprop}, \texttt{svcadm}, and \texttt{svccfg} commands are the interface that administrators use to interact with the service configuration repository.
SMF Concepts and Components

**FIGURE 1-1** Service Management Facility Framework

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

Chapter 1 • Introduction to the Service Management Facility

- 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 system init state

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 25. See “Service Bundles” on page 24 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 “Configuring Multiple Systems” on page 77.

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 25.

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 22.

A milestone service is a special type of service that represents a level of system readiness. 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.
Service Models

SMF services are one of the following three 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.

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.
- **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.
- **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 services.

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. See the smf(5) 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 machines 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 `smf_restarter(5)` 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.

```sh
$ 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(1M)` 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.

An example of a delegated restarter is `inetd`, which 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(1M)` 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
```

The delegated restarter specified by the `restarter` property is responsible for managing the service instance once that restarter is available.

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.

See the `smf(5)` 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 such as the
current state of each service instance on the system and the properties of each service and service instance.

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 25.
- 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(5) 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 25. For information about snapshots, see “Repository Snapshots” on page 26. 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 contain the complete set of properties associated with a specific set of services or service instances.

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 25 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(5) 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, system profiles, site 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 in the running configuration: whether a value was assigned in the manifest, in a profile, or was changed by an administrator.

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, the information about that property includes the name of that file.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>Any changes that are made by using the SMF commands or library interfaces, by an administrator or by an application. The admin layer also includes any changes that are made by importing a manifest or applying a profile from a non-standard location. See “Importing and Applying Manifests and Profiles” on page 77 for caution about the use of non-standard locations.</td>
</tr>
<tr>
<td>site-profile</td>
<td>Any values from profile files in the /etc/svc/profile/site directory or the legacy /etc/svc/profile/site.xml and /var/svc/profile/site.xml profiles. Note that /var/svc/profile is deprecated as a standard location and should not be used for new profiles.</td>
</tr>
<tr>
<td>system-profile</td>
<td>Any values from the /etc/svc/profile/generic.xml and /etc/svc/profile/platform.xml system profiles.</td>
</tr>
<tr>
<td>Layer</td>
<td>Content</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>manifest</td>
<td>Values from manifests in the /lib/svc/manifest and /var/svc/manifest directories. Note that /var/svc/manifest is deprecated as a standard location and should not be used for new manifests.</td>
</tr>
</tbody>
</table>

Property conflicts are not permitted within any layer. A conflicting property in the admin layer overwrites the previous property. If the same property is delivered by multiple files in any other layer, and is not set at a higher layer, the instance is tagged as in-conflict and is not started until the conflicting definition is removed or the property is set at a higher layer.

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 following priority order from top to bottom: admin layer, site-profile layer, system-profile layer, manifest 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 108.

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(4) man page. For instructions to convert a run control script, see “How to Convert a Run Control Script to an SMF Service” on page 92. After you convert an rc?.d script, rename the script from Sscript to sscript 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(4) man page.

Legacy init.d run control scripts are represented with SMF FMRIIs that begin with lrc instead of svc. For example, the /etc/rc2.d/S47pppd PPP configuration script is represented by the lrc:/etc/rc2_d/S47pppd service. 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.
Configuration Files and SMF Services

```
$ svcs lrc:*

STATE       STIME       FMRI
legacy_run   9:34:54     lrc:/etc/rc2_d/S47pppd
legacy_run   9:34:54     lrc:/etc/rc2_d/S89PRESERVE

$ svccfg -s lrc:/etc/rc2_d/S47pppd listprop
svccfg: Operation not supported for legacy service 'lrc:/etc/rc2_d/S47pppd'

$ svccfg -s lrc:/etc/rc2_d/S47pppd listprop
svccfg: Operation not supported for legacy service 'lrc:/etc/rc2_d/S47pppd'
```

/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 68 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(4)` man page. For information about run levels, see the inittab(4) 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 114. For inetd services that are already converted to SMF services, see “Modifying Services that are Controlled by inetd” on page 79.

/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 66.

The function of the nscd.conf file is replaced by the svc:/system/name-service-cache SMF service. See the nscd.conf(4) 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(4) 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(4) 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 81. 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 11.2 ” for more information about authorizations, profiles, and roles, including how to determine which profile or role you need and how to assign privileges.

Authorizations

See the smf_security(5) man page for detailed information about authorizations required for SMF operations. You can also inspect a particular service for properties such as action_authorization, modify_authorization, read_authorization, and value_authorization.

Rights profiles

Use the profiles command to list the rights profiles that are assigned to you. The Service Management rights profile grants you the solaris.smf.manage and solaris.smf.modify authorities and enables you to use the svcadm and svccfg commands. The Service Operator rights profile grants you the solaris.smf.manage and solaris.smf.modify.framework authorities.

Roles

Use the roles command to list the roles that are assigned to you. If you have the root role, you can use the su command to assume the root role.

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.
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 21 for descriptions of the states shown in these examples.

**EXAMPLE 2-1**  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 27 for more information about these legacy services. See “Getting More Information About Service States” on page 39 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.

```
$ svcs
STATE  STIME    FMRI
legacy_run Sep_09  /etc/rc2_d/S47pppd
```
Listing Services on the System

Listing Services on the System

EXAMPLE 2-2  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 2-3  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 33 for information about the -o option.

```
$ svcs -Ho inst identity
node
domain
```

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 all of the required dependencies of this service instance are online. The one dependency that is disabled is an optional dependency. For information about dependency types and restart attribute values, see “Showing Service Dependencies” on page 34. In svcs -l output, states other than those described in “Service States” on page 21 are possible for dependencies. See the SVCS(1) man page for descriptions. The following example also shows that the specified service instance is temporarily enabled, is online, and the service is a contract type service. See “Service Models” on page 20 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.

```
$ svcs -l net-snmp
fmri          svc:/application/management/net-snmp:default
```
Listing Services on the System

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:/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)

EXAMPLE 2-4  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         17:57:26 svc:/application/management/net-snmp:default
17:57:26     5022 snmpd

EXAMPLE 2-5  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 5022
$ svcs -p net-snmp
STATE          STIME    FMRI
online         17:57:59 svc:/application/management/net-snmp:default
17:57:59     5037 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.

**Showing Service Dependencies**

Dependency relationships govern service instance state transitions. See “Service Dependencies” on page 21 for a high-level description of dependencies. See Chapter 5, “Using SMF to Control Your Application” 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.
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   Sep_09           svc:/milestone/network:default
online   Sep_09           svc:/system/filesystem/local:default
online   Sep_09           svc:/system/filesystem/autofs: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 35 for descriptions of how dependencies in these groupings affect the dependent service. See “Dependency Groupings” on page 35 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      false
state        disabled
next_state   none
state_time   Mon Sep 09 18:42:28 2013
restarter    svc:/system/svc/restarter:default
manifest     /lib/svc/manifest/application/pkg/pkg-system-repository.xml
```
Showing Service Dependencies

Chapter 2 • Getting Information About Services

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
network/entities fmri svc:/milestone/network:default
network/grouping astring require_all
network/restart_on astring error
network/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
autofs/entities fmri svc:/system/filesystem/autofs:default
autofs/grouping astring optional_all
autofs/restart_on astring error
autofs/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
online         16:39:30 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. This output also shows that the `zones-proxyd` service will be restarted if the `system-repository` service is refreshed.

```
$ svcs -l zones-proxyd
fmri         svc:/application/pkg/zones-proxyd:default
name         Zones Proxy Daemon
enabled      true
state        online
```
Showing Service Dependencies

next_state   none
state_time   January 6, 2014 04:39:30 PM PST
restarter    svc:/system/svc/restarter:default
dependency   require_any/none svc:/system/filesystem/minimal (online)
dependency   require_any/error svc:/milestone/network (online)
dependency   require_all/restart svc:/application/pkg/system-repository (online)

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>
<thead>
<tr>
<th>require_all, require_any, or optional_all Dependency</th>
<th>Value of Dependency restart_on Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop or Refresh Event</td>
<td>none</td>
</tr>
<tr>
<td>Stop due to error</td>
<td>No restart</td>
</tr>
<tr>
<td>Other stop</td>
<td>No restart</td>
</tr>
<tr>
<td>Refresh</td>
<td>No restart</td>
</tr>
</tbody>
</table>
“Listing Instances That a Service Depends On” on page 36 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.

```
$ svccfg -s system-repository:default listprop -o proponame,propval '*restart_on'
  network/restart_on          astring     error
  filesystem-local/restart_on astring     none
  autofs/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(1M)
  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 40 for information about different ways to view log files. See “Repairing an Instance That Is Degraded, Offline, or in Maintenance” on page 103 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.
```
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 Sep_09   lrc:/etc/rc2.d/S47pppd
legacy_run Sep_09   lrc:/etc/rc2.d/S81dodatadm_udaplt
legacy_run Sep_09   lrc:/etc/rc2.d/S89PRESERVE
disabled     Sep_09   svc:/system/vbiosd:default
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
```

### 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.

**EXAMPLE 2-6  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 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.

```bash
$ 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.

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

**EXAMPLE 2-7**  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.

```bash
$ svcprop -p pkg/port pkg/server
svc:/application/pkg/server:oss:properties/pkg/port count 82
svc:/application/pkg/server:s11:properties/pkg/port count 81
$ svcprop -p pkg pkg/server:s11
pkg/inst_root astring /export/ipsrepos/Solaris11
pkg/port count 81
... pkg/ssl_cert_file astring ""
pkg/ssl_key_file astring ""
... 
```

**EXAMPLE 2-8**  Listing Service and Instance Values in the Editing View

With options, the `svcprop` command can show 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 or restarting the instance. The following commands illustrate the difference between the running snapshot and the editing view. The oss and s11 instances have just been created and property values have been changed, but the instances have not yet been refreshed. The first command shows the composed view of the running snapshot. Because these instances have not been refreshed since they were customized, the values shown are values from the pkg/server service. The `-c` option shows the composed view of the editing values. The `-C` option shows the editing values without
composition. Because this is not a composed view, no value is found for the instance that has not been customized.

```
$ svcprop -p pkg/port pkg/server
svc:/application/pkg/server:oss/:properties/pkg/port count 80
svc:/application/pkg/server:s11/:properties/pkg/port count 80
svc:/application/pkg/server:default/:properties/pkg/port count 80

$ svcprop -c -p pkg/port pkg/server
svc:/application/pkg/server:oss/:properties/pkg/port count 82
svc:/application/pkg/server:s11/:properties/pkg/port count 81
svc:/application/pkg/server/:properties/pkg/port count 80

$ svcprop -C -p pkg/port pkg/server
svc:/application/pkg/server:oss/:properties/pkg/port count 82
svc:/application/pkg/server:s11/:properties/pkg/port count 81
svcprop: Couldn't find property 'pkg/port' for instance 'svc:/application/pkg/server:default'.
```

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 47.

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 47.

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 2-9**  
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(1M)` man page for the list of valid column names.

```
$ svccfg -s pkg/server:s11 listprop pkg
pkg application
```
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 2-10  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
pkg             application
pkg_bui         application
pkg_secure      application
fs              dependency
autofs          dependency
ntp             dependency
network         dependency
general         framework
manifestfiles   framework
start           method
stop            method
tm_common_name  template
```  

```
$ svccfg -s pkg/server:s11 listpg
pkg        application
general    framework
restarter  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 2-11  Listing Properties of a Property Group Type

Use the -g option of the svcprop command to show properties in a specific property group type. Property group types include application, dependency, method, framework, and template.
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, site-profile, and admin. The output indicates whether a specific property value was set in the service manifest, a profile, or by an administrator. See “Repository Layers” on page 25 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, some were set by an administrator, and 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.

```bash
$ svccfg -s pkg/server:s11 listprop -l current pkg                                application admin
pkg/inst_root                     astring     admin     /export/ipsrepos/Solaris11
pkg/port                          count       admin     81
```

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.

```bash
$ svccfg -s pkg/server:s11 listprop -l current pkg
pkg/application admin
pkg/inst_root                     astring     admin     /export/ipsrepos/Solaris11
pkg/port                          count       admin     81
```
### 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 `svccfg` or `svcprop` 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 26 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 site-profile layer. The
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(5) 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
Showing Event Notification Parameters

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 59 for information about configuring event notification.
Administering Services

This chapter describes how to start, stop, and restart a service and how to reread service configuration. This chapter also describes how to configure the system to notify you of FMA events or service state transitions. These changes are admin level customizations as described in “Repository Layers” on page 25.

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 21. 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 103. 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.
How to Enable a Service Instance

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 103.
Example 3-1  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
```

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 3-2  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
```

```
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.

```bash
$ svcadm enable -t net-snmp:default
$ svcs -l net-snmp
```

<table>
<thead>
<tr>
<th>fmri</th>
<th>svc:/application/management/net-snmp:default</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>net-snmp SNMP daemon</td>
</tr>
<tr>
<td>enabled</td>
<td>true (temporary)</td>
</tr>
<tr>
<td>state</td>
<td>online</td>
</tr>
<tr>
<td>next_state</td>
<td>none</td>
</tr>
<tr>
<td>state_time</td>
<td>September 17, 2013 05:57:26 PM PDT</td>
</tr>
<tr>
<td>logfile</td>
<td>/var/svc/log/application-management-net-snmp.default.log</td>
</tr>
<tr>
<td>restarter</td>
<td>svc:/system/svc/restarter:default</td>
</tr>
<tr>
<td>contract_id</td>
<td>160</td>
</tr>
<tr>
<td>manifest</td>
<td>/etc/svc/profile/generic.xml</td>
</tr>
<tr>
<td>manifest</td>
<td>/lib/svc/manifest/application/management/net-snmp.xml</td>
</tr>
<tr>
<td>dependency</td>
<td>require_all/none svc:/system/filesystem/local (online)</td>
</tr>
<tr>
<td>dependency</td>
<td>optional_all/none svc:/system/system-log (online)</td>
</tr>
<tr>
<td>dependency</td>
<td>optional_all/none svc:/network/rpc/rstat (disabled)</td>
</tr>
<tr>
<td>dependency</td>
<td>require_all/restart svc:/system/cryptosvc (online)</td>
</tr>
<tr>
<td>dependency</td>
<td>require_all/restart svc:/milestone/network (online)</td>
</tr>
<tr>
<td>dependency</td>
<td>require_all/refresh file://localhost/etc/net-snmp/snmp/snmpd.conf (online)</td>
</tr>
<tr>
<td>dependency</td>
<td>require_all/none svc:/milestone/multi-user (online)</td>
</tr>
</tbody>
</table>

```bash
$ svcs -p net-snmp
```

<table>
<thead>
<tr>
<th>STATE</th>
<th>STIME</th>
<th>FMRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>online</td>
<td>17:57:26</td>
<td>svc:/application/management/net-snmp:default</td>
</tr>
<tr>
<td></td>
<td>17:57:26</td>
<td>5022 snmpd</td>
</tr>
</tbody>
</table>

### 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.

#### How to Disable a Service Instance

1. **Check whether other services depend on this instance.**
   a. **List services that depend on this instance.**
      ```bash
      $ 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.

2. **Disable the instance.**

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

   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 3-3** Disabling a Service Instance

   This example shows that the `pkg/update:default` service instance is initially online and no other services depend on this instance. The `svcadm disable` command is successful, the instance is currently in the disabled state, and the restart attempt fails.
How to Restart a Service Instance

$ svcs pkg/update
STATE STIME FMRI
online 7:18:17 svc:/application/pkg/update:default

$ svcs -D pkg/update:default

$ svcadm disable pkg/update

$ svcs pkg/update
STATE STIME FMRI
disabled 22:51:12 svc:/application/pkg/update:default

$ svcadm restart pkg/update:default

$ svcs pkg/update
STATE STIME FMRI
disabled 22:51:12 svc:/application/pkg/update: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. The svcadm restart command runs the stop method of the instance and then runs the start method of the instance. 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 57 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.
How to Restart a Service Instance

Chapter 3 • Administering Services

$ 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 103.

$ svcs FMRI

Rereading Service Configuration

When you change service configuration, those changes do not immediately appear in the running snapshot. Those changes are stored in the service configuration repository as current, or editing, property values. The refresh operation updates the running snapshot of the specified service instance with the values from the editing configuration.

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 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 56. 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(5) 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 59.

#### 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.

**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.
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.

The following examples show how to set notification parameters for SMF and FMA events and how to delete notification parameters.

**EXAMPLE 3-4**  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(5) man page. Specifying only the state name includes both to-state and from-state transitions. This event could 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 3-5**  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:apache22 service transitions out of the online state.

```
$ svccfg -s http:apache22 setnotify from-online mailto:webservices@example.com
```
EXAMPLE 3-6  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(5)` man page.

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

EXAMPLE 3-7  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
```
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 41. For committing configuration modifications into the running snapshot, see “Rereading Service Configuration” on page 57.

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 also shows how to modify an `inetd` service.

SMF configuration changes can be logged by using the Oracle Solaris auditing framework. Refer to “Configuring the Audit Service (Task Map)” in Managing Auditing in Oracle Solaris 11.2 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 25 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.

```bash
$ 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
```
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 4-1 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 4-2 Setting a Value that Contains a Colon Character

If the property value contains a colon character (:`), then specify the property type as shown in the following example where the type is `astring`:

$ svccfg -s system-repository:default setprop config/http_proxy = astring: https://proxyURI

Use the `listprop` subcommand to find the type of the property you want to set.

$ svccfg -s system-repository:default listprop config/http_proxy
config/http_proxy astring
EXAMPLE 4-3  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 4-4  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 enclose the value set in single quotation marks as well.

$ 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 4-5  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
dvalues, 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 Modify a ttymon Property Value

This procedure shows how to modify parameters passed to ttymon.

1. Identify the service to modify.

   The ttymon(1M) man page states that the service to modify is svc:/system/console-login. The ttymon(1M) 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:
Adding Property Groups, Properties, and Property Values

Chapter 4 • Configuring Services

71

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 4-6 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 Chapter 5, “Using SMF to Control Your Application” 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.

EXAMPLE 4-7 Using setprop to Create a New Property

Use the setprop subcommand to set a property value as described in “Setting Property Values” on page 66. 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/vendor = astring: vendora
$ svccfg -s my-svc:default listprop config/vendor
config/vendor astring vendora
EXAMPLE  4-8  Using addpropvalue to Create a New Property

Use the addpropvalue subcommand to add a property value as described in “Setting Property Values” on page 66. 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/vendor astring: vendorb
$ svccfg -s my-svc:default addpropvalue config/customer astring: acustomer
$ svccfg -s my-svc:default listprop config
  config      application
  config/vendor astring   "vendora" "vendorb"
  config/customer astring  acustomer

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 25.
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 74 for information about deleting configuration that exists in other layers.

**EXAMPLE 4-9**  Deleting All Values of a Property

Use the `setprop` subcommand as described in “Setting Property Values” on page 66. 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 4-10**  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 4-11**  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
```
Deleting Property Groups, Properties, and Property Values

config           application
config/customer astring     acustomer
config/vendor   astring     vendora

EXAMPLE 4-12 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 4-13 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 site-profile, system-profile, and manifest layers of the service configuration repository is defined in service manifests and profile files. See “Repository Layers” on page 25 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.
Deleting Property Groups, Properties, and Property Values

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(5) man page for a description of masked entities.

Deleting configuration that has bundle support is an administrative customization. In this case, the delcust subcommand un_masks 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 4-14 Deleting Configuration that has Bundle Support

In “Deleting Administrative Configuration” on page 72, 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 4-15 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.

```bash
$ 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.

```bash
$ 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.

```bash
$ svcprop -s previous -l admin 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 24 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 manifest, system-profile, or site-profile layer of affected instances, affected instances are refreshed and validated, and a new snapshot is created.

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.

Configuring Multiple Systems

To implement the same configuration on multiple systems, create an SMF profile that specifies the services you want enabled and the values of properties, and put that profile in the site directory on each system. As stated in “Repository Layers” on page 25, local customizations are preferred over the values that were provided when the system was installed, and customizations

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
Profiles add and set properties for existing services and instances and specify new service instances. Profiles can specify almost anything that a manifest can specify.

Use one of the following methods to create a site profile:

- Use the `svcbundle` command with `bundle-type=profile` to create a new profile file.
- Use the `svccfg extract` command to capture profile information from an existing service.

Customize property values in the profile file, and include comments about the reason for each customization. Copy the file to `/etc/svc/profile/site`, and restart the `manifest-import` service.

### How to Create a Profile by Using `svcbundle`

The `svc:/system/rmtmpfiles` service is responsible for cleaning up the `/tmp` directory on boot. By default, the `rmtmpfiles` service does not clean up `/var/tmp`. To clean up `/var/tmp` during the boot process, change the behavior of the `svc:/system/rmtmpfiles` service by setting the `options/clean_vartmp` property to `true`. The easiest way to achieve this behavior on multiple systems is to create a profile and place it in `/etc/svc/profile/site` on each system.

1. **Create the profile.**
   
   The following command creates a new profile in `/tmp/rmtmpfiles.xml`.
   
   ```bash
   $ svcbundle -o /tmp/rmtmpfiles.xml -s service-name=system/rmtmpfiles -s bundle-type=profile -s service-property=options/clean_vartmp:boolean:true
   ```

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

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

   ```bash
   $ cp /tmp/rmtmpfiles.xml /etc/svc/profile/site/rmtmpfiles.xml
   ```

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

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

**Example 4-16  Automatically Installing a Profile by Using `svcbundle`**

If you do not need to make any changes to the new profile, you can use the `-i` option to install the profile as soon as it is created. The `svcbundle` command will write the profile to `/etc/svc/`
How to Create a Profile by Using svccfg

1. **Create a profile.**
   The `svccfg extract` command displays a service profile for the specified service or instance for the `admin` and `site-profile` layers. To extract values from other layers, use the `-l` option.
   
   ```bash
   $ 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:
   
   ```xml
   <service_bundle type='profile' name='extract'>
   ```

3. **Copy the profile to the correct directory.**
   
   ```bash
   $ cp dnsclientprofile.xml /etc/svc/profile/site/dnsclientprofile.xml
   ```

4. **Restart the manifest import service to apply the profile to the system.**
   
   ```bash
   $ svcadm restart manifest-import
   ```

## 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`, invoke the `inetadm` command with no options or arguments to list all `inetd` controlled services. The following example shows only a partial list.

```bash
$ inetadm
ENABLED    STATE    FMRI
```
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.
4. **Confirm that the change has taken effect.**

    Confirm that the property change has the expected effect.

**Example 4-17**  Modifying the Command to Execute When an inetd Controlled Service Starts

This example shows how to add or remove an option or argument to the command line of a service that is controlled by inetd. The command that runs when the service starts is the value of the exec property.

Use the -l option of the/inetadm command to list all the properties of the specified service so that you can inspect the current value of the exec property. This example shows the svc:/application/x11/xfs service, which is the X Window System font server. See the xfs(1) man page for more information.

```
$ inetadm -l xfs | grep exec
exec="/usr/bin/xfs -inetd"
```

Use the -m option of the/inetadm command to change the value of the exec property of the specified service.

```
$ inetadm -m xfs exec="/usr/bin/xfs -inetd -config /opt/site/fs/config"
```

Verify that the property value is changed.

```
$ inetadm -l xfs | grep exec
exec="/usr/bin/xfs -inetd -config /opt/site/fs/config"
```

---

**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 “Using a Stencil to Create a Configuration File” on page 93.

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(1M)` man page for information about using `pfedit`.

```
$ svcadm enable svc:/network/ipsec/ike:ikev2
```
Creating an SMF Service

A service manifest contains the complete set of properties associated with a specific service, including instances, dependencies, scripts to run when the service starts and stops, and default application property values. Manifests also provide template information such as a description of the service. See the `service_bundle(4)` man page and the `/usr/share/lib/xml/dtd/service_bundle.dtd.1` service bundle DTD for a complete description of the contents and format of an SMF manifest. See also “Naming Services, Instances, Property Groups, and Properties” on page 91 for naming rules and assigning property group types.

Using the `svcbundle` service bundle generator tool is a good way to create a simple service or to start a more complex service. For more information, see the `svcbundle(1M)` man page. You can use the service bundle DTD and other service manifests to complete a more complex service.

The standard location for custom manifests is `/lib/svc/manifest/site`. Manifests stored in this location are imported into the service configuration repository by the `svc:/system/early-manifest-import:default` service during the boot process before any services start. Running the import process early ensures that the repository will contain information from the latest manifests before the services are started. Manifests stored in this standard location are also imported when the `svc:/system/manifest-import` service is restarted.

Multiple manifests can be used to describe a single service. This can be useful, for example, to define a new instance of a service without modifying the existing manifest for the service.
However, if the same property in the same layer for the same service or instance is defined by multiple manifests, 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 25 for more information about layers.

Add name and description metadata to your manifests so that users can get information about this service from the `svcs` and `svccfg describe` commands. You can also add descriptions of property values. See the `value`, `values`, and `template` elements in the DTD.

Use the `svccfg validate` command to validate your service manifest file or service instance FMRI. With your manifest, method, and profile files in standard locations, restart the `manifest-import` service to install and configure your service instances. Use the `svcs` command to check the status of your service instances.

This section shows how to create custom SMF services. The `site` prefix is reserved for site-specific customizations. A service named `svc:/site/service-name` will not conflict with the services delivered in an Oracle Solaris release.

🚨 How to Create an SMF Service Using the Service Bundle Generator Tool

This procedure shows how to create a service that uses an existing custom script as the start method.

1. **Determine the service model.**
   By default, `svcbundle` creates a transient service. Determine whether the start method script for this service starts any long running daemon and is a contract service. See “Service Models” on page 20 and the `svc.startd(1M)` man page for information about service models.

2. **Copy the script to the standard location.**
   In this example, the script that will be the start method script for this service is named `ex_svc`. Copy this script to `/lib/svc/method/ex_svc`.

3. **Create an initial manifest.**
   In this example, the service name is `site/ex_svc`. This is a transient service and does not need a stop method.

   ```bash
   $ svcbundle -o /tmp/ex_svc.xml -s service-name=site/ex_svc \\   -s start-method=/lib/svc/method/ex_svc
   ```

4. **Make any necessary changes to the manifest.**
Verify that the content of the `/tmp/ex_svc.xml` manifest is what you need. Add comments as needed.

5. **Verify that the manifest is valid.**

   Use the `svccfg validate` command to make sure the service manifest is valid.

6. **Copy the manifest to the standard directory.**

   ```
   $ cp /tmp/ex_svc.xml /lib/svc/manifest/site/ex_svc.xml
   ```

7. **Import the manifest and start the service.**

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

**Example 5-1** Automatically Installing a Generated Manifest

If you do not need to make any changes to the new service manifest, you can use the `-i` option to install the manifest as soon as it is created. The `svcbundle` command will write the manifest to `/lib/svc/manifest/site` and restart the `manifest-import` service. Any existing file with the same name in the `/lib/svc/manifest/site` directory will be overwritten.

```bash
# svcbundle -i -s service-name=site/ex_svc \
   -s start-method=/lib/svc/method/ex_svc
```

### Creating a Service to Start or Stop an Oracle Database Instance

This example presents the following services that help manage the Oracle Database:

- A database service that starts or stops an Oracle Database instance
- A listener service that starts the listener, which is a process that manages the incoming traffic of client connection requests to the database instance

This example uses file-backed storage. An alternative to using file-backed storage is to use the Automatic Storage Management (ASM) feature. ASM is a volume manager and a file system for Oracle Database files.

The following environment variables must be set for each installation of the Oracle Database:

- **ORACLE_HOME**
  The location where the database is installed. In the example in this section, the location of the database installation is `/opt/oracle/product/home`.

- **ORACLE_SID**
  The systems ID to uniquely identify a particular database on a system.
In this example, these environment variables are set in the service manifests and then used in the method scripts.

**Database Start and Stop Service**

This section shows the Oracle Database instance control service manifest, `/lib/svc/manifest/site/oracle.xml`. The following are some features to note about this service manifest:

- **One service instance is defined, named `svc:/site/application/database/oracle:default`.** This instance is enabled by default.
  This example shows two ways to define the default instance. In this example, the default instance is defined in the `create_default_instance` element at the top of the manifest. The `instance` element at the bottom of the manifest shows the other way to do this.

- **This service requires all local file systems to be mounted and all network interfaces to be initialized.**
  If you are using a file-backed database, the database service should depend on the local filesystem. If you are using ASM, the database service should depend on the service that manages ASM. The database service should depend on networking to allow for remote client connections.

- **The `method_environment` element in the `method_context` element defines the `ORACLE_HOME` and `ORACLE_SID` environment variables, which identify the database instance to start or stop.** These values are then available for the method script to use.
  If you create multiple instances of this service (see the `instance` element at the bottom of the manifest), then each instance might need its own `method_context` element to define the unique `ORACLE_HOME` and `ORACLE_SID` values for that particular database.

- **Attributes of the `method_context` element can define a resource pool in addition to the project and working directory shown in this example.** You can also define either a `method_profile` or `method_credential` element in the `method_context` element. The `method_credential` element can specify `supp_groups` and `limit_privileges` values in addition to the `user`, `group`, and `privileges` values shown in this example. See the DTD for more information.

- **The start/stop method script is `/lib/svc/method/oracle`.** The number of seconds before the method times out is increased from the default.

- **A user must be assigned the `solaris.smf.manage.oracle` authorization to enable or disable this service instance.** In this example, the user `oracle` is assigned the `solaris.smf.manage.oracle` authorization.

```xml
<?xml version="1.0"?>
<!--
Define a service to control the startup and shutdown of a database instance.
-->```
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">

<service_bundle type="manifest" name="oracle">
  <service name="site/application/database/oracle" type="service" version="1">
    <create_default_instance enabled="true"/>
  </service>

  <!-- Wait for all local file systems to be mounted.
  Wait for all network interfaces to be initialized. -->

  <dependency type="service"
    name="fs-local"
    grouping="require_all"
    restart_on="none">
    <service_fmri value="svc:/system/filesystem/local"/>
  </dependency>

  <dependency type="service"
    name="network"
    grouping="require_all"
    restart_on="none">
    <service_fmri value="svc:/milestone/network:default"/>
  </dependency>

  <!-- Define the methods. -->

  <method_context project=":default" working_directory=":default">
    <method_credential user="oracle" group="dba" privileges=":default"/>
    <method_environment>
      <envvar name="ORACLE_HOME" value="/opt/oracle/product/home"/>
      <envvar name="ORACLE_SID" value="oracle"/>
    </method_environment>
  </method_context>

  <exec_method type="method"
    name="start"
    exec="/lib/svc/method/oracle start"
    timeout_seconds="120"/>

  <exec_method type="method"
    name="stop"
    exec="/lib/svc/method/oracle stop"
    timeout_seconds="120"/>

  <!-- What authorization is needed to allow the framework
general/enabled property to be changed when performing the
action (enable, disable, etc) on the service. -->

  <property_group name="general" type="framework">
    <propval type="astring"
      name="action_authorization"
      value="solaris.smf.manage.oracle"/>
  </property_group>

  <!-- Define an instance of the database. -->
How to Create an SMF Service Using the Service Bundle Generator Tool

Add name and description metadata to the manifest so that users can get information about this service from the `svcs` and `svccfg describe` commands. See the template element in the DTD.

Use the `svccfg validate` command to make sure the service manifest is valid.

The following is the start/stop method script, `/lib/svc/method/oracle`, for the Oracle Database instance control service. This method calls the database `dbstart` and `dbshut` commands.

```bash
#!/bin/ksh -p

. /lib/svc/share/smf_include.sh

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
export PATH=$PATH:$ORACLE_HOME/bin

function startup
{
    dbstart $ORACLE_HOME
}

function shutdown
{
    dbshut $ORACLE_HOME
}

case $1 in
    start) startup ;;
    stop) shutdown ;;
    *) echo "Usage: $0 { start | stop }") >&2
        exit $SMF_EXIT_ERR_FATAL

    esac

exit $SMF_EXIT_OK
```

Database Listener Service

The listener is a process that manages the incoming traffic of client connection requests to the database instance. The listener service depends on the database service instance whose client connections it is managing.
This section shows the Oracle Database instance listener service manifest, `/lib/svc/manifest/site/listener.xml`. The following are some features to note about this service manifest:

- One service instance is defined, named `svc:/site/application/database/listener:default`. This instance is enabled by default.
- This service requires the Oracle Database instance control service, `svc:/site/application/database/oracle`, to be started. If the database instance is restarted for any reason, the listener will also be restarted.
- The `method_environment` element in the `method_context` element defines the `ORACLE_HOME` and `ORACLE_SID` environment variables, which identify the database instance to start or stop. These values are then available for the method script to use.
  
  If you create multiple instances of this service (see the `instance` element at the bottom of the manifest), then each instance might need its own `method_context` element to define the unique `ORACLE_HOME` and `ORACLE_SID` values for that particular database.
- Attributes of the `method_context` element can define a resource pool in addition to the project and working directory shown in this example. You can also define either a `method_profile` or `method_credential` element in the `method_context` element. The `method_credential` element can specify `supp_groups` and `limit_privileges` values in addition to the `user`, `group`, and `privileges` values shown in this example. See the DTD for more information.
- The start/stop method script is `/lib/svc/method/listener`. The number of seconds before the method times out is increased from the default.
- A user must be assigned the `solaris.smf.manage.oracle` authorization to enable or disable this service instance.
- The service is transient. See “Service Models” on page 20.

```xml
<?xml version="1.0"?>
<!--
Define a service to control the startup and shutdown of a database listener.
-->
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="manifest" name="listener">
    <service name="site/application/database/listener" type="service" version="1">
        <create_default_instance enabled="true" />
        <!-- Wait for the database to be started. -->
        <dependency type="service" name="oracle" grouping="require_all" restart_on="refresh">
            <service_fmri value="svc:/site/application/database/oracle" />
        </dependency>
    </service>
</service_bundle>
```
Add name and description metadata to the manifest so that users can get information about this service from the `svcs` and `svccfg describe` commands. See the `template` element in the DTD.

The following is the start/stop method script, `/lib/svc/method/listener`, for the Oracle Database instance listener service. This method starts or stop a listener process, `lsnrctl`. When `lsnrctl` starts, it tests the status of the database service.

```bash
#!/bin/ksh -p
```
. /lib/svc/share/smf_include.sh

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
export PATH=$PATH:$ORACLE_HOME/bin

function startup
{
  lsnrctl start

  # Wait for the listener to report ready.
  i=0
  while ! lsnrctl status | grep -i ready ; do
    ((i = i+1))
    if (( $i == 120 )) ; then
      # It’s been at least 2 minutes, time to give up.
      echo "The listener failed to report ready." >&2
      exit $SMF_EXIT_ERR_FATAL
    fi
    sleep 1
  done

  # Ping the database once to prove it is now available.
  if ! tnsping $ORACLE_SID ; then
    exit $SMF_EXIT_ERR_FATAL
  fi
}

function shutdown
{
  lsnrctl stop
}

case $1 in
  start)
    startup ;;
  stop)
    shutdown ;;
  *)
    echo "Usage: $0 { start | stop }" >&2
    exit $SMF_EXIT_ERR_FATAL
    ;;
esac

exit $SMF_EXIT_OK

Naming Services, Instances, Property Groups, and Properties

Service and instance names must fit the following expression:

([A-Za-z0-9][A-Za-z0-9.-]+)?[A-Za-z0-9][A-Za-z0-9.-]*
How to Convert a Run Control Script to an SMF Service

A service or instance name is case sensitive, must begin with an alphanumeric character, and can contain alphanumeric characters, the underscore (_), the hyphen (-), and the dot (.). To maintain backward compatibility, a single comma (,) is allowed between the first and last character.

Property group and property names must fit the following expression:

```regex
[A-Za-z0-9-_.~/?#\[\]%]+\n```

A property group or property name is case sensitive and can contain alphanumeric characters, the hyphen (-), dot (.), underscore (_), tilde (~), colon (:), forward slash (/), question mark (?), pound character (#), square brackets ([ and ]), at sign (@), exclamation point (!), dollar sign ($), ampersand (&), single quotation mark ('), parentheses (( and )), asterisk (*), plus sign (+), comma (,), semicolon (;), equal sign (=), space, and percent sign (%).

In an FMRI, property group and property names are encoded according to Uniform Resource Identifier (URI) Generic Syntax RFC 3986 except that the comma character is not encoded.

A property group type is a category for this property group. Property group types include application, dependency, method, framework, implementation, and template. Additional property group types can be introduced, provided they conform to the extended naming convention in smf(5). Do not specify framework, implementation, or template as the type of your property group. Property groups of type framework, implementation, or template have special use in SMF. Property groups of type application are expected to be only of interest to the service to which this group is attached.

**How to Convert a Run Control Script to an SMF Service**

This procedure shows how to replace a run control script with an SMF service manifest so that the run control service can be managed by SMF. To convert a run control script, use the `rc-script` name with the `-s` option of the `svcbundle` command. See the `svcbundle(1M)` man page for more information or enter `svcbundle help rc-script`.

1. **Determine the service model.**
   By default, `svcbundle` creates a transient service. Determine whether this run control script starts any long running daemon and is a contract service. See “Service Models” on page 20 and the `svc.startd(1M)` man page for information about service models.

2. **Create an initial manifest.**
   In this example, the service name is `ex_con` and is a contract service that runs at level 2.
Using a Stencil to Create a Configuration File

If your application cannot use `libscf` library interfaces to read properties, you can use a stencil to create a configuration file. A *stencil service* creates configuration files by using a stencil file and property values defined in the stencil service. A *stencil file* contains a structural definition of a configuration file that is required by a service even though that service is now managed by SMF. Stencil services enable you to take advantage of SMF configuration management with no change to the existing application.

The stencil is used to generate a configuration file immediately before the service instance start method is executed. If you update the stenciled property values, restart the service to incorporate the changes into the configuration file before the application starts and reads the configuration file.
How to Create a Stencil Service

A stencil file contains a structural definition of a configuration file that continues to be required by a service even though that service is now managed by SMF. The svcio utility generates the configuration file from the definitions in the stencil file and properties in the SMF service. See the svcio(1) man page for more information about the svcio utility and the smf_stencil(4) man page for information about stencil file format.

1. Create a stencil file.

The stencil file tells the svcio utility the format to use to create the configuration file. The svcio utility converts SMF properties into application-specific configuration files based on a template called a stencil.

2. Add a property group to the service.

The stencil service property group tells the svcio utility the path and ownership to use to create the configuration file. SMF regenerates configuration for all stencil-aware services before running the start or refresh methods. Property groups of type configfile tell SMF how to generate configuration files. Each configfile property group describes a single configuration file for the service and tells svcio how to generate these files from other properties stored in the SMF repository.

To configure a service to be stencil-aware, add a property group for each managed configuration file that contains the paths of both the stencil file to use as a template and the resulting configuration file. The property group has the following properties:

- **path**
  The path to which to write the configuration file, for example /etc/svc.conf.

- **stencil**
  The path of the stencil file to use, relative to /lib/svc/stencils. For example, if the value of the stencil property is svc.stencil, the /lib/svc/stencils/svc.stencil file will be used.

- **mode**
  The mode to use for the configuration file (path), for example 644.

- **owner**
  The owner to set for the configuration file (path). If this property is not set, the owner of the file is the user who invokes svcio.

- **group**
  The group to set for the configuration file (path). If this property is not set, the group will be the default group for path.
Stencil Service Examples in Oracle Solaris

Services for Puppet and Kerberos use stencils to provide configuration files.

Puppet Stencil Service

Puppet is a toolkit for managing the configuration of many systems. On Oracle Solaris, the Puppet application is managed by SMF.

High Level View of Puppet Services

When you install the system/management/puppet package, you get two SMF service instances: puppet:master and puppet:agent. These instances are disabled by default.

After you enable these instances, the following command shows that both puppet:master and puppet:agent are contract services:

```
$ svcs -p puppet

STATE STIME FMRI
online 17:19:32 svc:/application/puppet:agent
17:19:32 2565 puppet
online 17:19:32 svc:/application/puppet:master
17:19:32 2567 puppet
```

The following command shows a little more information about the processes started by the contract services:

```
$ ps -o pid,args -p 2565,2567

PID COMMAND
2565 /usr/ruby/1.9/bin/ruby /usr/sbin/puppet agent --logdest /var/log/puppet/puppet-
2567 /usr/ruby/1.9/bin/ruby /usr/sbin/puppet master --logdest /var/log/puppet/puppet
```

As suggested by the ps output, puppet is writing to log files in /var/log/puppet:

```
$ ls /var/log/puppet
puppet-agent.log  puppet-master.log
```

Initial Puppet Configuration File

Puppet expects to use a configuration file named /etc/puppet/puppet.conf. The /usr/sbin/puppet application reads configuration information from /etc/puppet/puppet.conf and not from properties set in the application/puppet service instances. To provide the required configuration file, each puppet instance provides a stencil file and configfile property group. The configfile property group tells the svcio utility to run and create the specified
configuration file. The stencil file is used to write data from service property values to the configuration file in the correct format.

The following command shows all puppet service properties that are in a property group of type configfile. This output shows that both instances of the puppet service have the same configfile properties with the same values. Each puppet service instance provides the path to the configuration file, the mode of the configuration file, and the path to the stencil file.

```
$ svcprop -g configfile puppet
svc:/application/puppet:master/:properties/puppet_stencil/mode astring 0444
svc:/application/puppet:master/:properties/puppet_stencil/path astring /etc/puppet/puppet.conf
svc:/application/puppet:master/:properties/puppet_stencil/stencil astring puppet.stencil
svc:/application/puppet:agent/:properties/puppet_stencil/mode astring 0444
svc:/application/puppet:agent/:properties/puppet_stencil/path astring /etc/puppet/puppet.conf
svc:/application/puppet:agent/:properties/puppet_stencil/stencil astring puppet.stencil
```

The following commands confirm that these instance properties are inherited from the parent service.

```
$ svccfg -s puppet listprop -l all puppet_stencil
puppet_stencil          configfile  manifest
puppet_stencil/mode    astring     manifest              0444
puppet_stencil/path    astring     manifest              /etc/puppet/puppet.conf
puppet_stencil/stencil astring     manifest              puppet.stencil
$ svccfg -s puppet:agent listprop -l all puppet_stencil
$ svccfg -s puppet:master listprop -l all puppet_stencil
```

For your infrastructure, you might need puppet:agent1 and puppet:agent2 instances, for example. In that case, you would customize property values and add properties for each instance as shown in “Modifying the Puppet Configuration File” on page 97.

The following is the initial content of the configuration file, /etc/puppet/puppet.conf:

```
# WARNING: THIS FILE GENERATED FROM SMF DATA.
# DO NOT EDIT THIS FILE. EDITS WILL BE LOST.
#
# See puppet.conf(5) and http://docs.puppetlabs.com/guides/configuring.html
# for details.

Puppet Stencil File

The content of the stencil file tells you what properties and other information are written to the configuration file. The puppet_stencil path that is the value of the puppet_stencil/stencil property is relative to /lib/svc/stencils. The following is the content of the stencil file, /lib/svc/stencils/puppet.stencil:

```
# WARNING: THIS FILE GENERATED FROM SMF DATA.
# DO NOT EDIT THIS FILE. EDITS WILL BE LOST.
#
# See puppet.conf(5) and http://docs.puppetlabs.com/guides/configuring.html
# for details.
; walk each instance and extract all properties from the config PG
```
How to Create a Stencil Service

Chapter 5 • Using SMF to Control Your Application

In the stencil file, svc:/$%s:(.*)/:properties (or %1) expands to svc:/application/puppet:agent/:properties and svc:/application/puppet:master/:properties, where .* (or %2) matches every instance. The instance name is then used to label the block in the configuration file. The next occurrence of .* (or %3) matches every property in the config property group for the %1 service instance. The stencil tells svcio to write the property name and the value of that property from the service instance to the configuration file.

Modifying the Puppet Configuration File

As you can see in “Initial Puppet Configuration File” on page 95, initially only the literal comment lines are written to the configuration file. Writing property values to the configuration file is prevented by the test of the value of the general/enabled property in the stencil file. The following command shows that by default, the value of the general/enabled property is false:

```
$ svcprop -p general/enabled puppet
svc:/application/puppet:master/:properties/general/enabled boolean false
svc:/application/puppet:agent/:properties/general/enabled boolean false
```

Using the svcadm enable command to enable an instance does not change the value of the general/enabled property. When you change the value of the general/enabled property to true and restart the instance, all the properties in the config property group for that instance are written to the configuration file.

```
$ svccfg -s puppet:agent setprop general/enabled=true
$ svcprop -p general/enabled puppet:agent
false
$ svcadm refresh puppet:agent
$ svcprop -p general/enabled puppet:agent
true
$ svcadm restart puppet:agent
```

The following command shows that initially the only property in the config property group is the path to the log file for each instance:

```
$ svcprop -p config puppet
svc:/application/puppet:master/:properties/config/logdest astring /var/log/puppet/puppet-master.log
svc:/application/puppet:agent/:properties/config/logdest astring /var/log/puppet/puppet-agent.log
```

The config property for the enabled instance has been added to the configuration file in a block labeled with the instance name:

```
# WARNING: THIS FILE GENERATED FROM SMF DATA.
```
The Puppet configuration documentation says that the configuration file can have [main], [agent], and [master] blocks. Configuration in the [main] block applies to both the agent and the master. For the Puppet agent, configuration in the [agent] block overrides the same configuration in the [main] block. For the Puppet master, configuration in the [master] block overrides the same configuration in the [main] block. If you want to provide a [main] block for configuration that is common to both the agent and master, create a puppet:main instance and appropriate config properties for that instance.

The following commands show how to add configuration to your Puppet configuration file.

```bash
$ svccfg -s puppet:agent
svc:/application/puppet:agent> setprop config/report=true
svc:/application/puppet:agent> setprop config/pluginsync=true
svc:/application/puppet:agent> refresh
svc:/application/puppet:agent> exit
$ svcadm restart puppet:agent
$ cat /etc/puppet/puppet.conf

# WARNING: THIS FILE GENERATED FROM SMF DATA.
#     DO NOT EDIT THIS FILE.  EDITS WILL BE LOST.
#
# See puppet.conf(5) and http://docs.puppetlabs.com/guides/configuring.html
# for details.

[agent]
logdest = /var/log/puppet/puppet-agent.log
pluginsync = true
report = true

Similar commands can be used to remove properties and change property values. See Chapter 4, “Configuring Services”. To add a main instance, use the svccfg add command as shown in “Adding Service Instances” on page 76.

**Kerberos Stencil Service**

Another example of an Oracle Solaris service that uses a stencil is Kerberos. The following command shows that the configfile property group is krb5_conf, the stencil file is /lib/svc/stencils/krb5.conf.stencil, and the configuration file is /etc/krb5/krb5.conf.

```bash
$ svccfg -g configfile svc:/system/kerberos/install:default
```
krb5.conf-disabled boolean true
krb5.conf-group astring sys
krb5.conf-mode integer 644
krb5.conf-owner astring root
krb5.conf/path astring /etc/krb5/krb5.conf
krb5.conf/stencil astring krb5.conf.stencil
SMF Best Practices and Troubleshooting

This appendix gives some recommended best practices and shows how to troubleshoot some SMF service problems.

**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.

Do not modify manifests and system profiles that are delivered by Oracle or 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, either create a site profile to customize the service, or use the svccfg command or the inetadm command to manipulate the properties directly. The /lib/svc/manifest/site and /var/svc/manifest/site directories are also reserved for site-specific use. Oracle Solaris does not deliver manifests into those directories.

To apply the same custom configuration to multiple systems, 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 file to /etc/svc/profile/site on each system, and restart the manifest-import service on each system. See “Configuring Multiple Systems” on page 77.

When you create a site profile, make sure the configuration defined does not conflict with configuration defined in another site profile for the same service or service instance. When SMF finds conflicting configuration in the same layer of the service configuration repository, the affected service instance is placed in the maintenance state.

Do not use non-standard locations for manifest and profile files. See “Service Bundles” on page 24 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 110.

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 either of the following descriptions:

- The service is enabled but is not running.
- The service is preventing another enabled service from running.

The following list summarizes how to approach service problems:

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

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*.

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(1M)
    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 40 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 13:38:42 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.

```bash
$ 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 return the service to the online state. For services in the maintenance state, the `clear` subcommand tells the restarter for that service that the service is repaired.

```
$ 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(5) 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(1M) will provide a sulogin(1M) 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. Do not directly invoke the svc.configd daemon. The svc.configd daemon will start 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 108.

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

1. **Log in.**

   Using the root password, log in either remotely or at the sulogin prompt.

2. **Run the repository restore command:**

   ```bash
   # /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 26.

   SMF maintains persistent and non-persistent configuration data. See “Service Configuration Repository” on page 23 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(5) 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:

<table>
<thead>
<tr>
<th>CHOICE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot</td>
<td>restore the most recent post-boot backup</td>
</tr>
<tr>
<td>manifest_import</td>
<td>restore the most recent manifest_import backup</td>
</tr>
<tr>
<td>-seed-</td>
<td>restore the initial starting repository (All customizations will be lost, including those made by the install/upgrade process.)</td>
</tr>
<tr>
<td>-quit-</td>
<td>cancel script and quit</td>
</tr>
</tbody>
</table>

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.

**Note** - Selecting -seed- restores the seed repository. This repository is designed for use during initial installation and upgrades. Using the seed repository for recovery purposes should be a last resort.

After the backup to restore has been selected, it is validated and its integrity is checked. If there are any problems, the restore_repository command prints error messages and prompts you for another selection. Once a valid backup is selected, the following information is printed, and you are prompted for final confirmation.

**After confirmation, the following steps will be taken:**

```
svc.startd(1M) and svc.configd(1M) 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(1M).
```

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(1M)` 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 by Editing the GRUB Menu at Boot Time” in “Booting and Shutting Down Oracle Solaris 11.2 Systems” and “Messages options” in the `kernel(1M)` 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 A-1** SMF Startup Message Logging Levels

<table>
<thead>
<tr>
<th>Logging Level Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quiet</td>
<td>Display on the console any error messages that require administrative intervention. Also record these messages in <code>syslog</code> and in <code>/var/svc/log/svc.startd.log</code>.</td>
</tr>
<tr>
<td>verbose</td>
<td>In addition to the messaging provided at the quiet level, display on the console a single message for each service started, and record in <code>/var/svc/log/svc.startd.log</code> information about errors that do not require administrative intervention.</td>
</tr>
<tr>
<td>debug</td>
<td>In addition to the messaging provided at the quiet level, display on the console a single message for each service started, and record any <code>svc.startd</code> debug messages in <code>/var/svc/log/svc.startd.log</code>.</td>
</tr>
</tbody>
</table>
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 A-2.

- When booting a SPARC system, specify the -m option to the boot command at the ok prompt. See the -m option in the kernel(1M) 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 by Editing the GRUB Menu at Boot Time” in “Booting and Shutting Down Oracle Solaris 11.2 Systems” and the -m option in the kernel(1M) 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 11.2 Systems”, the inittab(4) man page, and the /etc/init.d/README file. For more information about SMF boot milestones, see the milestone subcommand in the svcadm(1M) man page.
### TABLE A-2  
SMF Boot Milestones and Corresponding Run Levels

<table>
<thead>
<tr>
<th>SMF Milestone FMRI or Keyword</th>
<th>Corresponding Run Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
<td>The <code>none</code> keyword represents a milestone where no services are running except for the master restarter. When <code>none</code> is specified, all services except for <code>svc:/system/svc/restarter:default</code> are temporarily disabled. The <code>none</code> milestone can be useful when for debugging startup problems. See “How to Investigate Problems Starting Services at System Boot” on page 113 for specific instructions.</td>
</tr>
<tr>
<td>all</td>
<td></td>
<td>The <code>all</code> keyword represents a milestone that depends on every service. When <code>all</code> is specified, temporary enable and disable requests are ignored for all services. This is the default milestone used by <code>svc.startd</code>.</td>
</tr>
<tr>
<td>svc:/milestone/single-user</td>
<td>s or S</td>
<td>Ignore temporary enable and disable requests for <code>svc:/milestone/single-user:default</code> and all services on which it depends either directly or indirectly. Temporarily disable all other services.</td>
</tr>
<tr>
<td>svc:/milestone/multi-user</td>
<td>2</td>
<td>Ignore temporary enable and disable requests for <code>svc:/milestone/multi-user:default</code> and all services on which it depends either directly or indirectly. Temporarily disable all other services.</td>
</tr>
<tr>
<td>svc:/milestone/multi-user-server</td>
<td>3</td>
<td>Ignore temporary enable and disable requests for <code>svc:/milestone/multi-user-server:default</code> and all services on which it depends either directly or indirectly. Temporarily disable all other services.</td>
</tr>
</tbody>
</table>

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 111 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.**
   ```bash
   $ 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.**
   ```bash
   $ 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:
   ```bash
   $ 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(1M)` 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 79.
Index

A
actions, 18
admin layer, 25, 47
ASM, 85
authorizations, 29
Automatic Storage Management (ASM), 85

B
backups, 26
boot backups, 26
boot command
  milestone option, 111
  none milestone, 113
booting
  current milestone, 112
  logging levels, 110
  milestones, 111
  to an SMF milestone, 113
  without starting services, 113
bundle support, 74
bundles, 24

c
child service, 20
composed view, 23, 42
config property group, 23
configuration
  adding property groups and properties, 70
  adding property values, 66
  deleting bundle-supported configuration, 74
  deleting customizations, 72, 74
  deleting properties, 72
  deleting property groups, 72
  deleting property values, 72
  masking, 74
  modifying, 63
  refreshing, 57
  setting property values, 66
  unmasking, 74
configuration files, 15, 16, 27, 94
configuration repository See service configuration repository
contract service, 20
customizations
  listing, 47

d
daemon service, 20
default restarter, 22
degraded service state, 21, 103, 106
degraded state, 47
dependencies, 21, 21
  effect on the state of a dependent service, 38
  groupings, 35
  instances that a service depends on, 36
  instances that depend on a service, 37
  listing, 34
describe subcommand
  svccfg command, 42
disable a service, 54
disabled service state, 21
DTD, 83

e
enable a service, 51
environment variables
modifying in methods, 69
error logging, 40
/etc/inetd.conf file, 27, 81, 114
/etc/init.d scripts, 27
/etc/inittab entries, 27
/etc/rc?.d scripts, 27
event notification, 59

F
Fault Management Resource Identifier See FMRI
FMRI, 20, 92

G
general property group, 23

I
inetadm command, 114
 example, 79
inetconv command, 114
inetd command, 114
inetd service
 convert to SMF service, 114
 modifying command-line arguments, 79
 modifying properties, 79
inetd.conf file, 27, 114
init states See run levels
init.d scripts, 27
initial snapshot, 26
inittab entries, 27
instances, 18
 adding, 76
 naming, 20, 91

K
Kerberos
 stencil service example, 98

L
layers, 25

M
maintenance service state, 21, 103, 106
manifest layer, 25, 47
manifest-import backups, 26
manifests, 18
 format, 83
 site directory, 83
 standard location, 24, 83
masked configuration, 74
masked entities, 47, 58
master restarter daemon, 22
methods, 18
milestone services, 18
milestones
 booting, 111
 corresponding run levels, 111
 currently booted, 112
 none, 113

N
non-persistent properties, 46
non-persistent property groups, 45
notification configuration, 59
notification parameters
 displaying, 48
O
offline service state, 21, 103
online service state, 21
Oracle Database, 85
   ASM, 85
database service, 86
   listener service, 88
P
permissions, 29
previous snapshot, 26
privileges, 29
profiles, 18
   creating, 77
   site directory, 24, 77
   standard location, 24
properties, 23, 23, 41
   composed view of, 23, 42
   in the running snapshot, 42
   naming, 91
   non-persistent, 45
property groups, 23, 41
   naming, 91
   non-persistent, 45
type, 45, 92
Puppet
   stencil service example, 95
R
rc?.d scripts, 27
refresh service configuration, 57
remove a service, 58
repository See service configuration repository
resolv.conf file, 27
restart a service, 56
restart the manifest-import service, 56
restarter property group, 23
restarters, 18, 22
   default restarter, 22
   delegated restarters, 23
   svc.startd master restarter daemon, 22
restore_repository command, 108
rights profiles, 29
   roles, 29
run control scripts, 27
   converting to SMF service, 92
run levels
   corresponding SMF milestones, 111
current, 112
default level, 111
   running snapshot, 26, 42
S
selectsnap subcommand
   svccfg command, 44
service bundles
   DTD, 83
   standard locations, 24
Service Configuration Facility library, libscf, 23
service configuration repository, 18, 23
   backups, 26
   layers, 25
   library interfaces, 23
   modifying, 63
   repairing, 107
   snapshots, 26
   svc.configd daemon, 107
   template data, 23
service contract, 20
service instances, 18
   adding, 76
service models, 20
service properties, 23, 23, 41
service property groups, 23, 41
service restarter, 22
service states
   changing manually, 103, 106
   description, 21
   listing, 31
   transitions, 21
services, 18
   deleting, 58
   disabling, 54
   enabling, 51
   modifying configuration, 63
   naming, 20, 91
   refreshing configuration, 57
restarting, 56
starting, 51
stopping, 54
single-user login, 107
site-profile layer, 25, 47
SMF database See service configuration repository snapshots, 44, 47
initial, 26
previous, 26
reverting, 76
running, 26
running snapshot, 42
start, 26
upating running configuration, 57
start a service, 51
start property group, 23
start snapshot, 26
STATE column, 31
state transition notification, 59
stencil files, 16, 93, 94
stencil service, 93
Kerberos example, 98
Puppet example, 95
stop a service, 54
sulogin command, 107
svc.configd repository daemon, 107
svc.startd master restarter daemon, 22, 107
svcadm command, 51
clear subcommand, 103
disable subcommand, 54
enable subcommand, 51
mark subcommand, 106
milestone subcommand, 111, 113
refresh subcommand, 57
restart manifest-import subcommand, 56, 58
restart subcommand, 56, 103
synchronous options, 16
svcbundle command
creating manifests, 83
creating profiles, 78
installing automatically, 78, 85
rc-script service, 92
svccfg command
add subcommand, 76
addpg subcommand, 70
addpropvalue subcommand, 66, 70
apply subcommand, 77
command input file, 66
delcust -M subcommand, 74
delcust subcommand, 72, 74
delete subcommand, 58
delpg subcommand, 72
delprop subcommand, 72
delpropvalue subcommand, 72
describe subcommand, 42
dirop prop subcommand, 64
extract subcommand, 79
import subcommand, 77
interactive use, 65
listcust -M subcommand, 58, 74
listcust subcommand, 44, 47
listpg subcommand, 45
listprop subcommand, 44
listsnap subcommand, 47
refresh subcommand, 57
revert subcommand, 76
selectsnap subcommand, 44, 76
setenv subcommand, 66, 69
setprop subcommand, 66, 70, 72
showing properties, 41
unsetenv subcommand, 69
validate subcommand, 84
svcio utility, 16, 94
svcprop command
showing properties, 41, 42
svcs command, 31
system-profile layer, 25

T
template data, 23
transient service, 20

U
uninitialized service state, 21
update service configuration, 57
W
wait service, 20