Using SMF for Appserver Processes as Services: A Design Document

TOC

1 Preliminary Introduction to SMF

Lot has been written about this novel way of looking at Solaris Platform Services. Refer to bigadmin, docs.sun.com, predictive self-healing features of Solaris for the details. Aligning Appserver with advances in the Solaris Platform makes (a lot of) sense.

Read the FAQ section as well.

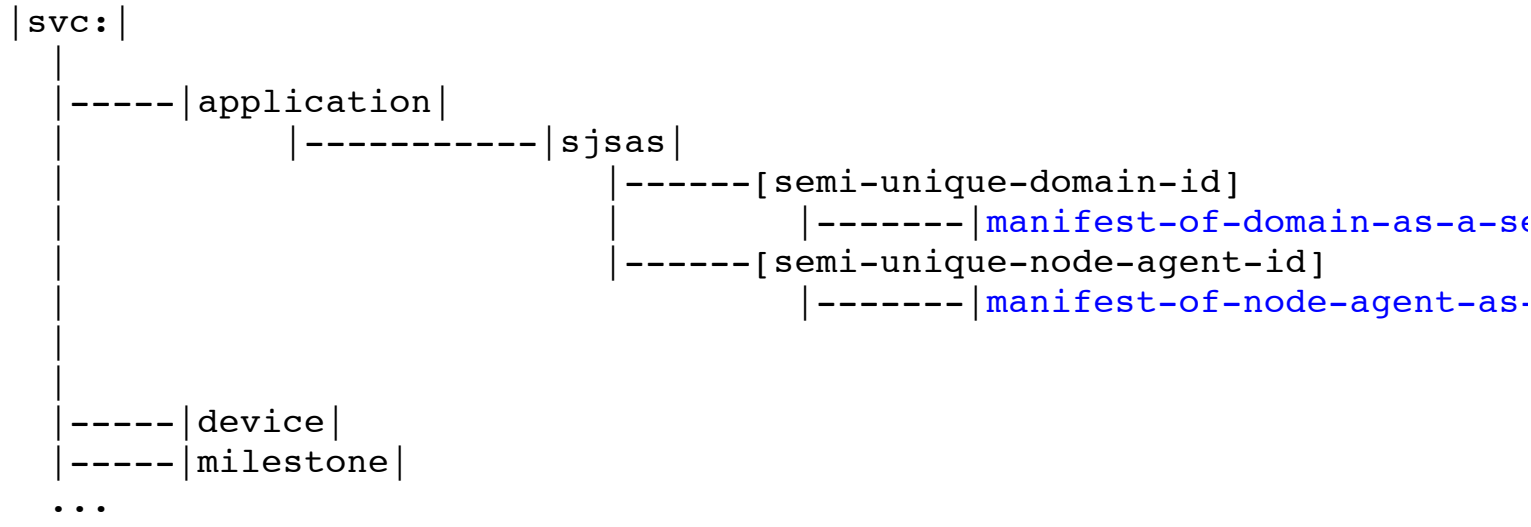
We are leveraging this feature to satisfy following requirements:

- To provide an automatic restart support for the Domain Administrative Server (DAS) of SJSAS. This enables minimal loss of service in the event of DAs failure.
- To provide an automatic restart support for the Node Agent (NA) of SJSAS. This enables minimal loss of service in the event of NA failure.
- To be able to restart the services where the service processes need to be owned by users other than the super-user. This is a feature that Appserver 7.0 had but was not carried forward in Appserver 8.x. The reintroduction of the so-called *run-as user* helps in cases where the appserver domains (and servers and clusters therein) are not owned by the super-user. It is believed that this is a significant value-add.

2 Appserver and Solaris 10: Service Configuration

Application Server software can be installed on a Solaris 10 system multiple times, such that different operating system users can use it independently. This independence is realized through an abstract boundary of an appserver administrative domain (domain, henceforth unless noted otherwise). Almost always, a domain comprises of a domain administrative server (DAs), one or more Node-Agents (NA) and a set of actual JEE (Java Enterprise Edition) Engines typically grouped in clusters that run user applications. An instance of DAs identifies a domain. One or more associated Node-Agent instances control the life-cycle of JEE Engines underneath. Thus, SMF is leveraged for both DAs and NA, whereas the NA controls the actual JEE Engines.

To adequately reflect the hierarchy, arranging the service manifests of various DAs and NA instances on the given Solaris System is in order. SMF guidelines recommend that we use the **application** namespace for this purpose. Following structure emerges:



DAs and NA names are unique for the system. Following is the algorithm that determines this id.

Id-of-the-domain = name-of-the-domain concatenated by *dot-representation* of absolute location that stores domain configuration.

Id-of-the-node-agent=name-of-the-node-agent concatenated by *dot-representation* of absolute location that stores node-agent configuration.

The dot-representation of the absolute location where a particular configuration resides is the string obtained by replacing the '/' character in the location by a '.'. Following are the examples of the unique-ids in certain cases:

Domain `domain1` placed in `root directory`	`domain1.`
Domain `salesdomain` placed in `/var/appserver/domains`	`salesdomain.var.appserver.domai`
Node Agent `node.agent.1` placed in `/var/appserver/nodeagents`	`node.agent.1.var.appserver.node`

It is important to note that the basic functionality of domain and node-agent (configuration) creation already guards against creating two entities with the same name in the same directory on the disk.

It should be noted that the above algorithm just preserves the dots that may be already present in the name of the domain or node-agent.

The manifest-file-templates are distributed as part of appserver installation. As part of CLI-command execution, following things happen sequentially:

- Template is token-replaced by appropriate values
- Concrete (token-replaced) service-manifest-file is copied to appropriate location under appropriate name (e.g. `/var/svc/manifest/application/sjsas/<unique-domain-id>/Domain-service.xml for domain, and /var/svc/manifest/application/sjsas/<unique-node-agent-id>/NodeAgent-service.xml`)
- The service-manifest-file is validated
- The service-manifest-file is imported as a service
- Clean-up in case service-creation fails

The templates used are:

- Domain-service-smf.xml.template - for appserver Domain
- NodeAgent-service-smf.xml.template - for appserver Node Agent

Following tokens are replaced:

ID	Token (a token appears as %%%XYZ%%% in the template)	Value
1	DATE_CREATED	The standard date when the manifest file was created (`c.f. java.util.Date`)
2	NAME	Name of the Domain or Node-Agent. (e.g. domain1, myagent)
3	LOCATION	Absolute location where the configuration of Domain or Node Agent resides
4	FQSN	Fully Qualified Service Name -- the unique id (a function of NAME and LOCATION) of the Domain or Node Agent
5	AS_ADMIN_PATH	Absolute location of command line interface, *asadmin*
		Absolute location of the (secure) file where the administrative user, administrative

6	PASSWORD_FILE_PATH	the administrative user, administrative password and appserver master password are stored in clear-text
7	TIMEOUT_SECONDS	Timeout in seconds ("0" implies infinite) that determines how long the boot sequence should wait before giving up starting/stopping this service
8	OS_USER	The *run-as user-id* who owns the configuration of the Domain or Node-Agent, this denotes the user who owns all the processes of this service

3 Goal for SJSAS 9.0

For SJSAS9.0, creation of a service alone is supported. No additional support to delete a service or list services. Creating an existing service should result in an error.

It should be noted that this facility is not to replace or hide the svcs, svcadm interface(s) available in Solaris 10. It only facilitates the creation of the service as it involves few steps including creation of a manifest file.

The user is supposed to *enable* the service when needed.

4 CLI Support

The CLI command used for this purpose is **create-service**. The syntax could be found [here].

5 Interfaces and Implementations

The interface and implementation Javadoc is here. The usage is shown clearly in the Javadoc.

6 Limitations

For SJSAS 9.0, only creation of service is possible. Users have to depend on svcs, svcadm, svccfg interfaces from Solaris 10 to actually enable the service.

TOC

$Author: kedar $
$Id: design.html,v 1.2 2005/09/16 17:50:11 kedar Exp $