Migrating Packages from Legacy to Modular Style

Introduction

Abstract

Modular packages provide a new interface for configuring Serviceguard packages. They modularize both package configuration files and the control script, allowing easier integration and validation of user-defined functions and parameters. This document describes how to migrate current ("legacy") packages to the new modular package configuration. For more information, see the latest edition of the *Managing Serviceguard* manual at http://docs.hp.com -> High Availability -> Serviceguard (or Serviceguard for Linux).

Intended Audience

This document is intended for Serviceguard cluster Administrators. It assumes the reader is familiar with both the legacy package and the new modular package configuration, and with the basics of HP-UX or Linux shell scripting.

Related Documents

Managing Serviceguard (fourteenth edition or later), chapters 3, 4, 6, and 7.
Man page for *cmmigratepkg*.

Terms and Definitions

Modular Package	Single package configuration file, introduced in Serviceguard A.11.18. The package configuration information is included in only the package configuration (ASCII) file, whereas in pre-11.18 packages configuration information is in both the package ASCII file and the package control script.
Legacy Package	Package Configuration pre-11.18
Attribute	A configurable parameter in the package configuration file
Module	A building block that includes a specific set of attributes for package configuration.
ADF Module File	Attribute Definition File that defines a module. The ADF Modules are used to build a modular package's ASCII configuaration file. They define the attributes which the package ASCII file includes.
PEV	Package Environment Variable. A user-definable variable that can be passed to external scripts.
External Script	User-created script invoked by Serviceguard at validation/start/stop of a package.
Toolkits	The toolkits refers to the following products: Enterprise Cluster Master Toolkit, Oracle Toolkit for SG/Linux, Serviceguard Extensions for SAP/R3, Metroclusters and ContinentalClusters

Customer Defined Area of control script	The area where a user can add code in the legacy package control script, defined by the comments "#START CUSTOMER DEFINED FUNCTIONS" and "#END CUSTOMER DEFINED FUNCTIONS".

Modular Packages

Serviceguard A.11.18 introduces a new style of package which differs from those used in earlier releases.

The benefits of a modular package are:

- Simplified Package Configuration
    - All configuration data for the package is now in the package configuration file. Previously, configuration data was included in both the configuration file and in the package control script.
- Modularized Approach to packages
    - All package parameters are now configured in one place, the package configuration file. You no longer need to create and distribute a separate package control file.
    - External scripts offer an improved means of application integration. These replace the Customer Defined Functions in the legacy package control script. You *do* need to distribute these scripts to all the nodes that can run the package.
    - The modular approach allows you to build a package from building blocks containing only the functions needed by this package.
    - Packages are built from a set of modules that define only the specific functionality that each package needs; paramters that are not needed are not included.
    - There are well-defined entry points for user-defined scripts
    - Software partnerscan easily plug in custom modules for their products
- Support Benefit
    - Modular packages can be enhanced in a patch without requiring re-integration. Enhancements to the package scripts do not require an update to the package configuration.
- Package Environment Variables allow you to add variables to the package configuration file. These can be passed to external scripts that you create, maintaining the model of separating configuration data and control scripts.

Note: Serviceguard A.11.18 continues to support legacy as well as modular packages.

Expected Usage of Modular Packages

Use of modular packages is preferred, keeping in mind the guidelines that follow.

New Serviceguard Installations

For new Serviceguard installations, modular packages are the preferred style. In this case, you can immediately benefit from the new 11.18 features. See chapter 6 of *Managing Serviceguard*. In subsequent Serviceguard releases, new package features may be available only in modular packages.

Existing Serviceguard Installations

In existing installations, you may prefer not to redefine existing packages because they are well-tested and operating as desired. You can continue to use and maintain your legacy packages on Serviceguard A.11.18. See chapter 7 of *Managing Serviceguard*.

Whether or not you convert existing packages, HP recommends that you create new packages as modular packages to take advantage of the new features. Modular Packages and legacy packages can coexist on the same cluster.

Package Environment Variables (PEV)

You can add variables to the package using package environment variables (PEVs). PEVs allow attributes to be added to the Modular Package configuration file. cmviewcl's –f line displays the packages' PEVs and their value. PEVs are passed as environment variables to any external scripts that you add to the package. Define these attributes in the package configuration file using the prefix "PEV_"(uppercase with an underscore as shown).

For example if a legacy package control script has the variable APP_DIRECTORY, the corresponding variable in the package configuration file would be PEV_APP_DIRECTORY:

PEV_APP_DIRECTORY /var/opt/app

You can add your own custom scripts to the package definition using two well-defined entry points, external_script and external_pre_script. Any PEVs you define are passed into the script when it is executed.

In the example above, the environment variable PEV_APP_DIRECTORY with the value of "/var/opt/app" would be passed to the external script. All environment variables are passed to the scripts in uppercase.

For more information about user-created scripts and PEVs, see chapters 4 and 6 of the *Managing Serviceguard* manual.

Package Migration

Overview

The following sections describe the migration of different types of package, and provide instructions for migrating legacy packages to modular packages.

The high-level steps are:

1. Determine the profile of the package from the tables that follow.

2. Determine from the package profile if the package can be migrated using the *cmmigratepkg* command.
    a. If it *cannot* be migrated using *cmmigratepkg,* follow instructions in the section on "Manual steps for migrating a package".
    b. If it *can* be migrated using *cmmigratepkg:*
        i. Check the control script for changed or obsolete functions.
        ii. Select the options for the *cmmigratepkg* migration tool and migrate the package following the examples for the profile that matches this package.

Package Profiles

The first step is to determine the profile of legacy package you want to migrate. Use the table that follows.

The following describes the package profiles which can be migrated using the automated tool.

Profile	**Description**
Simple Package	A failover package that does not have any additional code in the customer-defined user area of the control script, and whose package control script does not have any non-Serviceguard variables defined or used.
Package with Customer Defined Script	A failover package whose control script's customer-defined area has code which invokes another script or includes functions that are not a part of the Serviceguard control script template.
Package With Control Script with variables defined	A failover package whose package control script has user-defined environment variables that are not Serviceguard parameters.
Package with Dependencies	A package that depends on another package. The package can be converted to a modular package and the package which it depends upon can remain a legacy package.

The following packages must be manually migrated.

Profile	**Description**
Package with non-Serviceguard-generated Control Script	A package whose control script was not generated using the cmmakepkg –s command. (The migration tool supports all Serviceguard-generated versions created after 11.09).
Multi-Node and System Multi-Node Packages	A package that runs on more than one node at the same time.

	These types of package can be automatically migrated only if the control script is the same on every node that the package runs on. Otherwise the package should be manually migrated.
Packages created on versions of Serviceguard older than A.11.09	The migration tool cannot be used on any packages that were created on versions older than A.11.09. These packages were created using an older template for the control script. If the package control script's version number is older than A.11.09, the migration tool will output a message to that effect, and exit.

The following packages cannot be migrated.

Profile	**Description**
CFS Package	Veritas Cluster File System from Symantec. No CFS-defined package should be migrated, including all package named "SG-CFS-*".
CVM 3.5 Packages	A package used by the Veritas Cluster Volume Manager, version 3.5. No CVM 3.5 package should be migrated, including the system nulti-node package VxVM-CVM-pkg.
Toolkit Packages for (Oracle, Apache, Samba, Tomcat, HA NFS, SGeSap, SGeRac 10g RAC, Linux, HPVM toolkits) DTS (Plug-ins for XPCA, SRDF, and EVA CA, used in Metrocluster and Continentalclusterconfigurations)	A package generated by an HP toolkit. Do not attempt to migrate such packages. Only legacy packages are supported within the toolkits in Serviceguard A.11.18. Serviceguard. The next release of Serviceguard will support modular packages within the toolkits.

Migration Utility

The *cmmigratepkg* command automates the migration of a legacy package to a modular package. It creates a package configuration (ASCII) file for a modular package with the legacy package's information. The syntax is as follows:

cmmigratepkg –p <package_name> [-x <external_script_name>] [-e] –o <output_file>

The package must be a configured legacy package. It can be on-line or off-line. *cmmigratepkg* reads the the package's control script, so the control script must be located on the node where the command is run. *cmmigratepkg* can migrate packages created by Serviceguard versions A.11.09 through A.11.18.

The control script's customer-defined area is any code between "#START CUSTOMER DEFINED FUNCTIONS" and "#END CUSTOMER DEFINED FUNCTIONS". If the customer-defined area has

code that needs to be migrated, *cmmigratepkg* can create an external script by inserting the shell code from the control script into the external_script file. The external script is generated from the external template found in $SGCONF/examples/external_script.template. (See chapter 4 of the *Managing Serviceguard* manual for more information about the template.)

Non-Serviceguard variables defined in the legacy package control script can be converted to PEVs. *cmmigratepkg* displays informational messages identifying variables which it cannot convert because they are non-Serviceguard variables. If these non-Serviceguard variables are defined and used in the customer-defined area, there is no need to convert them to PEVs. If they are defined in another location in the control script, and the values should be part of the package, they can be converted to PEVs using the *–e* option. These PEVs are defined in the new package configuration file.

If non-Serviceguard functions are defined in the legacy package control script, *cmmigratepkg* lists them to STDOUT. These functions are not converted. You must determine if the functions are called and how you want to migrate them to the modular package. You can then put them into the new external script file.

Input Parameters

-p *package_name*

The name of an existing legacy package. Used to obtain the current configuration information.

-x *external_script_name*

The name of the external-script file. *cmmigratepkg* reads the user-defined functions for start and halt and creates an external script from them. The external_script_name is the full pathname of the target file.

-o *outputfile*

The name of the taget file that will contain the configuration for the new modular package.

-e

Create PEVs from parameters found in the package control script. *cmmigratepkg* adds the prefix "PEV_" to the parameter name and writes the resulting name to *outputfile*.

Checklist for Migrating a Legacy Package

Check legacy packages for the following to determine how to migrate the package:

1. Package parameters which are not supported in 11.18
   - See the Unsupported Package Parameters table below.
2. Any of the following in the package control script:
   - Non-Serviceguard parameters:
     i. If any non-Serviceguard parameters are defined, note the parameter names. Look at the Serviceguard Control Script Parameters tables below. *cmmigratepkg* lists any parameters which are non-Serviceguard Parameters. Determine where the non-Serviceguard parameters are defined in the control script:

- If the parameters are defined between the "#START OF CUSTOMER DEFINED" and "#END OF CUSTOMER DEFINED", *cmmigratepkg* can copy the code to the new external script if you use the *–x* option to generate an external script.
- If the parameters are defined elsewhere in the script, they can be converted to PEVs.
- User-created scripts must be modified to use the new environment variable names.

- Non-Serviceguard functions:
  i. Determine if there are any non-Serviceguard functions in the package control script. If the package control script was generated using *cmmakepkg –s* and no additional functions were added, then there are no non-Serviceguard functions. Otherwise, check the Unsupported Control Script Functions and Replaced Control Script Functions tables below. Check whether any of these functions are in the package's control script. Run *cmmigratepkg* and see if the command lists any non-Serviceguard functions.
  ii. If the functions are defined between the "#START OF CUSTOMER DEFINED FUNCTIONS" and "#END of CUSTOMER DEFINED FUNCTIONS", *cmmigratepkg* can copy them to an external script if you use the *–x* option.
  iii. If the functions are not in the customer-defined area of the control script, and are listed under Unsupported Control Script Functions below, then you must manually edit the external script.
  iv. If the functions are not in the Customer defined area and are listed in the Replaced Control Script Function Table below, then you must edit the external script to use the new names.
  v. If the functions are not covered by any of cases ii-iv above, then you must manually add them to the external script.
- Modifications to Serviceguard functions
  i. Determine if any modifications have been made to any Serviceguard functions defined in the Serviceguard package control script template. Someone well acquainted with the code in the package control script must determine this; it entails reviewing the code and manually comparing the Serviceguard template with this package's control script.
  ii. If there are modifications to any of the Serviceguard package control script functions, and they must be included in the new package, then you can not migrate the package.

3. Make sure the subnet and ip_address parameters defined in the legacy package's control script are defined in the cluster configuration file as STATIONARY_IP addresses for the nodes in the cluster. If the package includes a subnet and ip_address which are not configured as a STATIONARY_IP address in the cluster configuration file, add the subnet as a monitored subnet by adding the address as a STATIONARY_IP to the cluster configuration file. Then apply the new cluster configuration. (*cmmigratepkg* may complete successfully if you fail to do this, but the resulting package will not run.) See chapter 4 of the *Managing Serviceguard* manual for more information about the cluster configuration file, and chapter 5 for information on applying the configuration.

The following are the Serviceguard parameters in the legacy control script.

Serviceguard Control Script Parameters

Parameter Name
CONCURRENT_FSCK_OPERATIONS
CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS
CONCURRENT_VGCHANGE_OPERATIONS

CVM_ACTIVATION_CMD
CVM_DG
DEACTIVATION_RETRY_COUNT
DEFERRED_RESOURCE_NAME
DTC_NAME
LV
FS
FS_MOUNT_OPT
FS_UMOUNT_OPT
FS_FSCK_OPT
FS_TYPE
FS_MOUNT_RETRY_COUNT
FS_UMOUNT_COUNT
IP
SUBNET
SERVICE_NAME
SERVICE_CMD
SERVICE_RESTART
VG
VGCHANGE
VXVM_DG
VXVOL
KILL_PROCESSES_ACCESSING_RAW_DEVICES

The following legacy package attributes are not supported by modular packages.

Unsupported Package Parameters

Package Attribute	**Description**
STORAGE_GROUP	If STORAGE_GROUP was defined for the legacy package, then the corresponding modular package must declare a dependency on the CVM System Multi-Node package.
MD, RAIDTAB, RAIDSTART, RAIDSTOP, DTC_NAME, DATA_REP	These parameters are obsolete in A.11.18. *cmmigratepkg* gives a warning message if it finds these parameters in the control script. If you need to use them, consult Linux XDC Toolkit support.

Serviceguard Control Script Supported Functions

activate_disk_group
activate_volume_group
add_ip_address
check_and_mount
check_dg
check_vxvm_vol_available
deactivate_disk_group
deactivate_volume_group

disown_dtc
freeup_busy_mountpoint_and_mount_fs
get_ownership_dtc
halt_services
remove_ip_address
retry_print
start_resources
start_services
umount_fs
verify_ha_nfs
verify_physical_data_replication
wait_for_cvm_dg_vols_enabled
stop_resources
ps_tree
show_users
disown_dtc
dg_fuser
test_return
activation_check
check_gfs
lvm_sanity_check
vg_tag
verify_evfs
get_md
activate_md
deactivate_md
wait_for_diskgroup_enable
deactivate_dg
deactivate_dg_with_retries
verify_ha_nfs
verify_ha_server
check_gfs
customer_defined_halt_cmds
customer_defined_run_cmds

If the control script's customer-defined section calls any of the following functions, the external script needs to be edited to update the names as follows.

Replaced Control Script Functions Table

Function	Replacement
ps_tree	sg_ps_tree
show_users	sg_show_users

Unsupported Control Script Functions Table

Disown_dtc
get_ownership_dtc
ha_nfs_file_locks
Disown_dtc

Migrating a Simple Package

If a package does not have a customer defined user script and does not include any of its own environment variables in the package control script, follow these broad steps (see the examples that follows for details):

1. Run the *cmmigratepkg* command and generate a new modular package configuration file.
2. Run *cmcheckconf* with the new modular package configuration file.
3. Halt the package
4. Run *cmapplyconf* with the new modular package configuration file.
5. Run the package

The following example migrates the simple package *pkg_simple*.

Examine the package using cmviewcl.

$cmviewcl –v –l line –p

```
sandy:/etc/cmcluster/pkg/pkg-simple>cmviewcl -v -f line -p pkg-
simple
name=pkg-simple
type=failover
status=down
state=halted
highly_available=no
summary=critical
autorun=disabled
owner=unowned
id=41475
initial_autorun=enabled
failover_policy=configured_node
failback_policy=manual
local_lan_failover_allowed=enabled
failfast=disabled
run_script=/etc/cmcluster/pkg/pkg-simple/pkg-simple.control
run_script_timeout=320
halt_script=/etc/cmcluster/pkg/pkg-simple/pkg-simple.control
halt_script_timeout=320
priority=no_priority
successor_halt_timeout=no_timeout
```

```
script_log_file=/etc/cmcluster/pkg/pkg-simple/pkg-simple.log
node:sandy|name=sandy
node:sandy|status=down
node:sandy|switching=enabled
node:sandy|last_run_time=0
node:sandy|last_halt_time=0
node:sandy|available=yes
node:sandy|type=Primary
node:sandy|order=1
node:krabs|name=krabs
node:krabs|status=down
node:krabs|switching=enabled
node:krabs|last_run_time=0
node:krabs|last_halt_time=0
node:krabs|available=yes
node:krabs|type=Alternate
node:krabs|order=2
subnet:192.42.2.0|name=192.42.2.0
subnet:192.42.2.0|node:krabs|status=up
subnet:192.42.2.0|node:sandy|status=up
subnet:fec0:0:0:2a02::/64|name=fec0:0:0:2a02::/64
subnet:fec0:0:0:2a02::/64|node:krabs|status=up
subnet:fec0:0:0:2a02::/64|node:sandy|status=up
service:pkg-simple_srv_1|name=pkg-simple_srv_1
service:pkg-simple_srv_1|id=1
service:pkg-simple_srv_1|failfast=disabled
service:pkg-simple_srv_1|halt_timeout=5
service:pkg-simple_srv_1|node:krabs|status=down
service:pkg-simple_srv_1|node:krabs|restart_limit=unknown
service:pkg-simple_srv_1|node:krabs|restart_count=0
service:pkg-simple_srv_1|node:sandy|status=down
service:pkg-simple_srv_1|node:sandy|restart_limit=unknown
service:pkg-simple_srv_1|node:sandy|restart_count=0
```

Go to the directory where the package ascii and control scripts are stored. Run *cmmigratepkg* on the package.

```
$cd /etc/cmcluster/pkg/pkg-simple
$cmmigrate –p pkg-simple –o pkg-simple.conf
```

Examine the output.

```
$cat pkg-simple.conf
# Package generated by Migration Program
package_name                    pkg-simple
module_name                     sg/basic
module_version                  1
module_name                     sg/failover
module_version                  1
module_name                     sg/priority
module_version                  1
module_name                     sg/dependency
module_version                  1
module_name                     sg/monitor_subnet
module_version                  1
module_name                     sg/package_ip
module_version                  1
```

```
module_name                     sg/service
module_version                  1
module_name                     sg/volume_group
module_version                  1
module_name                     sg/filesystem
module_version                  1
module_name                     sg/pev
module_version                  1
module_name                     sg/external_pre
module_name                     sg/external_pre
module_version                  1
module_name                     sg/external
module_version                  1
module_name                     sg/acp
module_version                  1
package_type    FAILOVER
auto_run        YES
node_fail_fast_enabled  NO
run_script_timeout      320
halt_script_timeout     320
script_log_file /etc/cmcluster/pkg/pkg-simple/pkg-simple.log
failover_policy CONFIGURED_NODE
failback_policy MANUAL
local_lan_failover_allowed      YES
node_name                       sandy
node_name                       krabs
operation_sequence
$SGCONF/scripts/sg/external_pre.sh
operation_sequence
$SGCONF/scripts/sg/volume_group.sh
operation_sequence
$SGCONF/scripts/sg/filesystem.sh
operation_sequence
$SGCONF/scripts/sg/package_ip.sh
operation_sequence
$SGCONF/scripts/sg/external.sh
operation_sequence
$SGCONF/scripts/sg/service.sh
service_name    pkg-simple_srv_1
service_cmd     "/usr/bin/X11/xclock -display 15.1.194.102"
service_fail_fast_enabled       NO
service_halt_timeout    5

ip_subnet       192.42.2.0
ip_address      "192.42.2.18"
ip_subnet       fec0:0:0:2a02::/64
ip_address      3ffe:1000:0:2a02::11/64
vxvm_dg                 dg_sandy_dd0
fs_name                 /dev/vx/dsk/dg_sandy_dd0/lvol1
fs_directory
/var/opt/sgtest/tmp/mnt/dev/vx/dsk/dg_sandy_dd0/
lvol1
fs_type                 ""
fs_mount_opt                    ""
fs_umount_opt                   ""
fs_fsck_opt                     ""
fs_name                 /dev/vx/dsk/dg_sandy_dd0/lvol1
```

```
      fs_directory
      /var/opt/sgtest/tmp/mnt/dev/vx/dsk/dg_sandy_dd0/
      lvol1
      fs_type                 ""
      fs_mount_opt                    ""
      fs_umount_opt                   ""
      fs_fsck_opt                     ""
      vgchange                        "vgchange -a e"
      cvm_activation_cmd                      "vxdg -g \$DiskGroup set
      activation=excl
      usivewrite"
      deactivation_retry_count                        2
      kill_processes_accessing_raw_devices                    NO
      vxvol                   "vxvol -g \$DiskGroup startall"
      fs_umount_retry_count                   1
      fs_mount_retry_count                    0
      concurrent_vgchange_operations                  1
      concurrent_fsck_operations                      1
      concurrent_mount_and_umount_operations                  1
      script_log_file                 /var/opt/sgtest/tmp/cmcluster/pkg-
      simple/pkg-simple.log
```

Halt the package.

```
      $cmhaltpkg pkg-simple
```

Check the configuration of the new package.

```
      $cmcheckconf –P pkg-simple.conf
      cmcheckconf: Verification completed with no errors found.
      Use the cmapplyconf command to apply the configuration.
```

Apply the new configuration.

```
      $cmapplyconf –P pkg-simple.conf

      Modify the package configuration ([y]/n)? y
      Completed the cluster update
```

Start the package.

```
      $cmrunpkg pkg-simple
```

Migrating a Package with Customer-Defined Functions

If the package includes a user-defined script, or the control script contains non-Serviceguard functions in the customer-defined functions area, follow these steps:

1. Run the *cmmigratepkg* command and generate a new modular package configuration file and external script file:
   $cmmigratepkg –p pkgA –x /etc/cmcluster/pkg/pkgA/myexternal.sh –o pkgA.conf
2. Make sure that the location of the external script is correct in the output modular package configuration file. (The –x option expects the full pathname of the external script.)
3. Copy the external script file to each node where the package can run and make sure that the script is owned by root and permissions are set to 744.*
4. Run *cmcheckconf* with the new modular package configuration file.
5. Halt the package

6. Run *cmapplyconf* with the new modular package configuration file.
7. Run the package.

* **Note:** cmmigratepkg wrongly sets permissions to 555; you need to reset them manually to 744.

Migrating a Package with Non-Serviceguard Parameters in the Control Script

If the package includes non-Serviceguard control-script parameters that are not defined in the Customer Defined Functions section, or you want to convert these non-Serviceguard parameters to PEVs, use the following steps:

1. Run *cmmigratepkg* command with the options for external script and generate PEV.
   $cmmigratepkg –p pkgA –x /etc/cmcluster/pkg/pkgA/myexternal.sh –e –o pkgA.conf
2. Make sure that the location of the external script is correct in the output modular package configuration file. (The –x option expects the full pathname of the external script.) Note that the new variable are added with the prefix "PEV_".
3. Edit the external script
   a. Make sure that the script is using the PEV names in the code. Change the varaible names to include the prefix "PEV_".
4. Copy the external script file to each node where the package can run and make sure that the script is owned by root and permissions are set to 744.*
5. Halt the package.
6. Run *cmcheckconf* with the new modular package configuration file.
7. Run *cmapplyconf* with the new modular package configuration file.
8. Run the package.

* **Note:** cmmigratepkg wrongly sets permissions to 555; you need to reset them manually to 744.

Manual Steps for Migrating Legacy Packages

1. Get a copy of the legacy package ASCII file by using the cmgetconf –p command. Modify the following in the ASCII file:

   a. Remove the RUN_SCRIPT, HALT_SCRIPT and STORAGE_GROUP attributes.
   b. Add the script_log_file attribute
   c. Add to the list of modules and versions after PACKAGE_NAME:

      1. module_name sg/all
      2. module_version 1

   d. Add the operation_sequence information after Service information

      1. operation_sequence $SGCONF/scripts/sg/external_pre.sh
      2. operation_sequence $SGCONF/scripts/sg/volume_group.sh
      3. operation_sequence $SGCONF/scripts/sg/filesystem.sh
      4. operation_sequence $SGCONF/scripts/sg/package_ip.sh
      5. operation_sequence $SGCONF/scripts/sg/external.sh
      6. operation_sequence $SGCONF/scripts/sg/service.sh

      If the package uses EMS resources add:
      operation_sequence $SGCONF/scripts/sg/resource.sh

Legacy Package ASCII file:

```
package_name    pkg-nu-2
package_type    FAILOVER
run_script_timeout      340
halt_script_timeout     340
successor_halt_timeout  NO_TIMEOUT
priority        663
run_script      /etc/cmcluster/pkg-nu-2/control.sh
halt_script     /etc/cmcluster/pkg-nu-2/control.sh
node_name       krabs
node_name       sandy
service_name    pkg-nu-2_srv_1
service_fail_fast_enabled       NO
service_halt_timeout    5
service_name    pkg-nu-2_srv_2
service_fail_fast_enabled       NO
service_halt_timeout    5
subnet  192.42.2.0
subnet  fec0:0:0:2a02::/64
```

Modified package ASCII File:

```
package_name    pkg-nu-2
module_name                     sg/all
module_version                  1
package_type    FAILOVER
run_script_timeout      340
halt_script_timeout     340
successor_halt_timeout  NO_TIMEOUT
priority        663
_script     /etc/cmcluster/pkg-nu-2/control.sh
node_name       krabs
node_name       sandy
service_name    pkg-nu-2_srv_1
service_fail_fast_enabled       NO
service_halt_timeout    5
service_name    pkg-nu-2_srv_2
service_fail_fast_enabled       NO
service_halt_timeout    5
subnet  192.42.2.0
subnet  fec0:0:0:2a02::/64
operation_sequence $SGCONF/scripts/sg/external_pre.sh
operation_sequence $SGCONF/scripts/sg/volume_group.sh
operation_sequence $SGCONF/scripts/sg/filesystem.sh
operation_sequence $SGCONF/scripts/sg/package_ip.sh
operation_sequence $SGCONF/scripts/sg/external.sh
operation_sequence $SGCONF/scripts/sg/service.sh
```

2. Modify a copy of the Package Control Script. Edit the file to:

   - Remove lines from # START OF CUSTOMER DEFINED FUNCTIONS to the end of the file.
   - Remove the top lines up to VGCHANGE
   - Remove the comment after VGCHANGE

- Change SUBNET to IP_SUBNET
- Move all IP_ADDRESSES under the appropriate IP_SUBNET's.
- Change "-R" to "unlimited"
- Change in SERVICE_RESTART values of "-r <value>" to just <value>
- Remove all of the array values -
- Remove all "="
- Remove "" from value for KILL_PROCESSES_ACCESSING_RAW_DEVICES
- Make sure all file system attributes are in separate lines
- Remove quotation marks from the values of FS and LV.

Package control script after the modifications:

```
VGCHANGE "vgchange -a e"
CVM_ACTIVATION_CMD "vxdg -g \$DiskGroup set activation exclusivewrite"
VG /dev/vglock2
DEACTIVATION_RETRY_COUNT 2
KILL_PROCESSES_ACCESSING_RAW_DEVICES NO
LV /dev/vglock2/lvol1
 FS /var/opt/tmp/mnt/dev/vglock2/lvol1
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol10
 FS /var/opt/tmp/mnt/dev/vglock2/lvol10
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol2
 FS /var/opt/tmp/mnt/dev/vglock2/lvol2
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol3
 FS /var/opt/tmp/mnt/dev/vglock2/lvol3
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol4
 FS /var/opt/tmp/mnt/dev/vglock2/lvol4
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol5
 FS /var/opt/tmp/mnt/dev/vglock2/lvol5
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol6
 FS /var/opt/tmp/mnt/dev/vglock2/lvol6
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
```

```
LV /dev/vglock2/lvol7
 FS /var/opt/sgtest/tmp/mnt/dev/vglock2/lvol7
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol8
 FS /var/opt/sgtest/tmp/mnt/dev/vglock2/lvol8
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol9
 FS /var/opt/tmp/mnt/dev/vglock2/lvol9
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
VXVOL "vxvol -g \$DiskGroup startall"
FS_UMOUNT_COUNT 1
FS_MOUNT_RETRY_COUNT 0
CONCURRENT_VGCHANGE_OPERATIONS 1
CONCURRENT_FSCK_OPERATIONS 1
CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS 1
IP_SUBNET 192.42.2.0

IP 192.42.2.19
IP_SUBNET 3ffe:1000:0:2a02::
IP 3ffe:1000:0:2a02::11/64
SERVICE_NAME pkg-nu-2_srv_1
SERVICE_CMD "/etc/cmcluster/pkg-nu-2/service.pl >> /etc/cmcluster/pkg-nu-
2/pkg-nu-2_srv_1.log 2>&1"
SERVICE_RESTART  2
SERVICE_NAME pkg-nu-2_srv_2
SERVICE_CMD "/etc/cmcluster/pkg-nu-2/simple >> /etc/cmcluster/pkg-nu-
2/pkg-nu-2_srv_2.log 2>&1"
SERVICE_RESTART  unlimited
```

3. Concatenate the modified package configuration (ASCII) file with the modified package control script
4. Fix the Service attribute information, by moving the SERVICE_COMMAND, SERVICE_RESTART up to the location where SERVICE_NAME is first defined. Remove extra SERVICE_NAME attributes.

```
PACKAGE_NAME                    pkg-nu-26561_3
module_name                       sg/all
module_version                    1

PACKAGE_TYPE                    FAILOVER
NODE_NAME                plankton
NODE_NAME                patrick
NODE_NAME                krabs
NODE_NAME                sandy
AUTO_RUN                 YES
NODE_FAIL_FAST_ENABLED              NO
```

```
RUN_SCRIPT_TIMEOUT                  340
HALT_SCRIPT_TIMEOUT                 340
FAILOVER_POLICY               CONFIGURED_NODE
FAILBACK_POLICY               MANUAL
LOCAL_LAN_FAILOVER_ALLOWED                YES
MONITORED_SUBNET              192.42.2.0
MONITORED_SUBNET              fec0:0:0:2a02::/64
SERVICE_NAME                  pkg-nu-26561_3srv26561_1
SERVICE_CMD "/etc/cmcluster/pkg-nu-2/service.pl >> /etc/cmcluster/pkg-nu-
2/pkg-nu-2_srv_1.log 2>&1"
SERVICE_RESTART  2
SERVICE_FAIL_FAST_ENABLED                 NO
SERVICE_HALT_TIMEOUT                5
SERVICE_NAME                  pkg-nu-26561_3srv26561_2
SERVICE_CMD "/etc/cmcluster/pkg-nu-2/simple >> /etc/cmcluster/pkg-nu-
2/pkg-nu-2_srv_2.log 2>&1"
SERVICE_RESTART  2
SERVICE_FAIL_FAST_ENABLED                 NO
SERVICE_HALT_TIMEOUT                5
operation_sequence $SGCONF/scripts/sg/external_pre.sh
operation_sequence $SGCONF/scripts/sg/volume_group.sh
operation_sequence $SGCONF/scripts/sg/filesystem.sh
operation_sequence $SGCONF/scripts/sg/package_ip.sh
operation_sequence $SGCONF/scripts/sg/external.sh
operation_sequence $SGCONF/scripts/sg/service.sh

VGCHANGE "vgchange -a e"
CVM_ACTIVATION_CMD "vxdg -g \$DiskGroup set activation exclusivewrite"
VG /dev/vglock2
DEACTIVATION_RETRY_COUNT 2
KILL_PROCESSES_ACCESSING_RAW_DEVICES NO
LV /dev/vglock2/lvol1
 FS /var/opt/tmp/mnt/dev/vglock2/lvol1
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol10
 FS /var/opt/tmp/mnt/dev/vglock2/lvol10
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol2
 FS /var/opt/tmp/mnt/dev/vglock2/lvol2
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol3
 FS /var/opt/tmp/mnt/dev/vglock2/lvol3
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol4
 FS /var/opt/tmp/mnt/dev/vglock2/lvol4
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
```

```
LV /dev/vglock2/lvol5
 FS /var/opt/tmp/mnt/dev/vglock2/lvol5
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol6
 FS /var/opt/tmp/mnt/dev/vglock2/lvol6
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol7
 FS /var/opt/tmp/mnt/dev/vglock2/lvol7
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol8
 FS /var/opt/tmp/mnt/dev/vglock2/lvol8
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
LV /dev/vglock2/lvol9
 FS /var/opt/tmp/mnt/dev/vglock2/lvol9
 FS_TYPE ""
 FS_MOUNT_OPT ""
 FS_UMOUNT_OPT ""
 FS_FSCK_OPT ""
VXVOL "vxvol -g \$DiskGroup startall"
FS_UMOUNT_RETRY_COUNT 1
FS_MOUNT_RETRY_COUNT 0
CONCURRENT_VGCHANGE_OPERATIONS 1
CONCURRENT_FSCK_OPERATIONS 1
CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS 1
IP_SUBNET 192.42.2.0
IP 192.42.2.19
IP_SUBNET 3ffe:1000:0:2a02::
IP 3ffe:1000:0:2a02::11/64
```

5. Run cmcheckconf on the new package ASCII file and resolve if there are any errors.
6. Halt the Package
7. Apply the new package ASCII file.

Appendix A

The following shell script modifies the package configuration (ASCII) file, making the changes prescribed under "Manual Steps for Migrating a Legacy Package" earlier in this document.

```
#!/bin/sh
# Input to script  is packagename
# The steps to convert are:
#  get the package controlscript name
#  get the package ascii file

# get arguments - input is package name
```

```
if (( $# == 0 ))
then
     echo "ERROR: Package name is required"
     exit 1;
fi

nupkg=$1
newpkg=$2

# Extract package attributes from package configuration file
# Remove run_script/halt_script and storage_group from ascii file
cmgetconf -v 0 -p $nupkg  |
sed -e '/^$/d'   \
    -e '/^RUN_SCRIPT    /d'   \
    -e '/^HALT_SCRIPT   /d' > ./tmp/nu-pkg.ascii


# Append the upcc module name, version and operation_sequence.
cat ./tmp/nu-pkg.ascii |
sed -e '/PACKAGE_NAME/a\
module_name                     sg/all \
module_version                  1 \
' \
-e '$a\
operation_sequence $SGCONF/scripts/sg/external_pre.sh \
operation_sequence $SGCONF/scripts/sg/volume_group.sh \
operation_sequence $SGCONF/scripts/sg/filesystem.sh \
operation_sequence $SGCONF/scripts/sg/package_ip.sh \
operation_sequence $SGCONF/scripts/sg/external.sh \
operation_sequence $SGCONF/scripts/sg/service.sh \
' > $newpkg
```

The following script automates some of the changes needed to modify the package control script.

```
#!/bin/sh
# Input to scrip  is packagename
# The steps to convert are:
#  get the package controlscript name
#  get the package ascii file

# get arguments - input is package name
if (( $# == 0 ))
then
     echo "ERROR: Package name is required"
     exit 1;
fi

nupkg=$1
nuscript=$2

# get control script from pkg1
nupkgcntl=$(cmviewcl -v -fline -p $nupkg | grep "run_script=")
nupkgcntl=${nupkgcntl#*=}

echo "control script = $nupkgcntl"

# Extract parameters from control script
# Get only parameters from control script
cat $nupkgcntl |
```

```
sed -e '/# START OF CUSTOMER DEFINED FUNCTIONS/,$d' > ./tmp/nu-pkg.cntl.0

# Extract package attributes from package control file
cat  ./tmp/nu-pkg.cntl.0   |  grep ^[A-Z_.-]*[*[0-9]*]*= |
egrep -v -i 'PATH|GFS|DATA_REP|RAID|HA_NFS_SCRIPT_EXTENSION' |
egrep -v -i 'DTC_NAME|^MD|HA_APP_SERVER|STORAGE_GROUP'  |
sed  -e 's/\[[0-9]*\]//g'  \
     -e 's/=/ /g'     \
     -e 's/# Default//'   \
     -e 's/^SUBNET/IP_SUBNET/'   \
     -e 's/\"-r\([ 0-9]*\)\"/\1/'  \
     -e 's/\"-R\"/unlimited/'> ./tmp/nu-pkg.cntl.1

# Put all parameters in separate lines
cat ./tmp/nu-pkg.cntl.1 |
tr '; ' '\012' > ./tmp/nu-pkg.cntl.2

# remove extra quotation marks
cat ./tmp/nu-pkg.cntl.2 |
sed -e '/FS /s/"//g' \
    -e '/^LV /s/"//g' > $nuscript
```

4AA0-XXXXENW, May 2007