



An Oracle White Paper
May 2010

Management of Systems and Services Made Simple with the Oracle[®] Solaris Service Management Facility

Best Practices for Oracle Solaris 10 in the Datacenter

Introduction.....	1
Managing Services in Older Oracle Solaris Operating Systems.....	1
Introducing the Oracle Solaris Service Management Facility.....	4
The SMF Framework.....	4
SMF Configuration Repository.....	4
SMF Restarter.....	5
SMF Service Instances.....	5
SMF Service Categories.....	5
Components of SMF Services.....	6
SMF Service Manifest.....	6
SMF Service Methods.....	7
SMF Service Executables or Daemons.....	7
SMF Service Log Files.....	7
Service Identifiers.....	7
SMF Services versus SMF Service Instances.....	8
SMF Service Profiles.....	9
Starting SMF at Boot Time.....	9
Inetd as a Restarter.....	10
Using and Administering SMF.....	11
What Happens When Starting and Stopping Services.....	11
Starting SMF Services.....	11
Stopping SMF Services.....	11
Restarting SMF Services.....	11
Monitoring SMF Services.....	13
Service States.....	13
Viewing Service Dependencies	14
Diagnosing Problems with Services.....	14

Securing Access to Services.....	15
Using Role-Based Access Controls.....	15
Modifying Existing Services.....	16
The inetconv Utility.....	16
Comparing System Service Administration and Oracle Solaris SMF Commands	16
Putting the Service Management Facility to Work.....	17
Creating a New SMF Service.....	17
Converting Existing Services to SMF.....	20
Convert a SAMP Stack from rc Scripts to SMF.....	21
Enabling inetd Services.....	25
Refreshing Service Configurations.....	26
Best Practices for SMF.....	26
Conclusion.....	26
About the Author.....	27
References.....	27

Introduction

Today's advanced systems and complex environments make it difficult for administrators to provide reliable services. As IT requirements evolve and businesses increasingly depend on IT systems, delivering rigorous service levels is paramount. Indeed, IT staff and management face serious consequences when systems fall short. The key to success is ensuring resiliency against hardware faults, understanding what needs to be running on systems for applications to work, and making sure the system foundation is available to applications.

Older UNIX® operating systems, such as Oracle Solaris 8 and Oracle Solaris 9, include a set of services — a set of individual applications that execute as a single process — that start at boot time and continue to execute while the system is up and running. These processes handle network, file system, security, device, print, cron, and other system services, fielding requests received. System services start sequentially, with some dependent on the availability of other running services in order to start. If a service requires another service that fails to start, the dependent service cannot start and the environment might fail to run completely.

Extensive computing demands have resulted in an ever-growing collection of services that has exceeded the utility of this original model. Older operating systems cannot handle the intricate interdependencies of system services. A solution is needed to help simplify system and service management and deliver required service levels.

Managing Services in Older Oracle Solaris Operating Systems

The methods used to manage system services in Oracle Solaris 8 and Oracle Solaris 9 can add complexity to system administration tasks. Typically, these environments start system services by running the `init` daemon. The `init` daemon executes `/etc/inittab` run control (`rc`) scripts that run sequentially to start all system services. The `inetd` daemon spawns other daemons to provide network services and the system reaches the default run level 3 — multiuser with

networking. However, this approach can create difficulties in managing and troubleshooting system services.

Modifying and Managing

In Oracle Solaris 8 and Oracle Solaris 9, changing daemon parameters requires knowing from which directory the daemon was started and how to change the start method parameters. Once the rc script used to start the daemon is identified, it is vital to make changes carefully as errors in startup scripts can prevent the system from booting properly or even cause the system to hang completely.

Methods for enabling or disabling system services in older Oracle Solaris operating systems can be haphazard, without a uniform and specific procedure for doing so across the operating system. To disable a service, administrators often rename a particular rc script to prevent it from running upon system reboot. This approach can fail as patch installations or upgrades may reinstall the original files — or updated versions — causing the disabled service to start the next time the system is rebooted. In addition, older Oracle Solaris operating systems do not contain an audit trail for modifications to services, making it difficult to track changes made to the system and by whom they are made.

Troubleshooting

Debugging rc scripts requires system reboots in order to test modifications to a system service. Administrators often find the debugging process challenging due to cryptic console messages during startups that can make it hard to pinpoint causes of service problems. Troubleshooting particularly difficult problems can result in the need for multiple system reboots as fixes are attempted, potentially impacting system availability. Furthermore, starting rc scripts in sequential order can cause the system boot process to run inefficiently. Indeed, if not planned carefully, dependent services can loop endlessly while waiting for a required process to start.

Without a uniform and consistent mechanism for defining and maintaining service definitions and configurations, system reboots also can potentially eradicate system services. In addition, operating system patches and upgrades can interrupt or halt services. Clearly these older operating system versions do not contain a way to consistently, effectively, and persistently manage system and application services, costing administrators significant amounts of time and effort.

Introducing the Oracle Solaris Service Management Facility

In order to address issues of system services management and control in older Oracle Solaris operating systems, Oracle Solaris 10 implements system services as persistent programs that handle system or user requests. These services include network services such as ftp, telnet, and rlogin, as well as file system, security, device, print, and cron services. The Service Management Facility (SMF) in Oracle Solaris provides a framework that simplifies the management of system services and delivers new and improved ways to manage them. SMF implements services as objects that can be viewed and managed, and makes it possible to view relationships between services and processes. Systems can boot faster and recover from errors such as hardware failures that cause services to fail. SMF supports the exact knowledge of the state of the system, what kernel and application services are running, their dependencies, and what is needed to restart them.

SMF removes inconsistencies found in the interface of rc scripts in older Oracle Solaris versions. SMF was introduced in Oracle Solaris 10, and prior to the implementation of SMF in Oracle Solaris, administrators needed to know that the tomcat script takes one set of keywords, mysql takes different keywords, and mysqsafe takes yet another set. SMF replaces these varying sets of keywords with a uniform, consistent command line and application programming interface. It also defines a uniform model for start scripts and provides a template that makes it easy to write new scripts.

The SMF framework is designed to support ISV applications such as Oracle database software, and Web services. In addition, SMF services can include physical network devices, configured IP addresses, kernel configuration information, and milestones that correspond to system initialization states such as multiuser run levels. The result is a scalable, supported, unified model for services and service management on Oracle Solaris platforms.

The SMF Framework

The SMF Framework consists of a service repository that contains the definitive configuration for each service, and a master restarter that manages service instances within SMF.

SMF Configuration Repository

The SMF configuration repository stores persistent configuration information as well as SMF runtime data for services. The repository is distributed among local memory and local files. Administrative simplicity is shared across many Oracle Solaris 10 instances, with configuration information shared and SMF instances utilized only for data manipulation or queries. Forming the core of SMF, the repository provides a consistent view of service states, a unified interface for retaining, viewing, and configuring service properties, and a persistent way to enable and disable services. In addition, the repository stores a snapshot of each service configuration at the time that the service starts successfully, making it possible to restore the service easily to a known good configuration if necessary.

SMF Restarter

In Oracle Solaris 10, a system boot starts the `init` process which starts the SMF master restarter daemon `svc.startd`. The `svc.startd` daemon queries the SMF repository to locate other system services and starts them according to their dependencies — in parallel whenever possible. As the SMF master restarter daemon initializes services, it enforces defined dependency relationships between services, starting only those services where dependency requirements have been met. The restarter also keeps track of service failures and dependency events in order to automatically restart a service when it detects a service has failed or one of its required dependencies is no longer available. In this way, the restarter helps to simplify and automate some service-related tasks.

SMF Service Instances

Systems frequently run multiple copies of the same service, usually with slightly different configurations. In SMF, these varied configurations are known as *service instances*. To facilitate configuration sharing, SMF extracts configuration properties for each service from the service instances, inheriting the instance's properties from the parent service unless the instance specifically overrides them. By offering each mode as a separate instance of the same service, SMF can use a single, common configuration for all three modes while allowing the administrator to enable or disable each mode independently. For example, the network login service can have the following service instances: `rlogin`, `rlogin` with Kerberos, and `rlogin` with Kerberos and encryption.

SMF Service Categories

System services in SMF fall into a number of service categories. These categories are not used by the system, but can be used to name the services and identify the general use of a service. The categories include:

- **Application** — Application services consist of higher-level applications such as Oracle Web Server.
- **Milestone** — Milestones are a special type of SMF service that do not run applications. Instead, milestone services indicate a level of system readiness through a core set of services brought online to reach that level of system functionality. Each milestone is dependent on its list of services. Milestone services supplant the traditional notion of run levels used in older Oracle Solaris operating systems. To see what services must be running before a milestone is reached, issue the command `svcs -d [milestone FMRI]`. Here is an example that shows that the `name-services` milestone depends on the default `nisplus`, `dns client`, `ldap client`, and `nis client` service instances. In this case, they are not online, so the `name-services` milestone cannot be reached:

- **Platform** — These are services that are specific to various hardware platforms. For example, users of Oracle Sun SPARC Enterprise M-Series servers can find Dynamic Reconfiguration service daemons running on the system.
- ```
svcs -d name-services
disabled Jan_04 svc:/network/rpc/nisplus:default
disabled Jan_04 svc:/network/dns/client:default
disabled Jan_04 svc:/network/ldap/client:default
online Jan_04 svc:/network/nis/client:default
```

- System — System services such as `coreadm` are Oracle Solaris system services that allow an administrator to control core file generation.
- Device — Device services pertain to system devices, such as disks and host bus adapter cards.
- Network — Network services are network or internet services such as protocols.
- Site — These services contain site-specific descriptions.

## Components of SMF Services

SMF services are comprised of one or more components that are utilized based on the functionality and category of the individual services. Figure 1 illustrates the various SMF components for the `cron` service.

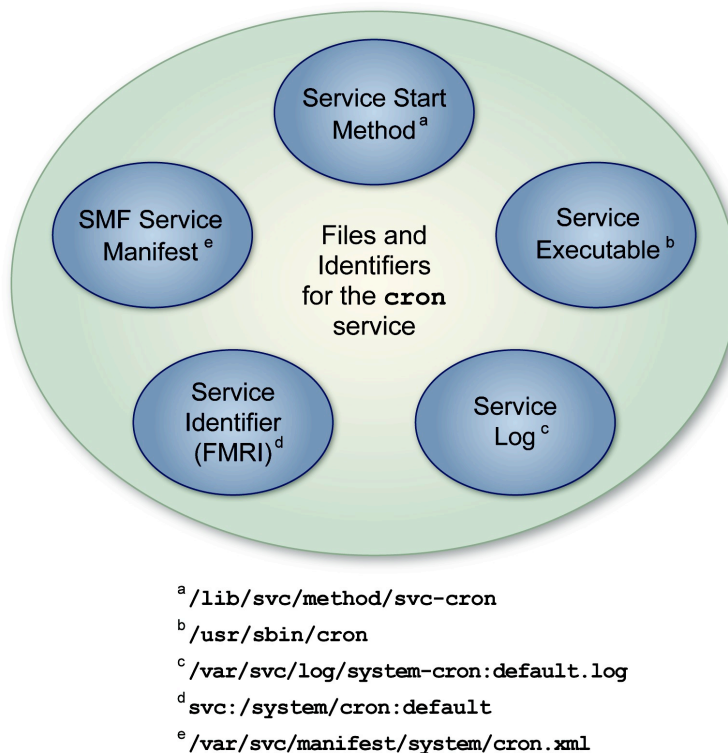


Figure 1. SMF Components for the `cron` Service

### SMF Service Manifest

SMF service manifests are the delivery mechanism for all services. Created as XML files that are imported into the SMF repository, SMF manifests contain a complete set of properties that are associated with a service or a service instance. These files are stored in the `/var/svc/manifest` file. To incorporate information from the manifest into the repository, the administrator must either run `svccfg import` or allow the service to import the information during a system boot. (See the `service_bundle(4)`, `svccfg(1M)`, or `smf_bootstrap(5)` man pages.)

Note – Directly modifying SMF manifest files provided in Oracle's Solaris 10 OS to change the properties of system services is not supported as customizations may not be preserved across software upgrades. The recommended way to change service configuration properties stored in the repository is with the `svccfg` command.

## SMF Service Methods

Service methods are invoked by the service restarter to move the service from one state to another whenever an administrative action is performed on the service. These methods are defined in the SMF configuration repository and can be executables, shell scripts, or keywords such as `:kill`. The service repository can contain a start method to initialize a service, a stop method to halt it, and a refresh method to reread its configuration. Methods for services managed by the master restarter `svc.startd` are often shell scripts similar to traditional `rc` scripts in previous Oracle Solaris versions. (For more information, see `smf_method(5)` or `svc.startd(1M)` man pages.)

## SMF Service Executables or Daemons

In some cases, a start method invokes a service executable that provides the capabilities of the service. The `/usr/sbin/cron` daemon is the executable for the `cron` service. A service executable can be invoked directly as a method.

## SMF Service Log Files

A service's restarter can specify a log file to capture information about the service. The `svc.startd` master restarter daemon maintains persistent log files for each service in the service's log file in the `/var/svc/log` directory. For example, the log file for the `cron` service is `/var/svc/log/system-cron:default.log`.

## Service Identifiers

Services managed by SMF use Fault Management Resource Identifiers (FMRI) to identify system objects for which advanced fault and resource management capabilities are provided. Services managed by SMF are assigned FMRI strings that typically contain three parts separated by colons. The general form of an FMRI string is *service indicator:service name:service instance*. The first component indicates that the FMRI describes an SMF service. The second component specifies the service name. The third component identifies the service instance.

The following three formats refer to the Oracle Solaris `syslogd(1M)` service:

- `svc://localhost/system/system-log:default`

This FMRI format is a full string and contains an absolute path, including a location path such as `localhost`. Full FMRI strings describe a service uniquely and should be used in shell scripts.

- `svc:/system/system-log:default`

This string is a path relative to the local machine.

- `system/system-log:default`

This format is simply the service identifier with the string prefixes implied. This abbreviated form can be used as long as there is only one instance of the service and SMF can infer the correct instance to be used.

Note that the SMF tools in the current release of Oracle Solaris 10 can only manage services on the local host. However, other management tools that operate on multiple types of resources or across machine boundaries can utilize one of the first two forms to describe services.

FMRI strings for multiple instances of a single service are almost identical except for the service instance. The example below shows an FMRI with various instances of the network login service:

- `svc:/network/login:rlogin` (rlogin)
- `svc:/network/login:klogin` (rlogin with Kerberos)
- `svc:/network/login:eklogin` (rlogin with Kerberos and encryption)

#### Legacy `init.d` scripts

Note that FMRI strings for services created through legacy `init.d` scripts start with `lrc` instead of `svc`, as seen in this example: `lrc:/etc/rcS_d/S35cacheos_sh`.

#### Converted `inetd.conf` Services

Services listed in the `/etc/inetd.conf` file in previous Oracle Solaris operating system versions are automatically converted into SMF services when a system is booted for the first time with SMF in Oracle Solaris 10. These FMRI strings use the syntax below where `<service-name>` is the name defined in `/etc/inetd.conf` and `<protocol>` is the protocol for the service:

```
network/<service-name>/<protocol>
```

#### SMF Services versus SMF Service Instances

A service consists of the general service definition and one or more varying configurations — or instances — that implement the service. An instance's properties are inherited from the service, unless the instance specifically overrides them. For example, a Web server is a service. An instance of the Web server is created by configuring a specific Web server daemon to listen on a specific port number.

If multiple binaries of a service running simultaneously on the system would cause an error, it must be defined as a *single\_instance* service. This tag tells the restarter not to start multiple service instances simultaneously, regardless of administrative configuration. The `cron` service is a good example of a service that must run as a single instance.

## SMF Service Profiles

SMF profiles are XML files that list sets of service instances and whether each should be enabled or disabled. Profiles can be used to enable or disable many services at once. Not all services need to be listed in a profile. Each profile only needs to include those services that need to be enabled or disabled to make the profile useful.

Administrators can use the `svccfg(1M)` command to export service profiles so they can be saved. Alternatively, a previously saved list can be loaded so the system can adopt those characteristics. When working with Oracle Solaris JumpStart for large numbers of identical systems, or to archive the system configuration for later restoration, extracting and saving profile data can be used to create a unique version of an SMF profile. Using this tool, administrators can bring systems to a known state, which is helpful when attempting to secure a system or facilitating the diagnosis of another system.

## Starting SMF at Boot Time

Figure 2 illustrates the SMF initialization process when a system boots. The `init` process is started by the Oracle Solaris 10 kernel, and it reads the `/etc/inittab` file. The `/etc/inittab` file contains an entry for the SMF master restarter daemon `svc.startd`. This master restarter daemon starts all SMF services that are started by the `svc.startd` daemon.

The `svc.startd` daemon calls `svc.configd`, the SMF configuration daemon that manages all access to the SMF repository. The `svc.configd` daemon starts each service according to the designated service restarter and the appropriate methods for starting the service.

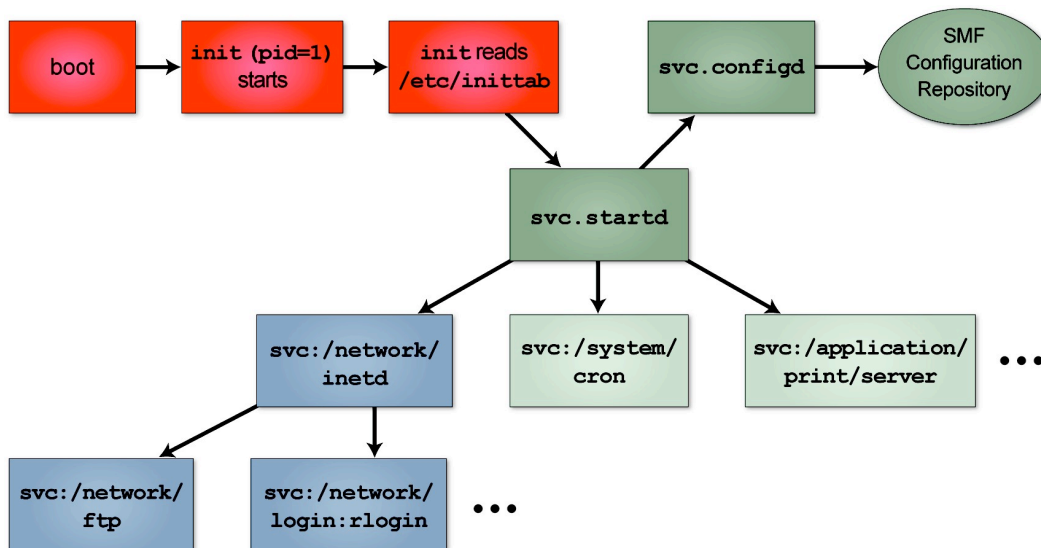


Figure 2. The SMF initialization process.

Note that the `svc.startd` daemon calls the `inetd` service, which then starts many of the network services that were started by the `inetd` daemon in Oracle Solaris 8 and 9.

## Inetd as a Restarter

The `inetd` service is the delegated restarter for many network services, managing the starting and stopping of services in lieu of the default restarter `svc.startd`. For example, in Figure 2, the `inetd` service is shown as the restarter for the `rlogin` and `ftp` network services.

When the master restarter `svc.startd` starts the service `inetd` at boot time, the `inetd` service listens for requests for network services such as `ftp`. When an incoming `ftp` request occurs, the `inetd` service determines that the request is for the `ftp` service (`network/ftp`), and invokes the appropriate start method (`/usr/sbin/in.ftpd -a`). Figure 3 illustrates the interaction of `inetd` and the `ftp` service.

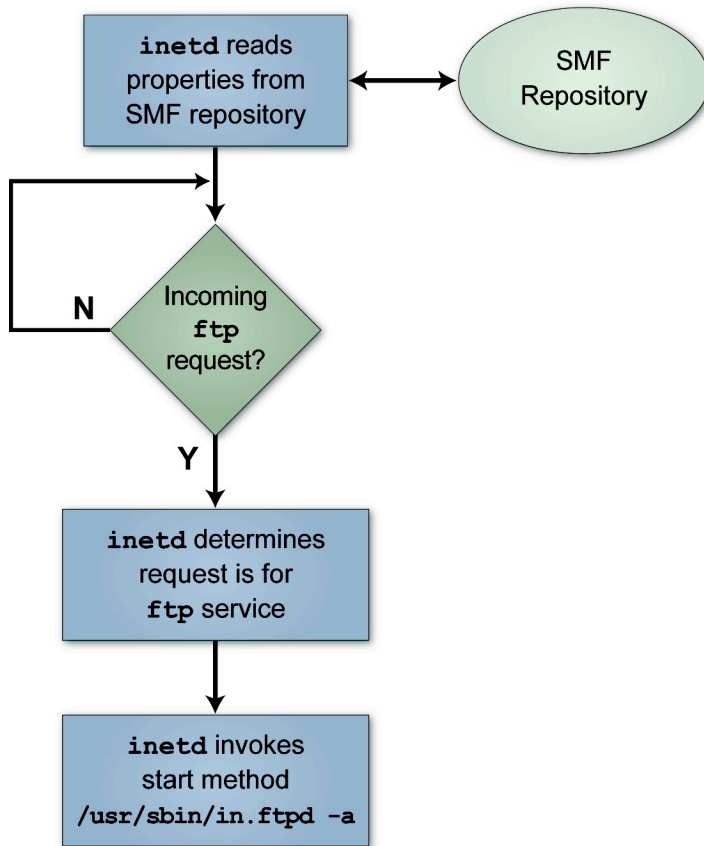


Figure 3. `ftp` is managed within SMF by the `inetd` service.

## Using and Administering SMF

SMF offers a consistent, reliable, and secure administrative model for managing system services. The result is a defined set of actions that can be invoked on a service by an administrator, including enable, disable, refresh, and restart.

## What Happens When Starting and Stopping Services

The `svcadm` command is used to request actions on services that are managed by SMF to start, stop, and restart services. When the `svcadm` command requests an action on an SMF service, SMF records the service status change in the service configuration repository. These state changes persist across reboots, patches, and system upgrades.

### Starting SMF Services

The `svcadm enable` command is used to start SMF services. For example, the following command starts (enables) the secure shell SMF service:

```
svcadm enable network/ssh
```

### Stopping SMF Services

The `svcadm disable` command is used to stop (disable) SMF services. For example, the following command stops the secure shell SMF service:

```
svcadm disable network/ssh
```

This command prevents the service from restarting. If the service needs to be stopped just until the next reboot, use the `-t` qualifier — which does not record the change in the service repository — to make the request temporary:

```
svcadm disable -t network/ssh
```

### Restarting SMF Services

If a service is running but needs to be restarted due to a configuration change or some other reason, the service can be restarted without using separate stop and start commands. Indeed, the only reason to issue separate stop and start commands is when it is necessary to make changes after the service is disabled and before re-enabling the service. The following command restarts the secure shell service:

```
svcadm restart network/ssh
```

Table 1 contains a listing of frequently used commands for administering SMF services.

---

**TABLE 1. FREQUENTLY USED COMMANDS FOR ADMINISTERING SMF SERVICES**

---

| COMMAND | EXAMPLES | DESCRIPTION |
|---------|----------|-------------|
|---------|----------|-------------|

---

|                                                     |                                                                                                                                     |                                                                                                                                                                               |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>svcs</b>                                         | <b>Displays information about service instances</b>                                                                                 |                                                                                                                                                                               |
| # svcs -a                                           |                                                                                                                                     | Lists all services, enabled or disabled.                                                                                                                                      |
| # svcs -p [FMRI]                                    |                                                                                                                                     | Lists any processes associated with each service instance.                                                                                                                    |
| # svcs -l                                           |                                                                                                                                     | Displays details about a particular service.                                                                                                                                  |
| # svcs -x                                           |                                                                                                                                     | Displays all broken services and gives a reason as to why SMF believes the service is broken. This command is a powerful troubleshooting tool.                                |
| # svcs -xv                                          |                                                                                                                                     | Displays all broken services, gives a reason as to why SMF believes the service is broken, and provides verbose text as part of the explanation (-v).                         |
| <b>svcadm</b>                                       | <b>Issues requests for actions on executing services, including enabling, disabling, and restarting service instances</b>           |                                                                                                                                                                               |
| # svcadm enable foo                                 |                                                                                                                                     | Enables the system service foo.                                                                                                                                               |
| # svcadm refresh print/server                       |                                                                                                                                     | Enables and refreshes a service instance, in this case the print server service. (Note that it is necessary to refresh a service after any configuration changes via svccfg.) |
| # svcadm disable -t cron                            |                                                                                                                                     | Disables the cron service temporarily (until the next reboot). To disable a service persistently across reboots, do not use the -t option.                                    |
| # svcadm restart foo                                |                                                                                                                                     | Restarts the system service foo.                                                                                                                                              |
| <b>svccfg</b>                                       | <b>Displays and manipulates the contents of the SMF repository</b>                                                                  |                                                                                                                                                                               |
| # svccfg                                            |                                                                                                                                     | When executed with no options, the svccfg command enters the svccfg interactive shell. Within the interactive shell, the help subcommand lists other valid subcommands.       |
| # svccfg import<br>/var/svc/manifest/system/foo.xml |                                                                                                                                     | Imports a service manifest into SMF.                                                                                                                                          |
| # svccfg delete FMRI                                |                                                                                                                                     | Deletes (removes) the service definition from the SMF repository.                                                                                                             |
| <b>svccprop</b>                                     | <b>Retrieves property values from the SMF repository, and outputs information in a format appropriate for use in shell scripts.</b> |                                                                                                                                                                               |
| # svccprop -p<br>propertygroup/property FMRI        |                                                                                                                                     | Retrieves property values for the specified service instance.                                                                                                                 |
| # svccprop -p start/exec<br>system/cron             |                                                                                                                                     | Retrieves property values for the specified service instance. The cron service is used as an example.                                                                         |

---

|                |                                                                                            |
|----------------|--------------------------------------------------------------------------------------------|
| <b>inetadm</b> | <b>Provides the ability to observe and configure network services controlled by inetd.</b> |
| # inetadm -p   | Display global defaults for all inetd services.                                            |

---

## Monitoring SMF Services

Older versions of Oracle Solaris lacked a consolidated view of services, making it difficult to monitor system services easily. SMF provides a unified structure with commands that make it easy to perform monitoring tasks.

The `svcs` command is the principal tool for determining service states and properties. When used without qualifiers, the command lists all enabled system services along with their state, the time the state was entered, and the FMRI. Those services that are not enabled or that failed to start due to an error, unresolved dependency, or for some other reason, are not listed.

In addition to SMF services, the `svcs` command lists legacy services started through `init.d` scripts. These legacy services have FMRI that start with `lrc`, and can be monitored — but not administered — with SMF. An example of a legacy service FMRI is `lrc:/etc/rc8_d/s35cacheos_sh`.

The `svcs` command supports qualifiers to manipulate returned output. These qualifiers can be useful for scripting purposes. For example, the `-H` option suppresses column headings, and the `-o` option selects specific columns of output, such as state or FMRI. With these qualifiers, administrators can write scripts that select services in a particular state and perform a specific action. For example, administrators can automate the process of flagging services in a degraded or maintenance state and have the system send an email for prompt notification to operators.

## Service States

The first column in a `svcs` command listing shows the service state. There are seven different service states in SMF, as follows:

- **Uninitialized** — Before the service configuration is read or the restarter for the service is started, a service is in an uninitialized state.
- **Disabled** — A service is considered disabled when the service instance is not enabled and is not running.
- **Offline** — Service instances that are enabled but whose dependencies are not met are in an offline service state.
- **Online** — An online service is one that is enabled and successfully started.
- **Degraded** — Service instances that are enabled but are running at less than full capacity are in a degraded service state.
- **Maintenance** — When services encounter an error that must be resolved by the administrator, they are listed as being in a maintenance state.

- `Legacy_run` — This state is used only for services that are started through legacy scripts.

## Viewing Service Dependencies

With a number of possible methods for starting services in earlier Oracle Solaris versions, it was difficult to determine what processes a service depended upon to start successfully. It was also time-consuming and labor-intensive to determine what services were dependent on it. When services are created in Oracle Solaris 10, the service definitions in the XML manifest file list any other services that must be running prior to starting the service being enabled, along with the services that are dependent on it. These dependency statements define the relationships between services and the systematization of these dependencies makes it possible to provide precise fault containment. In addition, formulating dependencies allows for scalable and reproducible initialization processes. By clearly defining all dependencies, independent services can be started simultaneously, taking advantage of modern, highly parallel machines to speed system boot time.

The `svcs -d [service_name]` command shows services or processes on which a specified service depends. For example, to troubleshoot the `inetd` service, administrators can issue an `svcs -d network/inetd` command to see a list of services necessary for `inetd` to run. Any disabled services in the output listing must be enabled before `inetd` can be started. Once all service dependencies are resolved, the dynamic dependency checking in SMF brings the `inetd` service online.

The `svcs -D [service_name]` command shows which services or processes depend on the specified service. For example, to see what services require the `sendmail` service to be running before they can start, administrators can issue the `svcs -D network/smtp:sendmail` command. This command is also a powerful administrative tool that makes it possible to see beforehand the system impact of shutting down specific services.

To see running processes that are dependent on a service, administrators can use the command `svcs -p`, for both SMF services and legacy `init.d` scripts. This command is useful for determining that all processes associated with a service instance are stopped.

## Diagnosing Problems with Services

With SMF, administrators no longer have to search for daemons and `init` scripts, `grep` for processes in lists, or forage for configuration files. The SMF interface allows administrators to observe service states, dependencies, and properties, and provides powerful centralized tools for configuring and modifying services easily.

SMF contains a number of features that can help administrators diagnose problems with system services. If a service fails, the `svcs [service-name]` command lists it as being in a maintenance state. The `svcs -x` command displays all broken services along with an explanation as to why the service is broken. The display also lists any additional dependent services or instances that are impacted. Changing the command to `svcs -xv` provides verbose text as part of the explanation.

Services typically keep a log that contains the output from the method script for the service. If the program writes out errors to `stdout` or `stderr`, most errors appear in the service log. When troubleshooting service failures, it can be useful to look in the log for the service's restarter process, typically `svc.startd`, to see whether any errors are reported.

Since services are automatically restarted whenever possible, it may seem that a process refuses to die. If the service is defective, the service is placed in maintenance mode. However, the service is restarted if the process for the service is killed. The `svcadm` command should be used to stop the processes of any SMF service that should not be running.

## Securing Access to Services

Prior to Oracle Solaris 10 and the introduction of SMF, services were started by the `init` process, or the `inetd` run control script. In order to start these processes and make changes or otherwise administer system services, `root` access was required. As an alternative, file-based access control lists could be used to grant write access to an otherwise protected file. These approaches were problematic because they allowed a user to modify run control scripts without restriction, and possibly without leaving an audit trail. SMF provides the mechanisms to provide more powerful security.

### Using Role-Based Access Controls

By using role-based access controls (RBAC), administrators can delegate access to core service management functions based on user roles on the server. RBACs support the delegation of only the privileges required to fulfill specific tasks, and no more. Profiles for special-purpose administrators can be created in the areas of security, networking, firewall, backups, and system operation, as well as for advanced users. Oracle Solaris 10 includes two RBAC rights profiles intended for managing system services as soon as the operating system is installed. These profiles are:

- Service Management — A user or role assigned this rights profile can manipulate any service in any way. It corresponds to the `solaris.smf.manage` and `solaris.smf.modify` authorizations.
- Service Operator — A user or role assigned this rights profile has the ability to enable or disable any service instance on the system, as well as to request that its restart or refresh method be executed. It corresponds to the `solaris.smf.manage` and `solaris.smf.modify.framework` authorizations.

Additional granularity can be used to more finely control which SMF settings can be changed, such as methods, dependencies, applications, and frameworks. To limit access even further, authorizations can be restricted to modifying states for a given service, adding, deleting, or modifying service properties, or changing service properties.

Another advantage of using RBACs is that administrative actions taken are traceable to an authenticated individual instead of to just the `root` account. This provides greater accountability and more granular control of the system.

## Modifying Existing Services

The recommended method for making changes to service configurations stored in the SMF repository is to use the `svccfg` command. In particular, SMF manifest files that are provided with Oracle Solaris 10 or by ISVs should never be modified directly as the customizations may not be preserved across software upgrades. For those services that are managed by the `inetd` daemon, changes should be made by using the `inetadm` command.

Built in network services in Oracle Solaris 10 are handled through SMF, and SMF tools can be used to control and observe these services. The configuration data for network services is no longer stored in the `/etc/inet/inetd.conf` file — this data can now be found in the SMF repository.

The `inetadm` command can be used to change properties of converted `inetd` services. SMF commands such as `svcs` and `svcadm` can also be used to monitor and administer the service.

### The `inetconv` Utility

After an operating system upgrade to Oracle Solaris 10 or the installation of third-party software, any records found in the `/etc/inet/inetd.conf` file must be converted to SMF and imported into the SMF repository for the service to be available. The `inetconv` utility can be used to convert `inetd.conf` entries to SMF. This utility runs automatically during the first reboot after an operating system installation or upgrade, and should always be run after any changes are made to the `/etc/inet/inetd.conf` file.

## Comparing System Service Administration and Oracle Solaris SMF Commands

Table 2 contains examples of the commands used to administer system services in Oracle Solaris 8 and 9 with the comparable SMF commands found in Oracle Solaris 10.

| TABLE 2. FREQUENTLY USED COMMANDS FOR ADMINISTERING SMF SERVICES  |                                                                                                                                       |                                   |
|-------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| TASK                                                              | SOLARIS 8/9 OS PROCEDURE                                                                                                              | COMPARABLE SMF PROCEDURE          |
| Disabling system services such as <code>cron</code>               | <code>rm /etc/rc2.d/S75cron</code><br>(Repeat after every <code>cron</code> patch application and system upgrade.)                    | <code>svcadm disable cron</code>  |
| To re-enable system services such as <code>cron</code>            | Reinstall <code>/etc/rc2.d/S75cron</code>                                                                                             | <code>svcadm enable cron</code>   |
| To enable <code>inetd</code> services such as <code>finger</code> | Edit <code>/etc/inet/inetd.conf</code><br>Uncomment the service to be enabled,<br>Issue this command:<br><code>kill -HUP inetd</code> | <code>svcadm enable finger</code> |

TABLE 2. FREQUENTLY USED COMMANDS FOR ADMINISTERING SMF SERVICES

|                                      |                                                                         |                                                                                                                                            |
|--------------------------------------|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| To stop services                     | <code>/etc/init.d/ssh stop</code>                                       | <code>svcadm disable -t ssh</code><br>(The <code>-t</code> indicates that the requested action should be temporary until the next reboot.) |
| To start services                    | <code>/etc/init.d/ssh start</code>                                      | <code>svcadm enable -t ssh</code>                                                                                                          |
| To restart services                  | <code>/etc/init.d/ssh stop</code><br><code>/etc/init.d/ssh start</code> | <code>svcadm restart ssh</code>                                                                                                            |
| To refresh the service configuration | <code>kill -HUP `cat /var/run/ssh.pid`</code>                           | <code>svcadm restart ssh</code>                                                                                                            |

## Putting the Service Management Facility to Work

The following sections provide illustrations of how to implement and utilize SMF services.

### Creating a New SMF Service

The procedure below provides a step-by-step example of how to set up a new hypothetical SMF service, `foo`, and requires the following elements:

- The daemon `/opt/SUNWsmftest/bin/foo`. This is a program with no controlling `tty` that constantly runs and looks for service requests.
- The method `/opt/SUNWsmftest/lib/svc-foo.sh`. This is a shell script that starts the daemon (see below).
- The XML manifest file `/var/svc/manifest/system/foo.xml`. The manifest describes the service and its properties to the SMF repository.

NOTE: XML manifest files shipped with Oracle Solaris 10 should never be edited to modify built-in system services and doing so is unsupported. Changes to existing system services should be made using the `svccfg` command. Copies of XML manifest files should be edited to create new services. The manifest file in this example, `/var/svc/manifest/system/foo.xml`, is a copy of the `cron` XML manifest file with a few simple modifications. The syntax specification for SMF manifests can be found in the directory `/usr/share/lib/xml/dtd/service_bundle.dtd.1`.

The following steps illustrate how to create the new service `foo`.

1. Create the method shell script with the following content.

```
#!/sbin/sh
. /lib/svc/share/smf_include.sh
if [-x /opt/SUNWsmftest/bin/foo]; then
 /opt/SUNWsmftest/bin/foo
else
 echo "/opt/SUNWsmftest/bin/foo is missing or not executable."
 exit $SMF_EXIT_ERR_CONFIG
fi

exit $SMF_EXIT_OK
```

2. Create the XML manifest file. The following `foo.xml` file shows example content.

```

<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM
"/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type='manifest' name='SUNWcsu:foo'>

 <service
 name='system/foo'
 type='service'
 version='1'>

 <single_instance />

 <!-- foo_opt says that foo depends on local filesystem /opt -->
 <dependency
 name='foo_opt'
 type='service'
 grouping='require_all'
 restart_on='none'>
 <service_fmri value='svc:/system/filesystem/local' />
 </dependency>

 <!-- foo_cron says that foo depends on svc:/system/cron -->
 <dependency
 name='foo_cron'
 type='service'
 grouping='require_all'
 restart_on='refresh'>
 <service_fmri value='svc:/system/cron' />
 </dependency>

 <exec_method
 type='method'
 name='start'
 exec='/opt/SUNWsmftest/lib/svc-foo.sh'
 timeout_seconds='60'>
 <method_context>
 <method_credential user='root' group='root' />
 </method_context>
 </exec_method>

 <exec_method
 type='method'
 name='stop'
 exec=':kill'

```

3. Use the `svccfg` command to read in the XML manifest and add the `foo` service to the SMF repository.
4. Verify that the `foo` service has been created and defined within SMF.  

```
svccfg import /var/svc/manifest/system/foo.xml
```

```
svcs foo
STATE STIME FMRI
disabled 10:56:21 svc:/system/foo:default_foo
```

The instance shows as disabled because the instance element in the manifest specifies that the service instance should be created in the disabled state. This allows the administrator to precisely control the start of a new service, rather than having it start automatically.

5. Enable the service.

```
svcadm enable foo
```

6. Verify that the service is online.

```
svcs foo
```

7. Verify that the daemon is running.

```
ps -ef | grep foo
root 753 1 89 10:57:11 ? 0:48 /opt/SUNWsmftest/bin/foo
```

The `foo` service is now available to handle service requests.

## Converting Existing Services to SMF

In Oracle Solaris 10, many of the scripts formerly found in `/etc/init.d` and `/etc/rc*.d` have been removed as they are no longer needed to enable or disable a service. Entries from the `/etc/inittab` file have also been removed, so that the services formerly started there can be administered using SMF. However, services started from legacy `init.d` scripts cannot be administered through SMF.

Services that are started by traditional `rc` scripts tend to continue to work. These services appear in the output of the `svcs(1)` command, with an FMRI based on the path name of their `rc` script. They should be stopped and started by running the `rc` script directly.

SMF preserves compatibility with existing administrative practices wherever possible. Most scripts and `inittab` entries provided by an ISV or developed locally will continue to run. The services may not start at exactly the same point in the boot process. They are not started before the SMF services, so any service dependencies can be properly maintained. Problems may arise for scripts that depend on running before certain `rc` scripts previously provided in the Oracle Solaris OS. To derive the most benefits from SMF, it is

The Sun Optimized Web Stack is written to utilize SMF and contains manifest, method, and other component files, making it possible to utilize SAMP (Solaris, Apache HTTP Web Server, MySQL Database, and PHP software) without having to undergo a conversion process to support SMF. Organizations can rapidly deploy Web applications and take advantage of SMF features that make administration easier.

recommended that administrators convert existing services to SMF. The procedures to follow for doing so can be found here.

## Convert a SAMP Stack from `rc` Scripts to SMF

A SAMP stack is a bundle of commonly used open source applications that include Oracle Solaris, a Web server, a database, and a scripting language for dynamic Web pages. A bundle of open source Web tier components called the Sun Optimized Web Stack that contains Apache HTTP Web Server, MySQL database, and PHP, along with a number of other popular open source applications, is available for Oracle Solaris platforms.

The Sun Optimized Web Stack provides built-in support for SMF, so it is no longer necessary to go through the process of converting SAMP components. However, the steps for converting the Apache HTTP Web Server component to support the SMF model are outlined below. All commands should be executed as `root`. The example uses the service name `csk-httpd` to indicate that this is a Sun Optimized Web Stack service and to distinguish it from the `httpd` service that already exists.

1. Create a manifest file named `/var/svc/manifest/network/cskapache2.xml` with the following contents:

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/
service_bundle.dtd.1">
<!--
 Copyright 2006-2007 Sun Microsystems, Inc. All rights reserved.
 CSKapache2 manifest - should reside in /var/svc/manifest/network.
-->
<service_bundle type='manifest' name='CSKamp:apache'>
<service
 name='network/csk-http'
 type='service'
 version='1'>
<!--
 Because we may have multiple instances of
 network/http provided by different implementations,
 we keep dependencies and methods within the
 instance.
-->
<instance name='CSKapache2' enabled='false'>
<!--
 Wait for network interfaces to be initialized.
-->
```

```
<dependency name='network'
 grouping='require_all'
 restart_on='error'
 type='service'>
 <service_fmri value='svc:/milestone/network:default' />
</dependency>
<!--
 Wait for all local filesystems to be mounted.
-->
<dependency name='filesystem-local'
 grouping='require_all'
 restart_on='none'
 type='service'>
 <service_fmri
 value='svc:/system/filesystem/local:default' />
</dependency>
<!--
 Wait for automounting to be available, as we may be serving data
 from home directories or other remote filesystems.
-->
<dependency name='autofs'
 grouping='optional_all'
 restart_on='error'
 type='service'>
 <service_fmri
 value='svc:/system/filesystem/autofs:default' />
</dependency>
<exec_method
 type='method'
 name='start'
 exec='/opt/coolstack/lib/svc/method/svc-cskapache2 start'
 timeout_seconds='60'>
 <method_context>
 <method_credential
 user='webservd' group='webservd'
 privileges='basic,!proc_session,!proc_info,!file_link_any,net_privaddr' />
 </method_context>
 </exec_method>
<exec_method
```

```

 type='method'
 name='stop'
 exec='/opt/coolstack/lib/svc/method/svc-cskapache2 stop'
 timeout_seconds='60'>
 <method_context />
 </exec_method>
 <exec_method
 type='method'
 name='refresh'
 exec='/opt/coolstack/lib/svc/method/svc-cskapache2 refresh'
 timeout_seconds='60'>
 <method_context />
 </exec_method>
 <property_group name='httpd' type='application'>
 <stability value='Evolving' />
 <propval name='ssl' type='boolean' value='false' />
 </property_group>
 <property_group name='startd' type='framework'>
 <!-- sub-process core dumps shouldn't restart session -->
 <propval name='ignore_error' type='astring' value='core,signal' />
 </property_group>
</instance>
<stability value='Evolving' />
<template>
 <common_name>
 <loctext xml:lang='C'>
 Apache 2 HTTP server
 </loctext>
 </common_name>
 <documentation>
 <manpage title='httpd' section='8'
 manpath='/opt/coolstack/apache2/man' />
 <doc_link name='apache.org'
 uri='http://httpd.apache.org' />
 </documentation>
</template>
</service>
</service_bundle>

```

2. Create the `/svc` and `/svc/method` directories under the `/opt/coolstack/lib` directory

3. Create a method file called `/opt/coolstack/lib/svc/method/svc-cskapache2`. This file is referenced in the manifest. The `svc-cskapache2` method file assumes a certain name and location for the Apache configuration and pid file. It requires edits if the default settings from Web Stack are not to be used. It should be created as an executable and contain the following:

```
#!/sbin/sh
#
Copyright 2004-2007 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
#
ident "@(#)http-apache2 1.2 04/11/11 SMI"
Modified for apache in CSKamp package of Cool Stack
This file should reside in /opt/coolstack/lib/svc/method

. /lib/svc/share/smf_include.sh

APACHE_HOME=/opt/coolstack/apache2
CONF_FILE=$APACHE_HOME/conf/httpd.conf
PIDFILE=$APACHE_HOME/logs/httpd.pid

[! -f ${CONF_FILE}] && exit $SMF_EXIT_ERR_CONFIG

case "$1" in
 start)
 /bin/rm -f ${PIDFILE}
 cmd="start"
 ;;
 refresh)
 cmd="graceful"
 ;;
 stop)
 cmd="stop"
 ;;
 *)
 echo "Usage: $0 {start|stop|refresh}"
 exit 1
 ;;
esac

exec ${APACHE_HOME}/bin/apachectl $cmd 2>&1
```

4. The Apache software should be started as user `webservd`, not as `root`. To ensure that the `webservd` account can write to the log directory and the pid files, change the ownerships on those files. The log files reside in `/opt/coolstack/apache2/logs` by default. The commands to change the file ownerships are as follows:

```
cd /opt/coolstack/apache2
chown -R webservd logs
chgrp -R webservd logs
```

5. There is an Apache2 service that ships with Oracle Solaris 10, and the Web Stack service should not be allowed to conflict with it. By default, the Oracle Solaris service is disabled. To make sure that it has not been enabled, issue the following command:

```
svcs |grep http
```

6. If no output is printed, then it is disabled. If the output contains either of the following two lines, the service is running:

```
maintenance 11:47:11 svc:/network/http:apache2
online 11:47:11 svc:/network/http:apache2
```

7. If the service is running, disable it with the following command. The system should return the follow message:

```
svcadm -v disable http
svc:/network/http:apache2 disabled
```

8. Import the new service configuration:

```
svccfg -v import /var/svc/manifest/network/cskapache2.xml
```

9. Start the service:

```
svcadm -v enable csk-http
```

10. If the service starts successfully, a `svcs` command should show `httpd` processes running. A log of the service startup can be found in the `/var/svc/log/network-csk-http:CSKapache2.log` file.

## Enabling `inetd` Services

SMF contains support for services that were started by the `inetd` process in Oracle Solaris 8 and 9. For example, to start the UNIX `finger` utility service in Oracle Solaris 8 and 9, administrators uncommented the line for `finger` in the `/etc/inet/inetd.conf` file, and then killed the `inetd` process in order to restart `inetd` and start the `finger` process.

With SMF support for `inetd` services in Oracle Solaris 10, administrators can issue the following command to enable the `finger` process:

```
svcadm enable network/finger:default
```

## Refreshing Service Configurations

Sometimes it is necessary for system administrators to make configuration changes to system services. In Oracle Solaris 8 and 9, the system service processes must be killed to force a restart with new configuration information. SMF makes it possible for administrators to refresh service configurations without killing services.

To refresh a system service, use the following command (this example uses the secure shell):

```
svcadm restart ssh
```

## Best Practices for SMF

Organizations wishing to implement best practices for utilizing SMF should adhere to the following guidelines:

- Do not create new files or modify files in the `/etc/init.d` and `rc?.d` scripts, `/etc/inittab` entries, and `/etc/inetd.conf` entries.
- Do not place non-private files in the `/etc/default` directory or the `/etc/*` configuration files.
- Do not add files, including non-private plain-text files, to the `/etc` file hierarchy.
- Transition legacy services reliant upon `/etc/init.d`, `rc` scripts, or other traditional start-up mechanisms to SMF.
- Create services in manifests added to the `/var/svc/manifest` directory. Oracle recommends including a line in the service manifest that disables the service by default. Doing so grants administrators greater control over when a service is started.
- Configure services to run with the most restricted context and the least possible privileges. See `privileges(5)`, `smf_method(5)` for additional details.
- Configure services with service-related RBAC authorizations, as appropriate, by providing service specific values for the `action_authorization`, `modify_authorization`, `value_authorization`, and `read_authorization` of property groups. See `smf_security(5)`.
- Make sure that authorizations follow the form `solaris.smf.{manage, modify, value, read}.service` respectively. These authorizations must be delivered into an appropriate new or existing rights profile.

## Conclusion

SMF removes developer and administrator dependency upon error prone `/etc/rc?.d` scripts and file-based system and service configurations within Oracle Solaris. New and existing services employing SMF form a unified model for services delivery within the operating system. Utilizing the SMF framework

results in services that are visible and manageable using SMF-specific command-line utilities and provides the ability to easily identify misconfigured, misbehaving, and failed services.

Administrators can rely on automated restart — in dependency order — of services that fail as the result of hardware and software faults and administrative error. The configuration of system services is consistent through the use of a central configuration repository, and built-in security features facilitate the secure delegation of administration to non-root users.

## About the Author

Brian Down serves as a Chief Technologist in Sun's North American Systems Line of Business. He is also the author of *Protecting Investments Through Technology Advancements* and co-author of *Migrating to the Solaris Operating System: The Discipline of Unix-to-Unix Migrations*. Brian is currently the Board Chair for the Consortium for Software Engineering Research (CSER) and is a board member of the Optical Research Advanced Network for Ontario (ORANO).

## References

To learn more about the Oracle Solaris 10 OS and the Service Management Facility, see the references listed in Table 3.

**TABLE 3. REFERENCES FOR MORE INFORMATION**

<b>Web Sites</b>	
Oracle Solaris	<a href="http://oracle.com/solaris">oracle.com/solaris</a>
Oracle Solaris Technical Content	<a href="http://developers.sun.com/solaris">developers.sun.com/solaris</a>
SMF Quickstart	<a href="http://sun.com/bigadmin/content/selfheal/smf-quickstart.html">sun.com/bigadmin/content/selfheal/smf-quickstart.html</a>
SMF Service Developer Introduction	<a href="http://sun.com/bigadmin/content/selfheal/sdev_intro.html">sun.com/bigadmin/content/selfheal/sdev_intro.html</a>
System Administration Guide: Managing Services (Overview)	<a href="http://docs.sun.com/app/docs/doc/817-1985/hbrunlevels-25516?a=view">docs.sun.com/app/docs/doc/817-1985/hbrunlevels-25516?a=view</a>
System Administration Guide: Managing Services (Tasks)	<a href="http://docs.sun.com/app/docs/doc/817-1985/faauf?a=view">docs.sun.com/app/docs/doc/817-1985/faauf?a=view</a>
Converted Services: Manifests and Methods	<a href="http://hub.opensolaris.org/bin/view/Community+Group+smf/manifests">hub.opensolaris.org/bin/view/Community+Group+smf/manifests</a>



Management of Systems and Services Made  
Simple with the Oracle Solaris Service  
Management Facility  
June 2010  
Author: Brian Down  
Contributor: Mary Jane Greenfield, Stéphanie  
Choyer

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
oracle.com



Oracle is committed to developing practices and products that help protect the environment

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

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

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. 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. UNIX is a registered trademark licensed through X/Open Company, Ltd. 0610

**SOFTWARE. HARDWARE. COMPLETE.**