# System Configuration SMF service Nodename

design specification

Version 0.4

Authors:
John Fischer

# Table of Contents

# 1 Motivation

The legacy Solaris installer uses the system configuration parameters specified by the user to configure the system in the installation environment, and then copies the configured files to the installed disk (alternate root). This method puts the responsibility on the installer to know how to configure each of the software components where configuration has been specified, and the installer must also track all of the changes happening with the configuration files. This results in a model where what is able to be configured is highly restricted by the installer capabilities.

# 2 Strategy

## *2.1 Overview*

One design principle of the OpenSolaris installation architecture is the separation of software installation from software configuration. The installer should be responsible only for the installation of the software, and configuration should be the responsibility of the software components themselves.

This approach eases the requirement on the installer to have specific knowledge on each configuration piece, or even inclusion of the software components in the installation environment. The software components won't be required to provide a special interface to configure their software in an alternate root either. Ideally, configuration happens in the environment of the installed system, where the software will actually be running.

## *2.2 Installer*

For purposes of the Installer (Automated, GUI or Text), configuration of the system to be installed will be specified in the form of a set of configuration parameters provided in an SMF profile file called System Configuration Manifest. SMF properties configured by that profile contain particular system configuration parameters. The Installer will install OpenSolaris on a target system and store the System Configuration Manifest into the appropriate place for purposes of Early Manifest Import.

The process of configuring the target system will be carried out in two steps during the first boot of the system. As part of Early Manifest Import process, an SMF profile carrying system configuration will be applied. Afterward, the SMF services responsible for particular configuration areas will process the SMF properties and configure the system accordingly.

The responsibility of a particular configuration area will be decentralized and delegated to appropriate SMF services. There are several scenarios with respect to how SMF services will need to be modified in order to be able to take care of applying the system configuration parameters.

### 2.2.1 Configuration being configured via SMF service

Configuration is already being configured as a property of an existing SMF service. For instance terminal type is configured as **ttymon/terminal_type** SMF property of **svc:/system/console-login** SMF service. No SMF changes are needed.

### 2.2.2 Configuration not stored via existing SMF service but configured

Configuration is currently not stored in the form of SMF properties, but there is an existing SMF service which already applies the configuration. In that case, the storage backend will be changed from a flat file to an SMF property and the SMF property will become persistent as well as the authoritative place. Initial storage backend will be either removed or left as read only for purposes of backward compatibility. For example, **nodename(4)** is currently stored in the **/etc/nodename** file and handled by the **svc:/system/identity:node** SMF service. This is the scenario which is covered by this design document.

### 2.2.3 Configuration not stored via existing SMF service nor configured

Configuration is currently not stored in the form of SMF properties, there is an existing SMF service which could be enhanced to apply the configuration, but it is not feasible to change the storage backend. In that case, SMF properties will be only used for initial configuration, the SMF service will process them and flush the configuration into existing storage beckend. Networking configuration might be an example.

### 2.2.4 Configuration not stored because no SMF service available

Configuration is currently not stored in the form of SMF properties and there is no viable existing SMF service which could be enhanced to take care of configuration. In that case, a new SMF service will be introduced to take care of this configuration area. Creating a user account and configuring the root account is an example.

# 3 Scope

## *3.1 In Scope*

This document covers the design of the System Configuration SMF service which will serve for configuring

- nodename

All Installer scenarios will be addressed within this project, as well as other non-install related ON components.

## *3.2 Out of Scope*

This document does not cover how to deal with components outside of the install or ON consolidations. Every effort will be made to inform other teams via flag day announcements. However, actual modification of these other components remain the responsibility of those teams.

# 4 Problem statement

Update the svc:/system/identity:node SMF service to take care of setting the nodename of the system installed by means of the Installer technologies. Furthermore, update various components that currently reference /etc/nodename to use the new mechanism, namely cvcd, setuname, metaset and nodename(4) for nodename.

# 5 Requirements

The nodename will be configurable via svc:/system/identity:node SMF service using uname – see uname(1) for details.

# 6 Assumptions

## *6.1 Alternate root awareness*

It is assumed that the mechanism will always run within the target environment and thus it will not be designed to be alternate root aware. This allows the project to take advantage of various tools during implementation which are not alternate root aware – for instance uname(1).

If there is a requirement for supporting scenarios where alternate root awareness would be required (e.g. Distribution Constructor), alternative solutions are to be considered (e.g. chroot(1), zones, different implementation for alternate root environments).

# 7 Dependencies

## *7.1 Early Manifest Import project*

Since system configuration will be applied in the form of an SMF profile during first boot, it is necessary that all SMF properties are properly populated before the System Configuration SMF service comes online. That will be assured by taking advantage of the Early Manifest Import project (PSARC 2010/013) which will take care of applying the SMF profile before any SMF service is started.

# 8 Solution

## *8.1 Overview*

svc:/system/identity:node SMF service will configure the nodename during boot of the system. The start methods of these SMF service will read the configuration parameter from the SMF properties and will call the appropriate CLI to set the configuration itself for nodename that would be 'uname -S'.

No validation of the of nodename will be done. It is assumed that 'uname -S' will take care of validating the parameter and report failures when invalid characters are provided. Error messages will be captured in SMF log file.

Once the configuration is applied, the SMF properties will be the persistent back-end storage. The nodename(4) file will no longer be the persistent back-end storage. In fact, the nodename file will no

longer be a deliverable for install or various ON tools.

## *8.2 Format of SMF properties*

### 8.2.1 Nodename

```
<service_bundle type="profile" name="default">
    <service name="system/identity" version="1" type="service">
        <instance name="domain" enabled="true">

            <!-- The following property group is used at install
                 time to configure the defaultdomain for the system -->

            <property_group name="config" type="application">
                <propval name="defaultdomain" type="astring" value=""/>
            </property_group>
        </instance>
    </service>
</service_bundle>
```

## *8.3 Default configuration*

svc:/system/identity:node SMF service will not specify default values for particular configurable parameters.

For purposes of the Automated Installer, default setting will be delivered by the System Configuration portion of **default.xml** AI manifest – i.e.,

- nodename: unknown

See 10.2 SC portion of AI default.xml for full specification of System Configuration portion of **default.xml** AI manifest.

The default value for the nodename which is not specified in the manifest will be determined by SMF service. The default value for the nodename will be unknown for the SMF service.

## *8.4 Error handling*

### 8.4.1 Fatal errors

Fatal errors are conditions which prevent the svc:/system/identity:node SMF service from successfully configuring system according to specified properties. In this case, the SMF service will exit with

$EXIT_ERR_FATAL error code and the service will subsequently enter 'maintenance' state.

Following conditions will be considered as fatal errors:

- 'uname -S' fails due to _SYS_NMLN being exceeded
- multiple lines within the nodename

## *8.5 ON components*

### 8.5.1 cvcd

The cvcd or virtual console daemon for SPARC hardware currently uses /etc/nodename as a fall back mechanism if 'uname -n' returns an error because the hostname is not yet set. However, since the svc:/system/cvc SMF service is depended upon svc:/system/identity:node it will not follow the code path. Therefore, this project will simply remove the dependent code.

### 8.5.2 setuname

This utility is a specification in some standards that are no longer of interest to Solaris. Thus it will be deprecated and removed from the release by updating the SUNWcs package.

### 8.5.3 metaset

The metaset command only references nodename in the command usage output. This reference will be replaced with a reference to the SMF svc:/system/identity:node nodename property.

### 8.5.4 man pages

Various man pages will need to remove nodename and point to the appropriate service including metaset(1M) and nodename(4).

# 9 Deliverables

System Configuration SMF service will be delivered by the SUNWcs package – the following components will be included:

- SMF manifest (**/lib/svc/manifest/system/identity.xml**)
- SMF node method (**/lib/svc/method/identity-node**)

Also, System Configuration portion of **default.xml** AI manifest delivered by **pkg:/system/install/auto-install/auto-install-common** package will be modified accordingly – see 10.2 SC portion of AI default.xml. Updated cvcd, metaset and SUNWcs package manifest to remove nodename dependencies.

# 10 Appendix

## *10.1 SMF manifest*

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<!--
 Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 Use is subject to license terms.

 CDDL HEADER START

 The contents of this file are subject to the terms of the
 Common Development and Distribution License, Version 1.0 only
 (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

      ident "%Z%%M%     %I%   %E% SMI"

      NOTE:  This service manifest is not editable; its contents will
      be overwritten by package or patch operations, including
      operating system upgrade.  Make customizations in a different
      file.
-->

<service_bundle type='manifest' name='SUNWcsr:identity'>

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

      <dependency
            name='loopback-network'
            grouping='require_any'
            restart_on='none'
            type='service'>
            <service_fmri value='svc:/network/loopback' />
```

```
</dependency>

<exec_method
      type='method'
      name='stop'
      exec=':true'
      timeout_seconds='60' />

<property_group name='startd' type='framework'>
      <propval name='duration' type='astring' value='transient' />
</property_group>

<property_group name="config" type="application">
      <propval name="nodename" type="astring" value=""/>
</property_group>

<instance name='node' enabled='true'>

      <exec_method
            type='method'
            name='start'
            exec='/lib/svc/method/identity-node'
            timeout_seconds='60' />

      <template>
            <common_name>
                  <loctext xml:lang='C'>
                        system identity (nodename)
                  </loctext>
            </common_name>
            <documentation>
                  <manpage title='nodename' section='4'
                        manpath='/usr/share/man' />
            </documentation>
      </template>

</instance>

<instance name='domain' enabled='false'>

      <dependency
            name='fs'
            grouping='require_all'
            restart_on='none'
            type='service'>
            <service_fmri value='svc:/system/filesystem/minimal' />
      </dependency>

      <exec_method
            type='method'
            name='start'
            exec='/lib/svc/method/identity-domain'
            timeout_seconds='60' />

      <template>
```

```
                  <common_name>
                        <loctext xml:lang='C'>
                              system identity (domainname)
                        </loctext>
                  </common_name>
                  <documentation>
                        <manpage title='domainname' section='1M'
                              manpath='/usr/share/man' />
                        <manpage title='defaultdomain' section='4'
                              manpath='/usr/share/man' />
                  </documentation>
            </template>

      </instance>

      <stability value='Unstable' />

</service>

</service_bundle>
```

## *10.2 SC portion of AI default.xml*

### 10.2.1 SC portion of AI default.xml for nodename

```
<service_bundle type="profile" name="default">
    <service name="system/identity" version="1" type="service">
        <instance name="node" enabled="true">

            <!-- The following property group is used at install
                 time to configure the nodename for the system -->

            <property_group name="config" type="application">
                <propval name="nodename" type="astring" value="unknown"/>
            </property_group>
        </instance>
    </service>
</service_bundle>
```