JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

VDCF - Virtual Datacenter Control Framework

for the Solaris™ Operating System

VDCF Blueprint

Software Deployment

Version 1.1
August 2014

Copyright © 2005-2014 JomaSoft GmbH
All rights reserved.

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

Table of Contents

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

1 Introduction

This documentation describes the Software Deployment features of the Virtual Datacenter Control Framework (VDCF) for the Solaris Operating System.

1.1 Overview

Like most automated installation frameworks, VDCF also supports the deployment of Software during installation of a node, guest domain or vServer (Zone). VDCF additionally supports deployment on existing systems.

With VDCF Software Deployment the following types are supported:
- COMMAND (run a command)
- SCRIPT (run a script)
- PKG (install System V packages)
- FILE (copy a file to a target directory)

Additionally the following types are used to customize the Solaris system environment:
- SERVICES (enable or disable SMF services)
- DEFAULTROUTE (configure the defaultrouter)
- ROUTE (configure host and/or network routes)
- DNS (configure DNS)
- NTP (configure the NTP client)
- SCSI_VHCI (configure I/O multipathing for non-Sun symmetric storage devices)

The focus of this Blueprint are COMMAND,SCRIPT,PKG and FILE

1.2 Requirements

No special requirements. Software Deployment is part of VDCF since the Version 4.2

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

2 VDCF Software Deployment

With VDCF you can easily deploy Software and modify your systems to your needs.
In this Blueprint we will show you how to deploy a Solaris package, copy configuration files, run scripts and start commands during the installation and on existing systems.

2.1 VDCF Command "config"

This command is used to manage base configurations. Base configurations hold the configuration values. Every base configuration belongs to a type and has a name. The base configurations are reusable building blocks to define configurations for servers.

2.1.1 Create or extend a configuration (config -c add)

As the first step you add the configuration values to the VDCF repository. Every configuration requires a Type and a Name to identify. The Name must be unique within the Type.

```
config -c add      type=<config type>
                   name=<name>
                 [ comment=<comment> ]
                   <args ...>            depending on type
```

For Software Deployment the Types COMMAND, SCRIPT, PKG and FILE are available.

Depending on the Configuration Type you must provide several additional arguments.
Use the manpages to learn more about the required arguments:

```
$ config -H PKG

User Commands                                          config PKG(1)

NAME
     config PKG - PKG base configuration

SYNOPSIS
     config -c add type=PKG
     name=<configuration name>
     pkgs=<pkg[,pkg,pkg]>
     pkgdevice=<device>
     [ options=<pkgadd options> ]
     [ comment=<comment> ]
```

The configuration types used for Software Deployment (COMMAND, SCRIPT, PKG, FILE) support multiple configuration entries for a given name. If a configuration with the given 'name' exists, the new value is appended to the existing configuration. This allows you to create one configuration which then executes multiple steps. For example copies multiple files, executed multiple commands or installs multiple packages.

2.1.2 Modify a configuration (config -c modify)

Modifies an existing base configuration for the given config type and name. The base configuration must already exist and will be overwritten completely.

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

2.1.3 Type COMMAND

```
$ config -H COMMAND

User Commands                                        config COMMAND(1)

NAME
     config COMMAND - COMMAND base configuration

SYNOPSIS
     config -c add type=COMMAND
     name=<configuration name>
     command=<command with options>
     [ comment=<comment> ]
```

The following example creates a base configuration of the type COMMAND with name FSS. The command "/usr/sbin/dispadmin -d FSS" is defined to be executed on the target system:

```
$ config -c add type=COMMAND name=FSS command="/usr/sbin/dispadmin -d FSS"
Configuration COMMAND (/usr/sbin/dispadmin -d FSS) successfully added.
```

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

2.1.4 Type SCRIPT

```
$ config -H SCRIPT

User Commands                                        config SCRIPT(1)

NAME
     config SCRIPT - SCRIPT base configuration

SYNOPSIS
     config -c add type=SCRIPT
     name=<configuration name>
     script=<script>
     [ comment=<comment> ]
```

The script must be runnable standalone without arguments and copied into the VDCF script directory: `/var/opt/jomasoft/vdcf/config/script`

Additional subdirectories in the script directory are supported.

The following example creates a base configuration of the type SCRIPT with name 'stduser'.
The script must be located at `/var/opt/jomasoft/vdcf/config/script/add_std_users`.

```
$ config -c add type=SCRIPT name=stduser script=add_std_users
Configuration SCRIPT successfully added.
```

Another sample within a subdirectory

```
$ ls -l /var/opt/jomasoft/vdcf/config/script/general/system_summary
-rwxr-xr-x   1 root     root         349 Feb 19 15:04
/var/opt/jomasoft/vdcf/config/script/general/system_summary

$ config -c add type=SCRIPT name=sys-summary script=general/system_summary
Configuration SCRIPT successfully added.
```

VDCF provides the following environment variables to the scripts:

```
VDCF_MNGT_SERVER:    VDCF Management Server hostname
VDCF_SYSTEM_NAME:    System hostname
VDCF_SYSTEM_TYPE:    System type
                     (Values: GDOM, CDOM, GZONE, VSERVER)
VDCF_SYSTEM_COMMENT: System comment
VDCF_SYSTEM_BUILD:   Build name used to setup this system
VDCF_NODE_LOCATION:  System location
VDCF_CONFIG_GROUPS:  VDCF configuration groups assigned to this system
                     (Values: comma separated list of group names)
```

2.1.5 Type FILE

```
$ config -H FILE

User Commands                                          config FILE(1)

NAME
     config FILE - FILE base configuration

SYNOPSIS
     config -c add type=FILE
     name=<configuration name>
     source=<file>
     target=<directory or file>
     owner=<fileowner>
     mode=<filemode>
     [ comment=<comment> ]
```

The following example creates a base configuration of the type FILE with name 'test'. It will copy the file located at `/var/opt/jomasoft/vdcf/config/file/test.cfg` to `/etc/test.cfg` on the target system with the ownership root:other and file mode 640

```
$ config -c add  type=FILE name=test source=test.cfg target=/etc/test.cfg \
 owner=root:other mode=640
Configuration FILE successfully added.
```

2.1.6 Type PKG

```
$ config -H PKG

User Commands                                           config PKG(1)

NAME
     config PKG - PKG base configuration

SYNOPSIS
     config -c add type=PKG
     name=<configuration name>
     pkgs=<pkg[,pkg,pkg]>
     pkgdevice=<device>
     [ options=<pkgadd options> ]
     [ comment=<comment> ]
```

The following example creates a base configuration of the type PKG with name 'tsm553'.
The package name is TIVsmCapi and the pkg is stored at
`/var/opt/jomasoft/vdcf/config/pkg/ibm/tsm/TIVsmCapi.pkg`.

```
$ config -c add type=PKG name=tsm553 pkgs=TIVsmCapi pkgdevice=ibm/tsm/TIVsmCapi.pkg
Configuration PKG successfully added.
```

By default packages are installed with the "-G" option if the option argument is not used.

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

2.2 VDCF Command "serverconfig"

This command is used to manage server configurations. Serverconfig requires existing base configurations, which are defined by the config command. You can add (connect) base configurations to servers, servergroups or to default.
Server configurations are used when installing nodes, guest domains and vServers.

There are 3 types of server configurations: default, servergroup, server.

Server configurations of type default are used for all nodes, guest domains and vServers. A server (node or vServer) belongs to one or multiple servergroups.

Sections
For Software Deployment the order of execution is defined using a "section" number. The Server configuration is then used in the order, where Section 1 is used first, then the configurations of Section 2 and so on. Default is Section 1.

2.2.1 Create a server configuration (serverconfig -c add)

```
$ serverconfig -H add

User Commands                                serverconfig -c add(1)

NAME
     serverconfig -c add - Add a  base  configuration  to  server
     configuration

SYNOPSIS
     serverconfig -c add
     type=<config type>
     name=<name>
     [ server=<node or vserver> ]
     [ group=<group> ]
     [ section=<section> ]
     [ comment=<comment> ]
```

The following example will add the existing base configuration with type COMMAND and name FSS to the servergroup node:

```
$ serverconfig -c add type=COMMAND name=FSS group=node
Configuration successfully added to group node
```

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

2.2.2 Modify a server configuration (serverconfig -c modify)

```
$ serverconfig -H modify

User Commands                              serverconfig -c modify(1)

NAME
     serverconfig -c modify - Modify the section of a server con-
     figuration

SYNOPSIS
     serverconfig -c modify
     type=<config type>
     name=<name>
     section=<section>
     [ default | group=<group> | server=<node or vserver> ]
     [ comment=<comment> ]
```

The following example will modify the existing server configuration with type COMMAND and name FSS, assigned to group node and change the section to 3.

```
$ serverconfig -c modify type=COMMAND name=FSS group=node section=3
Section changed for server config with type <COMMAND> name <FSS> of group <node>.
```

2.2.3 Serverconfig supergroup

If your configuration has many groups, you may use the serverconfig supergroup. A supergroup can bundle one to n serverconfig groups. Using supergroups you only need to assign one group to your system instead of many groups.

Create a supergroup:
`serverconfig -c create_group supergroup=<group> subgroups=<group,group,...>`

The following example creates a supergroup called 'node' and includes the serverconfig groups 'node_base' and 'oracle_tools':

```
$ serverconfig -c create_group supergroup=node subgroups=node_base,oracle_tools
Configuration group <node> successfully created
```

List the available Groups/Supergroups with: `serverconfig -c list groups`

```
$ serverconfig -c list groups

             Group  Subgroups                  Comment
               MSP
              node  node_base,oracle_tools     Config used for all Nodes
         node_base
            oracle
      oracle_nodes  node_base,oracle
      oracle_tools
           virtual                             Config used for all vServer
```

3 Deployment sample

Following an example to

- Deploy a Solaris package
- Update a configuration file from the package
- Execute a command from the package
- Execute a script during installation

3.1 Sample - Base configuration

The required package, files and scripts are already in the VDCF config directory:
`/var/opt/jomasoft/vdcf/config`

Command

```
$ config -c add type=COMMAND name=check_change \
 command='/opt/MSP/serverdocu/bin/check_change.ksh pre'
Configuration COMMAND (/opt/MSP/serverdocu/bin/check_change.ksh pre) successfully
added.
```

Script

```
$ cd /var/opt/jomasoft/vdcf/config/script

$ cd general

$ ls -l system_summary
-rwxr-xr-x   1 root     root         349 Feb 19 15:04 system_summary

$ more system_summary
#!/bin/ksh

echo "System Summary"
echo "--------------"
echo " "
echo "VDCF MNGT Server: $VDCF_MNGT_SERVER"
echo "The System $VDCF_SYSTEM_NAME ($VDCF_SYSTEM_TYPE)"
echo "has been installed date: $(date) "
echo " "
echo "OS System Name:   $(uname -a)”
echo " "
echo "VDCF Build:        $VDCF_SYSTEM_BUILD"
echo "OS Release:”
cat /etc/release
echo " "


$ config -c add type=SCRIPT name=sys-summary script=general/system_summary
Configuration SCRIPT successfully added.
```

Package

```
$ cd /var/opt/jomasoft/vdcf/config/pkg

$ ls -l sysdoc/MSPserdoc*

-rw-r--r--   1 root     other    4996096 Sep  6 15:53 sysdoc/MSPserdoc.pkg
-rw-r--r--   1 root     other         38 Sep  6 15:58 sysdoc/MSPserdoc.rsp
```

If a <packagename>.rsp File exists, the file will be used as Response File for the Solaris pkgadd command.

```
$ config -c add type=PKG name=sysdoc pkgs=MSPserdoc pkgdevice=sysdoc/MSPserdoc.pkg
Configuration PKG successfully added.
```

File

```
$ cd /var/opt/jomasoft/vdcf/config/file

$ ls -l sysdoc/customer_inv

-rw-r--r--   1 root     root        1383 Sep  6 16:20 sysdoc/customer_inv

$ config -c add type=FILE name=MSP_Inv source=sysdoc/customer_inv \
 target=/opt/MSP/serverdocu/data/customer_inv owner=root:root mode=0644
Configuration FILE successfully added.
```

This configuration File is now registered to be copied to the defined target path.

3.2 Sample - Server configuration

We define a serverconfig group and add the base configurations. We can individually assign the group to systems using the following command:

Nodes/Guest Domains:
```
nodecfg -c modify addgroup=<config group list> name=<node name>
```

vServer:
```
vserver -c modify addgroup=<config group list> name=<vserver name>
```

Command

```
$ serverconfig -c add type=COMMAND name=check_change group=MSP section=3
Configuration successfully added to group MSP
```

The command will be executed on the system in section 3. We only have a dependency to the package installation. The command and the file copy are independent of each other.
The first add to a group will create the group.

Script

```
$ serverconfig -c add type=SCRIPT name=sys-summary group=node section=5
Configuration successfully added to group node
```

The script will be executed on every physical node during installation in section 5.

Note: Nodes are assigned to the group node by default, at the time they are created using nodecfg -c add.

Package

```
$ serverconfig -c add type=PKG name=sysdoc group=MSP section=2
Configuration successfully added to group MSP
```

The package will be added in section 2, to make sure it is installed, the command and file configuration executed later.

File

```
$ serverconfig -c add type=FILE name=MSP_Inv group=MSP section=3
Configuration successfully added to group MSP
```

The configuration file will be copied to the system in section 3. The package will already be installed at this time.

show group

Use the command below to list the assigned base configurations of a group

```
$ serverconfig -c show group=MSP

section: 2  type: PKG          name: sysdoc        group: MSP
  value: MSPserdoc@sysdoc/MSPserdoc.pkg

section: 3  type: FILE         name: MSP_Inv       group: MSP
  value: sysdoc/customer_inv,/opt/MSP/serverdocu/data/customer_inv,root:root,0644

section: 3  type: COMMAND      name: check_change  group: MSP
  value: /opt/MSP/serverdocu/bin/check_change.ksh pre
```

show server

Now we add this group to our server.

```
$ nodecfg -c modify addgroup=MSP name=g0085
node modified successfully.
```

Use the command below to list the assigned base configurations of a server

```
$ serverconfig -c show server=g0085

section: 1  type: COMMAND      name: FSS           group: node
  value: /usr/sbin/dispadmin -d FSS

section: 5  type: SCRIPT       name: sys-summary   group: node
  value: general/system_summary

section: 2  type: PKG          name: sysdoc        group: MSP
  value: MSPserdoc@sysdoc/MSPserdoc.pkg

section: 3  type: FILE         name: MSP_Inv       group: MSP
  value: sysdoc/customer_inv,/opt/MSP/serverdocu/data/customer_inv,root:root,0644

section: 3  type: COMMAND      name: check_change  group: MSP
  value: /opt/MSP/serverdocu/bin/check_change.ksh pre
```

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

3.3 Sample - Node installation / console output

The following output will appear if a node configured with the server configuration defined above is installed using VDCF.

```
executing script: /tmp/customize_system      << Start of Node configuration
System Customize started ...
Executing Section: 1 ...
Section: 1 done
Executing Section: 2 ...                     << Section 2 with Solaris package
Installing Package MSPserdoc (2.2.0(patchdiag.xref06.08.2011)) ...
Exit-Code: 0  (Successful completion)
Check Logfile /var/tmp/vdcf/install_config.log for details about the installed
Packages
Section: 2 done
Executing Section: 3 ...                     << Section 3 with file and command
Copied file sysdoc/customer_inv to /opt/MSP/serverdocu/data/customer_inv
Executing command: /opt/MSP/serverdocu/bin/check_change.ksh pre
collecting data .... (please be patient)
Exit-Code: 0
Executing Section: 4 ...
Section: 4 done
Executing Section: 5 ...
Executing Script: general/system_summary     << Section 5 with script
System Summary
--------------

VDCF MNGT Server: g0069
The System g0085 (GDOM)
has been installed date: Wednesday, February 20, 2013 05:22:43 PM CET
OS System Name:   SunOS g0085 5.11 11.0 sun4v sparc sun4v
VDCF Build:        s11.0-s
OS Release:
                        Oracle Solaris 11 11/11 SPARC
  Copyright (c) 1983, 2011, Oracle and/or its affiliates.  All rights reserved.
                          Assembled 18 October 2011

Exit-Code: 0
Section: 5 done                              << Node configuration finished
```

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

4 Software Re-Deployment (serverconfig -c exec type=PKG)

In the following scenario a new Version of the MSPserdoc package is available. The new Version needs to be copied to `/var/opt/jomasoft/vdcf/config/pkg/sysdoc/MSPserdoc.pkg`

New installed Nodes or vServers would now get the new Version of the package. To update the existing package on the already installed Nodes and vServers, we can use the VDCF Re-Deployment feature.

If you want to replace an existing package with the package stored on the VDCF Management Server the following variable must be set in the customize.cfg: **`export CONFIG_EXEC_PKG_REPLACE=TRUE`**

Display the installed versions using the vpkgadm command:

```
$ vpkgadm -c show_server name=MSPserdoc

Package: MSPserdoc - Server-Documentation for Sun Solaris Systems
PKG-ID : MSPserdoc@2.2.11_(patchdiag.xref_23.08.2012)@S000020120824211215
Version: 2.2.11 (patchdiag.xref 23.08.2012)@S000020120824211215 is installed on:
   Name  Type     Comment
  s0006  Node     VDCF

Package: MSPserdoc - Server-Documentation for Sun Solaris Systems
PKG-ID : MSPserdoc@2.2.11b_(patchdiag.xref_18.07.2012)@S000020120718221213
Version: 2.2.11b (patchdiag.xref 18.07.2012)@S000020120718221213 is installed on:
   Name  Type     Comment
  s0005  Node     Demo Node
  s0011  Node     ZFS boot / IPMP
  v0102  vServer  imported vServer
  v0109  vServer  Solaris 8
...
```

The following example removes the existing MSPserdoc package on the target node and installs the new version from VDCF Management Server. VDCF does install the package from the VDCF Management Server independent of the package version.

```
$ serverconfig -c exec type=PKG name=sysdoc server=s0006

Executing on Node s0006 (VDCF)
Removing Package MSPserdoc (2.2.11 (patchdiag.xref 23.08.2012)) ...
Installing Package MSPserdoc (2.2.11b (patchdiag.xref 18.07.2012)) ...
Exit-code: 0  (Successful completion)
execution successful
```

As target the following options are available:

`server=<comma sep list>`	`server=node1,node2,vserver4`

`servergroup=<config group>`	`servergroup=MSP`
	`serverconfig -c show_members group=MSP`
	(lists the assigned node,vserver)

`serverfile=<abs. path to file>`	`# cat /tmp/srv_file`
	`node1`
	`node2`
	`vserver4`

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

5 Executing Commands (serverconfig -c exec command)

In the following scenario we need to verify the last logins of a user to the system. This information is requested for every vServer.

With the serverconfig exec feature we get this information centrally on the VDCF Management Server.

```
$ serverconfig -c exec command="last | grep vdcfexec" servergroup=virtual

Executing on vServer v0126 (vserver v0126) on Node g0069

Executing command: last | grep vdcfexec
vdcfexec  sshd         s0003-mngt.jomas Fri Apr 27 04:01 - 04:01  (00:00)
vdcfexec  sshd         s0003-mngt.jomas Thu Apr 26 04:01 - 04:01  (00:00)
vdcfexec  sshd         192.168.20.69    Wed Apr 25 10:01 - 10:01  (00:00)
vdcfexec  sshd         192.168.20.69    Wed Apr 25 10:01 - 10:01  (00:00)
Exit-Code: 0


Executing on vServer v0128 (vserver v0128) on Node s0024

Executing command: last | grep vdcfexec
vdcfexec  sshd         192.168.0.2      Tue Jan  3 16:53 - 16:54  (00:00)
vdcfexec  sshd         192.168.0.2      Wed Oct  5 09:43 - 09:43  (00:00)
vdcfexec  sshd         192.168.0.2      Wed Oct  5 09:43 - 09:43  (00:00)
vdcfexec  sshd         192.168.0.2      Wed Oct  5 09:43 - 09:43  (00:00)
vdcfexec  sshd         192.168.0.2      Wed Oct  5 09:43 - 09:43  (00:00)
vdcfexec  sshd         192.168.0.2      Wed Oct  5 09:43 - 09:43  (00:00)
vdcfexec  sshd         192.168.0.2      Wed Oct  5 09:43 - 09:43  (00:00)
Exit-Code: 0
```

With the command argument we specify the command to be run on the clients. By defining the servergroup 'virtual' the command is executed on every vServer.

If you need to run more complex commands, we recommend to create a VDCF configuration of Type SCRIPT and execute the script on the required systems.

JomaSoft GmbH
Falkensteinstr. 54a
9000 St. Gallen
Switzerland

6 Executing Scripts (serverconfig -c exec type=SCRIPT)

In the following scenario the login shell for local users need to be updated from Korn-Shell to Bash.

we created a new script:

```
# pwd
/var/opt/jomasoft/vdcf/config/script

# more update_usershell
#!/bin/bash

for user in marco mech marcel andy ; do
  if getent passwd $user >/dev/null; then
    if ! /usr/sbin/usermod -s /bin/bash $user; then
      echo "WARN: could not update shell for user: $user"
    fi
    getent passwd $user
  fi
done
```

and created a base configuration:

```
$ config -c add type=SCRIPT name=update_usershell script=update_usershell
Configuration SCRIPT successfully added.
```

Now we are able to update the required systems:

```
$ serverconfig -c exec type=SCRIPT name=update_usershell server=g0085,g0077,...

Executing on Node g0085 (Guest domain g0085)
Executing script: update_usershell
marco:x:202:10::/home/marco:/bin/bash
mech:x:150:10::/home/mech:/bin/bash
marcel:x:200:10::/home/marcel:/bin/bash
andy:x:201:10::/home/andy:/bin/bash
Exit-Code: 0

Executing on Node g0077 (Guest domain g0077)
Executing script: update_usershell
marco:x:202:10::/home/marco:/bin/bash
mech:x:150:10::/home/mech:/bin/bash
marcel:x:200:10::/home/marcel:/bin/bash
andy:x:201:10::/home/andy:/bin/bash
Exit-Code: 0

...

execution successful
```