HOW to CREATE an SMF MANIFEST

> **Solaris™ 10** How To Guides

Robert Lor, ISV Engineering
Angel Camacho, Solaris Marketing
Version 1.1 | *Last updated: 08/24/07*

About This Solaris How To Guide

The SMF How to Guide explains what a basic SMF service manifest is and shows how to develop a service manifest for a given application. The service manifest for the PostgreSQL database included in Solaris 10 is the example in this guide.

After reading this guide, the user will be able to create a simple service manifest to manage an application on a Solaris 10 system.

Another Solaris how to guide, *"Easier System Administration with Service Management Facility,"* is available at sun.com/software/solaris/howtoguides/servicemgmthowto.jsp for system administrators who wish to learn more about using SMF to manage services on Solaris 10 systems.

Contents

SMF Manifest How To Guide

Introduction

The *Service Management Facility* (SMF) is a core component of the new Predictive Self Healing (PSH) set of technologies introduced in Solaris 10. With SMF, system administrators can use simple command line utilities to easily identify, observe, and manage the services provided by the system and the system itself. A brief introduction to SMF along with examples is given in the *"Easier System Administration with Service Management Facility"* How to Guide, available at: sun.com/software/solaris/howtoguides/servicemgmthowto.jsp.

Anatomy of an SMF Manifest

The *service manifest* is an XML file that stores the relationship and dependency information between software services on a Solaris system. The service manifest also describes the conditions under which failed services may be automatically restarted. SMF uses this information to manage services and to determine root causes of service failures. A separate service manifest is required per service/application. You can get an empty template from: sun.com/bigadmin/content/selfheal/smf-hds/template.xml.

This section contains the relevant sections for most manifests. For a complete view, see the XML schema located at /usr/share/lib/xml/dtd/service_bundle.dtd.1

Section	XML Code

XML Header	`<?xml version="1.0"?>`
	`<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">`

Service Bundle	`<service_bundle type='manifest' name='`***REPLACE_ME***`'>`

Service Name**	`<service`
	`  name='`***REPLACE_ME***`'`
	`  type='service'`
	`  version='1'>`

Default Startup Flag*	`  <create_default_instance enabled='false' />`
Single Instance Flag*	`  <single_instance />`

Service Dependency**	`  <dependency`
	`    name='network'`
	`    grouping='require_all'`
	`    restart_on='none'`
	`    type='service'>`
	`      <service_fmri value='svc:/milestone/network:default' />`
	`  </dependency>`

Exec Methods**	`  <exec_method`
	`    type='method'`
	`    name='start'`
	`    exec='/lib/svc/method/`***REPLACE_ME***` start'`
	`    timeout_seconds='`***REPLACE_ME***`' />`
	`  <exec_method`
	`    type='method'`

Section (cont.)	XML Code (cont.)
Exec Methods** (cont.)	`name='stop'` `exec='/lib/svc/method/REPLACE_ME stop'` `timeout_seconds='REPLACE_ME' />` `<exec_method` `type='method'` `name='refresh'` `exec='/lib/svc/method/REPLACE_ME refresh'` `timeout_seconds='REPLACE_ME' />`
Method Context*	`<method_context>` `<method_credential user='REPLACE_ME ' group='REPLACE_ME' />` `</method_context>`
Property Group**	`<property_group name='REPLACE_ME' type='application'>` `<propval name='REPLACE_ME' type='astring' value='REPLACE_ME' />` `<propval name='REPLACE_ME ' type='astring' value='REPLACE_ME ' />` `</property_group>`
Stability*	`<stability value='Evolving' />`
Template*	`<template>` `<common_name>` `<loctext xml:lang='C'>REPLACE_ME </loctext>` `</common_name>` `<documentation>` `<manpage title='REPLACE_ME ' section='REPLACE_ME ' />` `<doc_link name='REPLACE_ME' uri='REPLACE_ME' />` `</documentation>` `</template>`
	`</service>`
	`</service_bundle>`

* Can occur exactly 0 or 1 time.
** Can occur 0 or more times.

Explanation of the Manifest sections

We will now explain each section of the manifest, describe each property, and list the possible values.

XML Header—Standard XM header.

Service Bundle—Specifies how this file should be used by the SMF framework.

Required Attributes:

Attributes	Description	Possible Values
type		'archive', 'manifest', 'profile'
name	A name for the bundle	

Service Name—Contains the set of instances defined by default for this service, an optional method for execution context, any default methods, the template, and various restrictions or advice applicable at installation.
Note: The method execution context and template elements are required for service_bundle documents with type "manifest" but are optional for "profile" or "archive" documents.

Required Attributes:

Attributes	Description	Possible Values
name	The canonical name for the service	See note below on naming conventions
version	The integer version of this service	
type		'service', 'restarter, 'milestone'

There are recommended conventions for naming a service. Services fall into different categories and these categories are listed in /var/svc/manifest; they are application, milestone, platform, system, device, network, and site. So, the service name includes the category (can be in multiple levels) plus the actual service name separated by '/'. For example, the service name for PostgreSQL is “application/database/postgresql.”

Instance Name—The service instance is the object representing a software component that will run on the system if enabled. It contains an enabled element, a set of dependencies on other services, potentially customized methods or configuration data, an optional method context, and a pointer to its restarter. If no restarter is specified, the master restarter, svc.startd(1M), is assumed to be responsible for the service.

Required Attributes:

Attributes	Description	Possible Values
name	The canonical name for this instance of the service	
enabled	The initial value for the enabled state of this instance	'true', 'false'

Service Dependency—This element identifies a group of FMRIs upon which the service is in some sense dependent.

Required Attributes:

Attributes	Description	Possible Values
name	The name of this dependency	
grouping	The relationship between the various FMRIs grouped here; "require_all" of the FMRIs to be online, "require_any" of the FMRIs to be online, or "exclude_all" of the FMRIs from being online for the dependency to be satisfied. "optional_all" dependencies are satisfied when all of the FMRIs are either online or unable to come online (because they are disabled, misconfigured, or one of their dependencies is unable to come online)	'require_all', 'require_any', 'exclude_all', 'optional_all'
restart_on	The type of events from the FMRIs that the service should be restarted for. Possible values are: • 'error' restarts the service if the dependency is restarted due to hardware fault • 'restart' restarts the service if the dependency is restarted for any reason, including hardware fault • 'refresh' restarts the service if the dependency is refreshed or restarted for any reason • 'none' will never restart the service due to dependency state changes	'error', 'restart', 'refresh', 'none'
type	The type of dependency	

Exec Methods—This element describes one of the methods used by the designated restarter to act on the service instance. The interpretation is left to the restarter as to which particular service instance is delegated. It contains a set of attributes, an optional method context, and an optional stability element for the optional properties that can be included.

Required Attributes:

Attributes	Description	Possible Values
type	The type of method	'method' or 'monitor'
name	Name of this execution method. The method names are usually a defined interface of the restarter to which an instance of this service is delegated.	
exec	The string identifying the action to take	
timeout_seconds	Duration, in seconds, to wait for this method to complete. A '0' or '-1' denotes an infinite timeout.	Integer value

Method Context—This element combines credential and resource management attributes for execution methods.

Required Attributes:

Attributes	Description	Possible Values
User	The user ID in numeric or text form	Alphanumeric value

Property Group—This element is for a set of related properties on a service or instance.

Required Attributes:

Attributes	Description	Possible Values
name	The name of this property group	Text
type	A category for this property group	Framework, implementation, template, application

Stability—This element associates Sun's stability level with the parent element.

Required Attributes:

Attributes	Description	Possible Values
value	The stability level of the parent element	Standard, Stable, Evolving, Unstable, External, Obsolete

Template—The template contains a collection of metadata about the service. It contains a localizable string that serves as a common, human-readable name for the service. The template may optionally contain a longer localizable description of the service and/or a collection of links to documentation, either in the form of manual pages or in the form of URI specifications to external documentation sources.

SMF for PostgreSQL

Now that we understand the template of a service manifest, we can start to build one. This section shows how the PostgreSQL manifest was created based on the template in the previous section. We begin with a manifest that only allows one running instance of PostgreSQL. Later, we create the manifest that allows multiple running instances of PostgreSQL.

The manifest uses a shell script to control PostgreSQL, and this script is shown in Example 3.

Example 1: Manifest for a single instance of PostgreSQL

Section	XML Code
XML Header	`<?xml version="1.0"?>`
	`<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">`
Service Bundle	`<service_bundle type='manifest' name='postgresql'>`
Service Name	`<service`
	`  name='application/database/postgresql'`
	`  type='service'`
	`  version='1'>`
Default Startup Flag	`  <create_default_instance enabled='false' />`
Single Instance Flag	`  <single_instance />`
Service Dependency	`  <dependency`
	`    name='network'`
	`    grouping='require_all'`
	`    restart_on='none'`
	`    type='service'>`
	`      <service_fmri value='svc:/milestone/network:default' />`
	`  </dependency>`
Exec Methods	`  <exec_method`
	`    type='method'`
	`    name='start'`
	`    exec='/lib/svc/method/postgresql start'`
	`    timeout_seconds='300' />`
	`  <exec_method`
	`    type='method'`
	`    name='stop'`
	`    exec='/lib/svc/method/postgresql stop'`
	`    timeout_seconds='300' />`
	`  <exec_method`
	`    type='method'`
	`    name='refresh'`
	`    exec='/lib/svc/method/postgresql refresh'`
	`    timeout_seconds='60' />`
Method Context	`  <method_context>`
	`    <method_credential user='postgres' group='postgres' />`
	`  </method_context>`
Property Group	`  <property_group name='postgresql' type='application'>`
	`    <propval name='data' type='astring' value='/var/lib/pgsql/data' />`
	`    <propval name='log' type='astring' value='postmaster.log' />`
	`  </property_group>`
Stability	`  <stability value='Evolving' />`

Section (cont.)	XML Code (cont.)
Template	`  <template>`
	`    <common_name>`
	`      <loctext xml:lang='C'>PostgreSQL RDBMS</loctext>`
	`    </common_name>`
	`    <documentation>`
	`      <manpage title='postgres' section='1M' />`
	`      <doc_link name='postgresql.org' uri='http://postgresql.org' />`
	`    </documentation>`
	`  </template>`
	`</service>`
	`</service_bundle>`

The structure of this manifest is the same as the template in section 3. We have customized it by inserting PostgreSQL specific values for all occurrences of ***REPLACE_ME***.

Example 2: Manifest for more than one instance of PostgreSQL

There are a few differences in the multiple instance manifest when compared to the single instance version.

- The single_instance element should be removed so SMF will allow more than one running PostgreSQL instance
- The create_default_instance is also removed and the instance element is used instead
- As explained in section 3, most elements can exist at both the service or instance level. Elements that are common for all instances stay at the service level to avoid duplication. These include dependency, exec_method, and template.

Section	XML Code
XML Header	`<?xml version="1.0"?>`
	`<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">`
Service Bundle	`<service_bundle type='manifest' name='postgresql'>`
Service Name	`<service`
	`  name='application/database/postgresql'`
	`  type='service'`
	`  version='1'>`
Service Dependency	`    <dependency`
	`      name='network'`
	`      grouping='require_all'`
	`      restart_on='none'`
	`      type='service'>`
	`        <service_fmri value='svc:/milestone/network:default' />`
	`    </dependency>`

Section (cont.)	XML Code (cont.)
Exec Methods	`<exec_method`
	`  type='method'`
	`  name='start'`
	`  exec='/lib/svc/method/postgresql start'`
	`  timeout_seconds='300' />`
	`<exec_method`
	`  type='method'`
	`  name='stop'`
	`  exec='/lib/svc/method/postgresql stop'`
	`  timeout_seconds='300' />`
	`<exec_method`
	`  type='method'`
	`  name='refresh'`
	`  exec='/lib/svc/method/postgresql refresh'`
	`  timeout_seconds='60' />`
Instance Name #1	`<instance name='default' enabled='false'>`
Method Context	`  <method_context>`
	`    <method_credential user='postgres' group='postgres' />`
	`  </method_context>`
Property Group	`  <property_group name='postgresql' type='application'>`
	`    <propval name='data' type='astring' value='/var/lib/pgsql/data' />`
	`    <propval name='log' type='astring' value='postmaster.log' />`
	`  </property_group>`
	`</instance>`
Instance Name #2	`<instance name='postgres' enabled='false'>`
Method Context	`  <method_context>`
	`    <method_credential user='postgres' group='postgres' />`
	`  </method_context>`
Property Group	`  <property_group name='postgresql' type='application'>`
	`    <propval name='data' type='astring' value='/var/lib/pgsql/data2' />`
	`    <propval name='log' type='astring' value='postmaster.log' />`
	`  </property_group>`
	`</instance>`
Stability	`<stability value='Evolving' />`

Section (cont.)	XML Code (cont.)
Template	`<template>` `<common_name>` `<loctext xml:lang='C'>PostgreSQL RDBMS</loctext>` `</common_name>` `<documentation>` `<manpage title='postgres' section='1M' />` `<doc_link name='postgresql.org' uri='http://postgresql.org' />` `</documentation>` `</template>` `</service>` `</service_bundle>`

Save this manifest for later use:

1. Save this manifest into a file called postgresql.xml using your favorite text editor, in your home directory. You can download this manifest from sun.com/bigadmin/content/selfheal/smf-hds/postgres/postgresql.xml.
2. Check if the /var/svc/manifest/application/database directory exists:

```
my_server# ls /var/svc/manifest/application/database
/var/svc/manifest/application/database: No such file or directory
```

3. If it doesn't exist, create it:

```
my_server# mv /var/svc/manifest/application
my_server# mkdir database
```

4. Change permission of the /var/svc/manifest/application/database directory to 555:

```
my_server# chmod 555 database
```

5. Place in the service manifest file in the /var/svc/manifest/application/database directory:

```
my_server# cd database
my_server# mv {from-home-directory}/postgresql.xml
```

The default instance of the manifest assumes the following:

• The database user is *postgres*
• The database cluster directory is */var/lib/pgsql/data*
• The postmaster log file is *postmaster.log*

If any of these are different, update your service manifest accordingly or use the svccfg command to change this property after the manifest has been imported. Step 7 in section 5 shows you how to do this.

Example 3: Shell script used to control PostgreSQL

This shell script (referred to as /lib/svc/method/postgresql) is used in the exec_method element in both of the examples above. The script controls the start, stop, and refresh actions for PostgreSQL which maps to enable, disable, and refresh in the SMF svcadm command respectively. More on this command later.

```
#!/sbin/sh
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "%Z%%M% %I%     %E SMI"

. /lib/svc/share/smf_include.sh

# SMF_FMRI is the name of the target service. This allows multiple instances
# to use the same script.

getproparg() {
        val=`svcprop -p $1 $SMF_FMRI`
        [ -n "$val" ] && echo $val
}

PGBIN=/usr/bin
PGDATA=`getproparg postgresql/data`
PGLOG=`getproparg postgresql/log`
```

```
if [ -z $SMF_FMRI ]; then
         echo "SMF framework variables are not initialized."
         exit $SMF_EXIT_ERR
fi

if [ -z $PGDATA ]; then
         echo "postgresql/data property not set"
         exit $SMF_EXIT_ERR_CONFIG
fi

if [ -z $PGLOG ]; then
         echo "postgresql/log property not set"
         exit $SMF_EXIT_ERR_CONFIG
fi

case "$1" in
'start')
         $PGBIN/pg_ctl -D $PGDATA -l $PGDATA/$PGLOG start
         ;;

'stop')
         $PGBIN/pg_ctl -D $PGDATA stop
         ;;

'refresh')
         $PGBIN/pg_ctl -D $PGDATA reload
         ;;

*)
         echo $"Usage: $0 {start|refresh}"
         exit 1
         ;;

esac
exit $SMF_EXIT_OK
```

1. Save this shell script using your favorite text editor into your home directory to a file called postgresql. You can download the script from sun.com/bigadmin/content/selfheal/smf-hds/postgres/postgresql.txt.
2. Change permission of the /lib/svc/method directory to 555. You need to have the appropriate write privileges to copy files into this directory:

   ```
   my_server# chmod 555 /lib/svc/method
   ```

3. Place in the shell script in the /lib/svc/method directory:

   ```
   my_server# cp {from-home-directory}/postgresql /lib/svc/method
   ```

Making it All Work

Now we are ready to get the service manifest created in Example 2 working with PostgreSQL. We will be working with two instances of the PostgreSQL database.

1. First, we validate the manifest before importing it:

```
my_server# cd /var/svc/manifest/application/database
my_server# svccfg validate postgresql.xml
```

2. Next, import the manifest into the SMF repository:

```
my_server# svccfg import postgresql.xml
```

3. Check the state of the service. By default, both service instances are disabled:

```
my_server# svcs postgresql
STATE          STIME    FMRI
disabled       19:00:01 svc:/application/database/postgresql:postgres
disabled       19:00:01 svc:/application/database/postgresql:default
my_server# svcs postgresql:default
STATE          STIME    FMRI
disabled       19:00:01 svc:/application/database/postgresql:default
```

Note: The full service name or fault management resource identifier (FMRI) for both instances are svc:/application/database/postgresql:default and svc:/application/database/postgresql:postgres respectively, but they can be referred to as postgresql:default and postgresql:postgres. For more information on using the svcs command see the "*Easier System Administration with Service Management Facility*" How to Guide at: sun.com/software/solaris/howtoguides/servicemgmthowto.jsp.

4. Start the PostgreSQL service for the first (default) instance:

```
my_server# svcadm enable postgresql:default
```

5. If service is maintenance mode, then there will be an error starting the service; check /var/svc/log for a log file starting with the service name:

```
my_server# svcs postgresql
STATE          STIME    FMRI
disabled       19:00:01 svc:/application/database/postgresql:postgres
maintenance    19:00:39 svc:/application/database/postgresql:default
my_server# cd /var/svc/log
my_server# ls *postgres*
application-database-postgresql:default.log
application-database-postgresql:postgres.log
my_server# tail application-database-postgresql:default.log
[ Feb  8 19:00:39 Method "start" exited with status 0 ]
[ Feb  8 19:00:39 Stopping because all processes in service exited. ]
[ Feb  8 19:00:39 Executing stop method ("/lib/svc/method/postgresql stop") ]
pg_ctl: cannot be run as root
Please log in (using, e.g., "su") as the (unprivileged) user that will
own the server process.
```

```
[ Feb  8 19:00:39 Method "stop" exited with status 0 ]
[ Feb  8 19:00:39 Restarting too quickly, changing state to maintenance ]
```

From this point on, the PostgreSQL process is controlled by Solaris SMF, and the system administrator can change its state by using the svcadm command. If the service is terminated for any reason, the SMF restarter daemon will attempt to restart it. Note that at system reboot, the PostgreSQL service will be started automatically unless it is disabled, eg., if it has been disabled before the reboot.

6. Start the second instance of the PostgreSQL service:

```
my_server# svcadm enable postgresql:postgres
```

7. Confirm they are working (in this example both instances are working):

```
my_server# svcs postgresql
STATE           STIME    FMRI
on-line        19:00:01 svc:/application/database/postgresql:postgres
on-line        19:02:38 svc:/application/database/postgresql:default
```

8. You can change any of the configurable properties (user, data, or log) in postgresql.xml dynamically after the manifest has been imported. To do so, you should first disable the service, then change the property value and refresh it, and finally restart the service. The next two examples show how we can change the administrative user and data directory.
9. To change the Solaris administrative user to “foo” for the default instance, perform the following steps:

```
my_server# svcadm disable postgresql:default
my_server# svccfg -s postgresql:default setprop method_context/user = "foo"
my_server# svcadm refresh postgresql:default
my_server# svcadm enable postgresql:default
```

10. To change the database cluster directory to “/pgdata” for the default instance, perform the following steps:

```
my_server# svcadm disable postgresql:default
my_server# svccfg -s postgresql:default setprop postgresql/data = "/pgdata"
my_server# svcadm refresh postgresql:default
my_server# svcadm enable postgresql:default
```

For More information

For more information on SMF and Predictive Self Healing, please visit the following websites:

Web Resources	
Solaris Website	sun.com/solaris/
SMF Community page	opensolaris.org/os/community/smf/
PSH (SMF/FMA) Bigadmin page	sun.com/bigadmin/content/selfheal/
SMF Quickstart	sun.com/bigadmin/content/selfheal/smf-quickstart.html
SMF Service Developer Introduction	sun.com/bigadmin/content/selfheal/sdev_intro.html
System Administration Guide: Managing Services	docs.sun.com/app/docs/doc/817-1985/6mhm8o5rl?a=view
Peter Baer Galvin's SMF article in Sys Admin	samag.com/documents/s=9766/sam0506i/0506i.htm
Blueprint: Service Management Facility (SMF) in the Solaris 10 Operating System	sun.com/blueprints/0206/819-5150.pdf

sun.com/solaris

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com