An Oracle White Paper
December 2010

Setting Up a Virtual Network Automatically Using Oracle Solaris 11 Express Project Crossbow

ORACLE

Introduction

This paper is a step-by-step guide on how to set up a virtual network automatically within a single box using Project Crossbow technology and the Bourne shell scripts supplied with this paper. Using the scripts, you can set up a virtual network with no manual intervention.

> **Note**: This paper does not explain how Oracle Solaris 11 Express Project Crossbow works. There are many resources on opensolaris.org about virtual networking using Project Crossbow. For example, for more information about Oracle Solaris networking, Project Crossbow network virtualization, and related topics, see the PDF presentation linked to from this page: http://hub.opensolaris.org/bin/view/User+Group+fraosug/Treffen+6

The scripts provided with this paper can help system administrators, solution architects, and systems engineers perform a quick setup of Project Crossbow for test, demo, or JumpStart installation purposes.

This paper also presents an interactive (manual) installation so you can better understand how to customize the scripts, for example, by extending them to use network address translation (NAT) and IP filtering. By customizing the scripts, you can extend them for your own requirements and delve more deeply into Oracle Solaris Containers, Project Crossbow, routing, NAT, and IP filtering.

Description of the Scripts for an Automated Installation

The Bourne shell scripts for setting up a virtual network in a fully automated fashion consist of two shell functions, one for creating the network and one for removing the network. You execute these functions by running two small exec scripts. There is also a variable declaration script for effectively controlling the parameters of the scripts.

The shell scripts' functions can help you encapsulate and group common tasks into many easily manageable, customizable executable modules.

Here is a description of the six scripts and two associated files:

> **Note**: See the appendix for a code listing of all the scripts and the `var.init` file. The scripts and associated files were tested with Oracle Solaris 11 Express and with Solaris Express Community Edition Build 130.

- `os_mk_crossbow.sh`—This script is a collection of function modules to perform a creation of etherstubs, vNICs, zones, and so on. This is a script for Oracle Solaris 11 Express based systems. For Solaris Express Community Edition based systems, use `nv_mk_crossbow.sh` instead.
- `exec_os_mk_crossbow.sh`—To execute and control a function of interest, use this script. If, for example, you want to create zones but not etherstubs, you just comment out the appropriate function. This is a script for Oracle Solaris 11 Express based systems. For Solaris Express Community Edition based systems, use `exec_nv_mk_crossbow.sh` instead.
- `nv_mk_crossbow.sh`—This script is a collection of function modules to perform a creation of etherstubs, vNICs, zones, and so on. This is a script for Solaris Express Community Edition based systems. For Oracle Solaris 11 Express based systems, use `os_mk_crossbow.sh` instead.
- `exec_nv_mk_crossbow.sh`—To execute and control a function of interest, use this script. If, for example, you want to create zones but not etherstubs, you just comment out the appropriate function. This is a script for Solaris Express Community Edition based systems. For Oracle Solaris 11 Express based systems, use `exec_os_mk_crossbow.sh` instead.
- `cl_crossbow.sh`—This script is a collection of function modules to remove the entire virtual network or just part of its configuration.
- `exec_cl_crossbow.sh`—Use this script to execute the `cl_crossbow.sh` script.

- `var.init`—All user-definable variables are initialized within this file.

  Note: The following line is commented out in the `var.init` file:

  ```
  #RPW=`cat /etc/shadow |grep root|awk -F":" '{print $2}'`
  ```

  Then, the next line contains a hard-coded, encrypted password. The password used to generate the encrypted password was `newroot`. If you prefer to extract the password from your system, uncomment the commented-out line and then comment out the next line.

- `site.xml`—This file is useful for enabling or disabling Oracle Solaris Service Management Facility (SMF) services. In our case, we need IPv4 routing and PIv4-forwarding enabled on the gateway host. You can extract this file from any system using the following commands, or you can use the `site.xml` file provided in the `crossbow_scripts.zip` file that is associated with this paper:

  ```
  global# svccfg extract > site.xml
  global# cp -p site.xml /zones/zgwhost/root/var/svc/profile/site.xml
  ```

Here is an excerpt from the `site.xml` file showing the relevant lines for our setup:

```
…
<service name='network/ipv4-forwarding' type='service' version='0'>
<instance name='default' enabled='true'/>
</service>
<service name='network/routing/route' type='service' version='0'>
<instance name='default' enabled='true'/>
</service>
…
```

Network Topology

The virtual network will run on a single system using Project Crossbow as a virtual wire. (My system is an Oracle Sun Ultra 24 workstation with quad Q8200 CPUs and 8 GBytes ECC RAM.)

As shown in Figure 1, the network consists of hosts (zones), vSwitches (etherstubs), and a router. The virtual network is created using Oracle Solaris 11 Express Project Crossbow technology and the hosts are created using Oracle Solaris Zones. Every step necessary to achieve the virtual topology is shown in this paper.

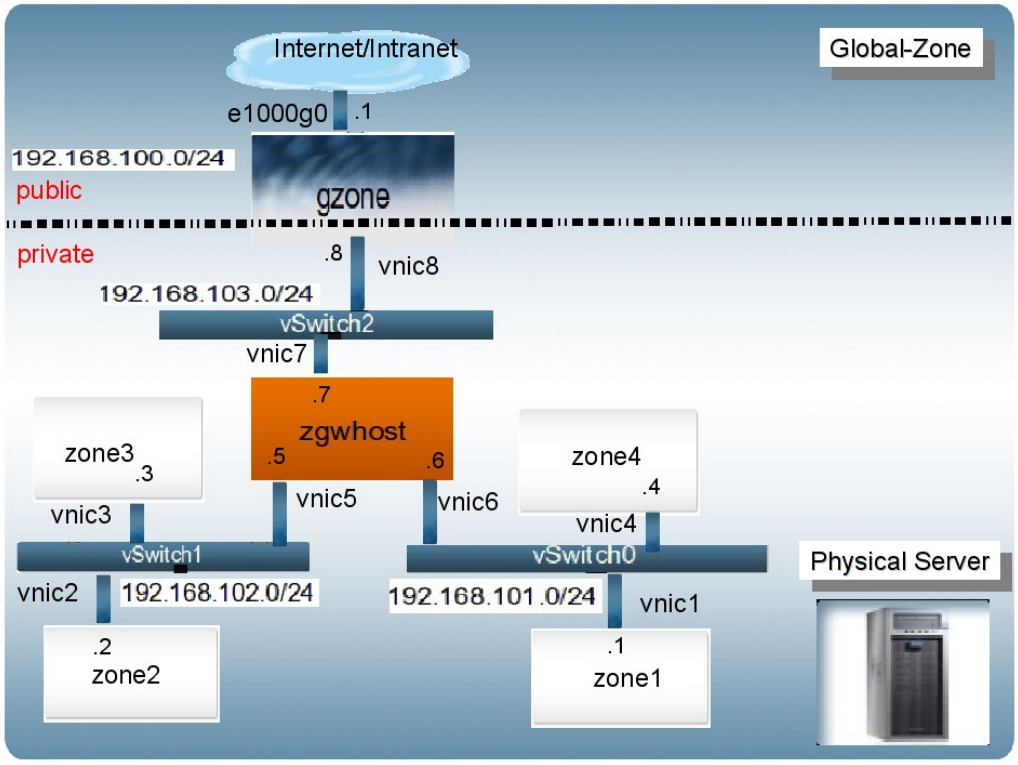

Figure 1: Network Topology

Prerequisites for Using the Scripts

All you need is a single host running Oracle Solaris 11 Express or Solaris Express Community Edition (preferably one of the newer builds, for example, 130).

Using the Scripts to Set Up a Virtual Network Automatically

Note: Before you use the scripts, it is a good idea to become familiar with what the scripts do by reading the "Setting Up the Virtual Network Interactively" section, which describes how to set up a virtual network interactively (manually).

1. Create a base directory called `/repository/crossbow-build` for `$CFG`.

Note: You can use a different name for the base directory. If you do, be sure you perform Step 4 to change the default parameter setting for `$CFG` in the `var.init` file.

2. Download the `crossbow_scripts.zip` file that is associated with this paper, which contains the scripts.
3. Copy the scripts to the base directory you created in Step 1.

4. If you used a name other than `/repository/crossbow-build` for the base directory or if you want to use a pool other than rpool for the zpool, edit the `var.init` file and change the value of the parameter settings for `$CFG` and `$MZNPL`, respectively.
5. Set the appropriate file permissions for the scripts (for example, 755).
6. Edit `exec_os_mk_crossbow.sh` (for an Oracle Solaris 11 Express system) or `exec_nv_mk_crossbow.sh` (for a Solaris Express Community Edition based system) to uncomment all functions so all functions are activated. For example:

```
. ./os_mk_crossbow.sh
verify_user
mketherstub
mkvnics
prepare_templatezone
prepare_gzone
mk_template_files
mkzones
```

Note: The first time you use the scripts, you need to execute all functions including the "template zone creation" function. Retain the template zone for later in case you need to remove or recreate the virtual network. Retaining the template zone saves a lot of time, because the other zones are cloned from the template zone.

7. Execute `exec_os_mk_crossbow.sh` (for an Oracle Solaris 11 Express system) or `exec_nv_mk_crossbow.sh` (for a Solaris Express Community Edition based system).
8. Go to the "Post-Installation Steps" section.

Using the Scripts to Remove or Re-create a Virtual Network

To remove a virtual network:

1. Check the `exec_cl_crossbow.sh` script to ensure the following lines look like this:

```
. ./cl_crossbow.sh
verify_user
cleanup_zones
cleanup_vinterfaces
```

2. Execute `exec_cl_crossbow.sh`.
3. If necessary, delete the mountpoints of every removed zone manually.

Note: After I successfully removed all zones including the template zone, I attempted to recreate the zones, but it did not work. The ZFS file systems are mounted out of order, and I got an error message that indicated this problem:

```
cannot mount  /.../...  directory not empty.
```

This happens only on Oracle Solaris 11 Express, not on Solaris Express Community Edition. The workaround (for Oracle Solaris 11 Express only): Prior to recreation, delete the mountpoints of every removed zone. (This workaround is integrated into the `exec_cl_crossbow.sh` cleanup script.)

To re-create a virtual network:

1. Comment out the `prepare_templatezone` function in the `exec_os_mk_crossbow.sh` script (if you have an Oracle Solaris 11 Express based system) or the `exec_nv_mk_crossbow.sh` script (if you have a Solaris Express Community Edition based system).
2. Execute `exec_os_mk_crossbow.sh` (if you have an Oracle Solaris 11 Express based system) or `exec_nv_mk_crossbow.sh` (if you have a Solaris Express Community Edition based system).
3. Go to the "Post-Installation Steps" section.

Post-Installation Steps

All the necessary steps for configuring this simple network are done by the scripts. After installation, just try to ping the hosts, or do any network communications from any to any host within the virtual network.

Feel free to extend the scripts for use in more complex networking environments and for implementing more networking features.

Setting Up a Virtual Network Interactively

If you want to create a virtual network interactively (manually), use the information in this section.

Note: All the manual configuration steps presented in this section are integrated into the shell scripts. Therefore, to understand what the scripts do, you should become familiar with these manual steps.

Overview of the Manual Steps

The starting point for creating a virtual network comprising hosts, switches, and routers is creating the necessary networking interfaces and setting up a template zone from which the other zones will be cloned. The virtual network is created using Oracle Solaris 11 Express Project Crossbow technology, and the hosts are created using Oracle Solaris Zones.

The general steps necessary for creating a virtual network interactively are as follows (see also Figure 1, Network Topology):

- Configure vSwitches for layer-two communication of all networked hosts.
- Configure a virtual network consisting of three subnetworks that are connected via a virtual router based on Project Crossbow.
- Set up Oracle Solaris Zones for the internal network and configure the global zone as a public network interface.

Create vNICs, vSwitches, and Oracle Solaris ZFS Pool and File System

The following sections show how to create vNICs, vSwitches, and an Oracle Solaris Zettabyte File System (ZFS) pool and file system.

Configure the Etherstubs (vSwitches)

```
Global# dladm create-etherstub etherstub0
Global# dladm create-etherstub etherstub1
Global# dladm create-etherstub etherstub2
```

Configure vNICs Within `etherstub0`

```
Global# dladm create-vnic -l etherstub0 vnic1
Global# dladm create-vnic -l etherstub0 vnic4
Global# dladm create-vnic -l etherstub0 vnic6
```

Configure vNICs Within `etherstub1`

```
Global# dladm create-vnic -l etherstub1 vnic2
Global# dladm create-vnic -l etherstub1 vnic3
Global# dladm create-vnic -l etherstub1 vnic5
```

Configure vNICs Within `etherstub2`

```
Global# dladm create-vnic -l etherstub2 vnic7
Global# dladm create-vnic -l etherstub2 vnic8
```

Configure ZFS Pool and File System for the Oracle Solaris Zones

```
Global# zfs create -o compression=on -o mountpoint=/zones rpool/zones
```

Create `szone` (Template Zone)

The following sections show how to create the template zone.

Configure Oracle Solaris Zone File for `szone`

Note: Exercise caution if you use Oracle Solaris 11 Express, because Solaris Express Community Edition and Oracle Solaris 11 Express use different packaging systems (IPS and SVR4). With Oracle Solaris 11 Express, you cannot set up a native zone with inherited and shared directories. Instead, Oracle Solaris 11 Express downloads the directories directly from the appropriate repository or publisher using the `ipkg` brand. Therefore, if you are using Oracle Solaris 11 Express, you must delete all lines that include the following:

```
add inherit-pkg-dir
set dir= xxx
end
```

I provided two scripts for creating the virtual network: one for Solaris Express Community Edition (`nv_mk_crossbow.sh`) and one for Oracle Solaris 11 Express (`os_mk_crossbow.sh`). Both are shown in their entirety in the appendix.

```
Global# cat > /repository/crossbow-build/szone << _EOF_
create -b
set zonepath=/zones/szone
set ip-type=exclusive
set autoboot=false
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
```

```
end
commit
_EOF_
```

Set Up `szone`

```
global# zonecfg -z <zone name> -f </path/to/config/file>
global# zonecfg -z szone -f /repository/crossbow-build/szone
global# zoneadm -z <zone name>  install
global# zoneadm -z szone install
```

Create `zone1`

The following sections show how to create the `zone1`.

Zone Configuration File for `zone1`

```
Global# cat > /repository/crossbow-build/zone1 << _EOF_
create -b
set zonepath=/zones/zone1
set ip-type=exclusive
set autoboot=true
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=vnic1
end
commit
_EOF_
```

Set Up `zone1`

```
global# zonecfg -z <zone name> -f </path/to/config/file>
global# zonecfg -z zone1 -f /repository/crossbow-build/zone1
global# zoneadm -z <zone name>  install
global# zoneadm -z zone1 install
```

Create `sysidcfg` for `zone1`

```
cat > /zones/zone1/root/etc/sysidcfg << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=i.93FnIMC3xc.
network_interface=vnic1
{primary
hostname=zone1
ip_address=192.168.101.1
netmask=255.255.255.0
protocol_ipv6=no
default_route=NONE }
timeserver=localhost
name_service=none
_EOF_
global# zoneadm -z zone1 boot
```

Create zone2

The following sections show how to create the zone2.

Zone Configuration File for zone2

```
Global#  cat >  /repository/crossbow-build/zone2 << _EOF_
create -b
set zonepath=/zones/zone2
set ip-type=exclusive
set autoboot=true
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=vnic2
end
commit
_EOF_
```

Set Up zone2

```
global# zonecfg -z <zone name> -f </path/to/config/file>
global# zonecfg -z zone2 -f /repository/crossbow-build/zone2
global# zoneadm -z <zone name>  install
global# zoneadm -z zone2 install
```

Create `sysidcfg` for `zone2`

```
cat > /zones/zone2/root/etc/sysidcfg << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=i.93FnIMC3xc.
network_interface=vnic2
{primary
hostname=zone2
ip_address=192.168.102.2
netmask=255.255.255.0
protocol_ipv6=no
default_route=NONE}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
global# zoneadm -z zone2  boot
```

Create `zone3`

The following sections show how to create the `zone3`.

Zone Configuration File for `zone3`

```
Global#  cat > /repository/crossbow-build/zone3 << _EOF_
create -b
set zonepath=/zones/zone3
set ip-type=exclusive
set autoboot=true
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
```

```
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=vnic3
end
commit
_EOF_
```

Set Up `zone3`

```
global# zonecfg -z <zone name> -f </path/to/config/file>
global# zonecfg -z zone3 -f /repository/crossbow-build/zone3
global# zoneadm -z <zone name>  install
global# zoneadm -z zone3 install
```

Create `sysidcfg` for `zone3`

```
cat > /zones/zone3/root/etc/sysidcfg << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=i.93FnIMC3xc.
network_interface=vnic3
{primary
hostname=zone3
ip_address=192.168.102.3
netmask=255.255.255.0
 protocol_ipv6=no
default_route=NONE}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
global# zoneadm -z zone3 boot
```

Create zone4

The following sections show how to create the zone4.

Zone Configuration File for zone4

```
Global#  cat > /repository/crossbow-build/zone4 << _EOF_
create -b
set zonepath=/zones/zone4
set ip-type=exclusive
set autoboot=true
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=vnic4
end
commit
_EOF_
```

Set Up zone4

```
global# zonecfg -z <zone name> -f </path/to/config/file>
global# zonecfg -z zone4 -f /repository/crossbow-build/zone4
global# zoneadm -z <zone name>  install
global# zoneadm -z zone4 install
```

Create `sysidcfg` for `zone4`

```
cat > /zones/zone4/root/etc/sysidcfg << EOF
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=i.93FnIMC3xc.
network_interface=vnic4
{primary hostname=zone4
ip_address=192.168.101.4
netmask=255.255.255.0
protocol_ipv6=no
default_route=NONE}
security_policy=none
timeserver=localhost
name_service=none
EOF
global# zoneadm -z zone4 boot
```

Create `zgwhost` (Gateway Host)

The following sections show how to create the gateway host and enable IPV4 forwarding and IPV4 routing.

Zone Configuration File for `zgwhost`

```
cat > /repository/crossbow-build/zgwhost << _EOF_
create -b
set zonepath=/zones/zgwhost
set ip-type=exclusive
set autoboot=true
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
```

```
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=vnic5
end
add net
set physical=vnic6
end
add net
set physical=vnic7
end
commit
_EOF_
```

Set Up Zone `zgwhost`

```
global# zonecfg -z <zone name> -f </path/to/config/file>
global# zonecfg -z zgwhost -f /repository/crossbow-build/zgwhost
global# zoneadm -z <zone name>  install
global# zoneadm -z zgwhost install
```

Create `sysidcfg` for `zgwhost`

```
cat > /zones/zgwhost/root/etc/sysidcfg << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=i.93FnIMC3xc.
network_interface=vnic5
{primary
hostname=zgwhost
ip_address=192.168.102.5
netmask=255.255.255.0
protocol_ipv6=no
default_route=NONE}
```

```
network_interface=vnic6
{primary
hostname=zgwhost1
ip_address=192.168.101.6
netmask=255.255.255.0
protocol_ipv6=no
default_route=NONE}
network_interface=vnic7
{primary
hostname=${ZGWHN}
ip_address=192.168.103.7
netmask=255.255.255.0
protocol_ipv6=no
default_route=NONE}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
```

Enable IPV4 Forwarding and IPV4 Routing

```
global# svccfg extract > site.xml
global# cp -p site.xml  /zones/zgwhost/root/var/svc/profile/site.xml
global# zoneadm -z zgwhost boot
```

Conclusion

This paper described how to set up a virtual network automatically, with no manual intervention, using Project Crossbow technology and Bourne shell scripts provided with the paper. This paper also described how to set up the same virtual network interactively (manually).

These methods were provided to help you better understand how network virtualization using Project Crossbow technology works and to make it easier for you to extend the scripts for use with more complex network features.

Appendix: Code for the Scripts and `var.init` File

The following sections show the content of the six scripts and the `var.init` file that are supplied in the `crossbow_scripts.zip` file associated with this paper.

> **Note**: See the "Description of the Scripts for an Automated Installation" section for more information about the scripts, the `var.init` file, and the `site.xml` file supplied in the `crossbow_scripts.zip` file.

The `os_mk_crossbow.sh` Script

This script is a collection of function modules to perform a creation of etherstubs, vNICs, zones, and so on. This is a script for Oracle Solaris 11 Express based systems. For Solaris Express Community Edition based systems, use `nv_mk_crossbow.sh` instead.

```
#!/bin/sh -x
DIR="`/bin/dirname $0`"
export DIR

. ${DIR}/var.init
#
# ============================================================================
# Verify you can only run this script as a root
# ============================================================================
verify_user()
{
# Verify we can only run as root
RUSER=0
EUSER=`id -u`
if [ ${RUSER} -ne ${EUSER} ]; then
    echo "you must be a root user;  Exiting..."
    exit 1
fi
}
#
# ============================================================================
# Function to check the existence of pool or filesystem for
# the zones, then create the template zone to clone the production zones
# ============================================================================
mketherstub()
{
```

```
#
#
# ========================================================================
# Create etherstub for the virtual network
# ========================================================================
for i in 0 1 2
do
#
eval DEST=`dladm show-etherstub|grep -v LINK|grep -i etherstub$i`
#
eval MESTS=`echo $MEST$i`
if [ -n "$MESTS" ]; then
       eval EST=`echo $MESTS`
   else
       eval EST=`echo $DEST`
fi
#
if  [ -z "$DEST" ]; then
               echo "Etherstub does not exists"
               dladm create-etherstub $EST
else
               echo "please make sure the same etherstub does not exist"
               exit 1
fi
#
echo "----------"
dladm show-etherstub
echo "-----------"
#
done
}
#
mkvnics()
{
# ========================================================================
# Create VNIC for the first subnetwork vswitch0
# ========================================================================
DEST=`dladm show-etherstub|grep -v LINK|grep -i $MEST0`
if  [ -n "$DEST" ]; then
for i in 1 4 6
do
```

```
eval DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -n "$MVNICS" ]; then
       eval VNIC=`echo $MVNICS`
   else
       eval VNIC=`echo $DVNIC`
fi
#
if  [ -z "$DVNIC" ]; then
               echo "VNIC does not exists"
               dladm create-vnic -l $DEST $VNIC
else
               echo "please make sure the same VNIC does not exist"
               exit 1
fi
#
echo "----------"
dladm show-vnic
echo "-----------"
#
done
fi
# ============================================================================
# Create VNIC for the second subnetwork vswitch1
# ============================================================================
DEST=`dladm show-etherstub|grep -v LINK|grep -i $MEST1`
if  [ -n "$DEST" ]; then
for i in 2 3 5
do
eval DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -n "$MVNICS" ]; then
       eval VNIC=`echo $MVNICS`
   else
       eval VNIC=`echo $DVNIC`
fi
#
if  [ -z "$DVNIC" ]; then
               echo "VNIC does not exists"
               dladm create-vnic -l $DEST $VNIC
else
```

```
                echo "please make sure the same VNIC does not exist"
                exit 1
fi
#
echo "----------"
dladm show-vnic
echo "-----------"
#
done
fi
# ========================================================================
# Create VNIC for the third subnetwork vswitch2
# ========================================================================
DEST=`dladm show-etherstub|grep -v LINK|grep -i $MEST2`
if  [ -n "$DEST" ]; then
for i in 7 8
do
eval DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -n "$MVNICS" ]; then
       eval VNIC=`echo $MVNICS`
   else
       eval VNIC=`echo $DVNIC`
fi
#
if  [ -z "$DVNIC" ]; then
                echo "VNIC does not exists"
                dladm create-vnic -l $DEST $VNIC
else
                echo "please make sure the same VNIC does not exist"
                exit 1
fi
#
echo "----------"
dladm show-vnic
echo "-----------"
#
done
fi
}
prepare_gzone()
```

```
{
DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic8|awk '{print $1}'`
echo "--------------------------------"
echo " configuring the Global Zone ..."
echo "--------------------------------"
if  [ -n "$DVNIC" ]; then
        echo "Configuring gzone ..."
        ifconfig  ${MVNIC8} plumb
        ifconfig  ${MVNIC8} ${VNIC8IP} netmask ${NM} up
        routeadm -e ipv4-forwarding
        routeadm -e ipv4-routing
        routeadm -u
else
                echo "VNIC does not exists"
                echo "please make sure the VNIC exists"
                exit 1
fi
echo "---------"
routeadm
echo "---------"
#
}
#
#
prepare_templatezone()
{
# ============================================================================
# Create a template zone to clone the other zones from
# Check the template zone is not used
# ============================================================================
DZNFS=`zfs list|grep -v "${ZZFS}"|grep ${MZNFS}|awk -F"/" '{print $NF}'`
DZNPL=`zfs list|grep -v "${ZZFS}"|grep ${MZNFS}|awk -F"/" '{print $1}'`
ZZ=`zoneadm list -c | grep -c ${SZHN}`
if [ ${ZZ} -gt 0 ]; then
        echo "${SZHN} exists"
        exit 1
fi
#
if  [ -n "${MZNFS}" ] &&
        [ -n "${MZNPL}" ]; then
       ZNFS="${MZNFS}"
```

```
       ZNPL="${MZNPL}"
   else
       ZNFS="${DZNFS}"
       ZNPL="${DZNPL}"
fi
if [ -z "${DZNFS}" ]; then
echo "--------------------------------"
echo " Filesystem does not exist ...  "
echo " Creating ZFS Filesystem ...    "
echo "--------------------------------"
      zfs create -o compression=on -o mountpoint=/${ZNFS} ${ZNPL}/${ZNFS}
cat > ${CFG}/${SZHN} << _EOF_
create -b
set zonepath=${PATHSZ}
set ip-type=exclusive
commit
_EOF_
        zonecfg -z ${SZHN} -f ${PATHSZCFG}
        zoneadm -z ${SZHN} install
else
        echo "Please delete this filesystem"
        exit 1
fi
}
#
mk_template_files()
{
# ============================================================================
# Create template file to create the zones cloned out of the template zone
# ============================================================================
for i in 1 2 3 4 5 6 7 8
do
DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -z "$DVNIC" ]; then
        echo " prior creating template files VNIC must exist, exiting ..."
        echo " Please please make sure VNIC is properly configured"
        exit 1
#
else
cat > ${CFG}/${ZGWHN} << _EOF_
```

```
create -b
set zonepath=${ZPATHGWHOST}
set ip-type=exclusive
set autoboot=true
add net
set physical=${MVNIC5}
end
add net
set physical=${MVNIC6}
end
add net
set physical=${MVNIC7}
end
commit
_EOF_
#
#
cat > ${CFG}/${Z1HN} << _EOF_
create -b
set zonepath=${ZPATHZONE1}
set ip-type=exclusive
set autoboot=true
add net
set physical=${MVNIC1}
end
commit
_EOF_
#
#
cat > ${CFG}/${Z2HN} << _EOF_
create -b
set zonepath=${ZPATHZONE2}
set ip-type=exclusive
set autoboot=true
add net
set physical=${MVNIC2}
end
commit
_EOF_
#
cat > ${CFG}/${Z3HN} << _EOF_
```

```
create -b
set zonepath=${ZPATHZONE3}
set autoboot=true
set ip-type=exclusive
add net
set physical=${MVNIC3}
end
commit
_EOF_
#
cat > ${CFG}/${Z4HN} << _EOF_
create -b
set zonepath=${ZPATHZONE4}
set autoboot=true
set ip-type=exclusive
add net
set physical=${MVNIC4}
end
commit
_EOF_
#
fi
done
}
#
mkzones()
{
#
# ============================================================================
# create the zone ${ZGWHN} which acts as a router
# ============================================================================
#
DZGWFS=`zfs list|grep ${ZGWHN}|awk -F"/" '{print $NF}'`
if [ -n "${DZGWFS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${ZGWHN} -f ${ZPATHGWTMP}
zoneadm -z ${ZGWHN} clone ${SZHN}
#
```

```
cat > ${ZPATHGWSYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC7} {primary hostname=${ZGWHN} ip_address=${VNIC7IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DRGW}}
network_interface=${MVNIC5} {hostname=${ZGWHN1} ip_address=${VNIC5IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR}}
network_interface=${MVNIC6} {hostname=${ZGWHN2} ip_address=${VNIC6IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
#cat > ${MZNGWFS}/root/etc/ipf/ipf_set.conf << _EOF_
#map vnic7 192.168.101.0/24 ->  192.168.103.7/32 portmap tcp/udp auto
#map vnic7 192.168.101.0/24 -> 192.168.103.7/32
#map vnic7 192.168.102.0/24 ->  192.168.103.7/32 portmap tcp/udp auto
#map vnic7 192.168.102.0/24 -> 192.168.103.7/32
#_EOF_
#
cp -p ${LSITE} ${RSITE}
zoneadm -z ${ZGWHN} boot
#
# ============================================================================
# create the zone ${Z1HN}
# ============================================================================
#
DZ1FS=`zfs list|grep ${Z1HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ1FS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z1HN} -f ${ZPATH1TMP}
zoneadm -z ${Z1HN} clone ${SZHN}
```

```
#
cat > ${ZPATH1SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC1} {primary hostname=${Z1HN} ip_address=${VNIC1IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR1}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z1HN} boot
# ============================================================================
# create the zone ${Z2HN}
# ============================================================================
#
DZ2FS=`zfs list|grep ${Z2HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ2FS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z2HN} -f ${ZPATH2TMP}
zoneadm -z ${Z2HN} clone ${SZHN}
#
cat > ${ZPATH2SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC2} {primary hostname=${Z2HN} ip_address=${VNIC2IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR2}}
security_policy=none
timeserver=localhost
```

```
name_service=none
_EOF_
fi
#
zoneadm -z ${Z2HN} boot
# ========================================================================
# create the zone ${Z3HN}
# ========================================================================
#
DZ3FS=`zfs list|grep ${Z3HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ3FS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z3HN} -f ${ZPATH3TMP}
zoneadm -z ${Z3HN} clone ${SZHN}
#
cat > ${ZPATH3SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC3} {primary hostname=${Z3HN} ip_address=${VNIC3IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR3}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z3HN} boot
# ========================================================================
# create the zone ${Z4HN}
# ========================================================================
#
DZ4FS=`zfs list|grep ${Z4HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ4FS}" ]; then
        echo " Filesystem exists, exiting ..."
```

```
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z4HN} -f ${ZPATH4TMP}
zoneadm -z ${Z4HN} clone ${SZHN}
#
cat > ${ZPATH4SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC4} {primary hostname=${Z4HN} ip_address=${VNIC4IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR4}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z4HN} boot
}
```

The `exec_os_mk_crossbow.sh` Script

To execute and control a function of interest, use this script. If, for example, you want to create zones but not etherstubs, you just comment out the appropriate function. This is a script for Oracle Solaris 11 Express based systems. For Solaris Express Community Edition based systems, use `exec_nv_mk_crossbow.sh` instead.

```
#!/bin/sh -x
#
DIR="`/bin/dirname $0`"
export DIR
. ${DIR}/var.init
#
#
. ./os_mk_crossbow.sh
verify_user
#mketherstub
#mkvnics
```

```
#prepare_templatezone
#prepare_gzone
#mk_template_files
#mkzones
```

The nv_mk_crossbow.sh Script

This script is a collection of function modules to perform a creation of etherstubs, vNICs, zones, and so on. This is a script for Solaris Express Community Edition based systems. For Oracle Solaris 11 Express based systems, use os_mk_crossbow.sh instead.

```
#!/bin/sh -x

DIR="`/bin/dirname $0`"
export DIR

. ${DIR}/var.init
#
# ============================================================================
# Verify you can only run this script as a root
# ============================================================================
verify_user()
{
# Verify we can only run as root
RUSER=0
EUSER=`id -u`
if [ ${RUSER} -ne ${EUSER} ]; then
    echo "you must be a root user;  Exiting..."
    exit 1
fi
}
#
# ============================================================================
# Function to check the existence of pool or filesystem for
# the zones, then create the template zone to clone the production zones
# ============================================================================
mketherstub()
{
#
#
# ============================================================================
# Create etherstub for the virtual network
```

```
# ========================================================================
for i in 0 1 2
do
#
eval DEST=`dladm show-etherstub|grep -v LINK|grep -i etherstub$i`
#
eval MESTS=`echo $MEST$i`
if [ -n "$MESTS" ]; then
       eval EST=`echo $MESTS`
   else
       eval EST=`echo $DEST`
fi
#
if  [ -z "$DEST" ]; then
               echo "Etherstub does not exists"
               dladm create-etherstub $EST
else
               echo "please make sure the same etherstub does not exist"
               exit 1
fi
#
echo "----------"
dladm show-etherstub
echo "-----------"
#
done
}
#
mkvnics()
{
# ========================================================================
# Create VNIC for the first subnetwork vswitch0
# ========================================================================
DEST=`dladm show-etherstub|grep -v LINK|grep -i $MEST0`
if  [ -n "$DEST" ]; then
for i in 1 4 6
do
eval DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -n "$MVNICS" ]; then
       eval VNIC=`echo $MVNICS`
```

```
   else
       eval VNIC=`echo $DVNIC`
fi
#
if  [ -z "$DVNIC" ]; then
               echo "VNIC does not exists"
               dladm create-vnic -l $DEST $VNIC
else
               echo "please make sure the same VNIC does not exist"
               exit 1
fi
#
echo "----------"
dladm show-vnic
echo "-----------"
#
done
fi
# ============================================================================
# Create VNIC for the second subnetwork vswitch1
# ============================================================================
DEST=`dladm show-etherstub|grep -v LINK|grep -i $MEST1`
if  [ -n "$DEST" ]; then
for i in 2 3 5
do
eval DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -n "$MVNICS" ]; then
       eval VNIC=`echo $MVNICS`
   else
       eval VNIC=`echo $DVNIC`
fi
#
if  [ -z "$DVNIC" ]; then
               echo "VNIC does not exists"
               dladm create-vnic -l $DEST $VNIC
else
               echo "please make sure the same VNIC does not exist"
               exit 1
fi
#
```

```
echo "----------"
dladm show-vnic
echo "-----------"
#
done
fi
# ============================================================================
# Create VNIC for the third subnetwork vswitch2
# ============================================================================
DEST=`dladm show-etherstub|grep -v LINK|grep -i $MEST2`
if  [ -n "$DEST" ]; then
for i in 7 8
do
eval DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -n "$MVNICS" ]; then
       eval VNIC=`echo $MVNICS`
   else
       eval VNIC=`echo $DVNIC`
fi
#
if  [ -z "$DVNIC" ]; then
               echo "VNIC does not exists"
               dladm create-vnic -l $DEST $VNIC
else
               echo "please make sure the same VNIC does not exist"
               exit 1
fi
#
echo "----------"
dladm show-vnic
echo "-----------"
#
done
fi
}
prepare_gzone()
{
DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic8|awk '{print $1}'`
echo "--------------------------------"
echo " configuring the Global Zone ..."
```

```
echo "--------------------------------"
if  [ -n "$DVNIC" ]; then
        echo "Configuring gzone ..."
        ifconfig  ${MVNIC8} plumb
        ifconfig  ${MVNIC8} ${VNIC8IP} netmask ${NM} up
        routeadm -e ipv4-forwarding
        routeadm -e ipv4-routing
        routeadm -u
else
                echo "VNIC does not exists"
                echo "please make sure the VNIC exists"
                exit 1
fi
echo "---------"
routeadm
echo "---------"
#
}
#
#
prepare_templatezone()
{
# ============================================================================
# Create a template zone to clone the other zones from
# Check the template zone is not used
# ============================================================================
DZNFS=`zfs list|grep -v "${ZZFS}"|grep ${MZNFS}|awk -F"/" '{print $NF}'`
DZNPL=`zfs list|grep -v "${ZZFS}"|grep ${MZNFS}|awk -F"/" '{print $1}'`
ZZ=`zoneadm list -c | grep -c ${SZHN}`
if [ ${ZZ} -gt 0 ]; then
        echo "${SZHN} exists"
        exit 1
fi
#
if  [ -n "${MZNFS}" ] &&
        [ -n "${MZNPL}" ]; then
       ZNFS="${MZNFS}"
       ZNPL="${MZNPL}"
   else
       ZNFS="${DZNFS}"
       ZNPL="${DZNPL}"
```

```
fi
if [ -z "${DZNFS}" ]; then
echo "--------------------------------"
echo " Filesystem does not exist ...  "
echo " Creating ZFS Filesystem ...    "
echo "--------------------------------"
      zfs create -o compression=on -o mountpoint=/${ZNFS} ${ZNPL}/${ZNFS}
cat > ${CFG}/${SZHN} << _EOF_
create -b
set zonepath=${PATHSZ}
set ip-type=exclusive
commit
_EOF_
        zonecfg -z ${SZHN} -f ${PATHSZCFG}
        zoneadm -z ${SZHN} install
else
        echo "Please delete this filesystem"
        exit 1
fi
}
#
mk_template_files()
{
# ============================================================================
# Create template file to create the zones cloned out of the template zone
# ============================================================================
for i in 1 2 3 4 5 6 7 8
do
DVNIC=`dladm show-vnic|grep -v LINK|grep -i vnic$i|awk '{print $1}'`
eval MVNICS=`echo $MVNIC$i`
if [ -z "$DVNIC" ]; then
        echo " prior creating template files VNIC must exist, exiting ..."
        echo " Please please make sure VNIC is properly configured"
        exit 1
#
else
cat > ${CFG}/${ZGWHN} << _EOF_
create -b
set zonepath=${ZPATHGWHOST}
set autoboot=true
set ip-type=exclusive
```

```
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=${MVNIC5}
end
add net
set physical=${MVNIC6}
end
add net
set physical=${MVNIC7}
end
commit
_EOF_
#
#
cat > ${CFG}/${Z1HN} << _EOF_
create -b
set zonepath=${ZPATHZONE1}
set autoboot=true
set ip-type=exclusive
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
```

```
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=${MVNIC1}
end
commit
_EOF_
#
#
cat > ${CFG}/${Z2HN} << _EOF_
create -b
set zonepath=${ZPATHZONE2}
set autoboot=true
set ip-type=exclusive
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=${MVNIC2}
end
commit
_EOF_
#
cat > ${CFG}/${Z3HN} << _EOF_
```

```
create -b
set zonepath=${ZPATHZONE3}
set autoboot=true
set ip-type=exclusive
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=${MVNIC3}
end
commit
_EOF_
#
cat > ${CFG}/${Z4HN} << _EOF_
create -b
set zonepath=${ZPATHZONE4}
set autoboot=true
set ip-type=exclusive
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
```

```
end
add inherit-pkg-dir
set dir=/opt
end
add net
set physical=${MVNIC4}
end
commit
_EOF_
#
fi
done
}
#
mkzones()
{
#
# ============================================================================
# create the zone ${ZGWHN} which acts as a router
# ============================================================================
#
DZGWFS=`zfs list|grep ${ZGWHN}|awk -F"/" '{print $NF}'`
if [ -n "${DZGWFS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${ZGWHN} -f ${ZPATHGWTMP}
zoneadm -z ${ZGWHN} clone ${SZHN}
#
cat > ${ZPATHGWSYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC7} {primary hostname=${ZGWHN} ip_address=${VNIC7IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DRGW}}
network_interface=${MVNIC5} {hostname=${ZGWHN1} ip_address=${VNIC5IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR}}
```

```
network_interface=${MVNIC6} {hostname=${ZGWHN2} ip_address=${VNIC6IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
#cat > ${MZNGWFS}/root/etc/ipf/ipf_set.conf << _EOF_
#map vnic7 192.168.101.0/24 ->  192.168.103.7/32 portmap tcp/udp auto
#map vnic7 192.168.101.0/24 -> 192.168.103.7/32
#map vnic7 192.168.102.0/24 ->  192.168.103.7/32 portmap tcp/udp auto
#map vnic7 192.168.102.0/24 -> 192.168.103.7/32
#_EOF_
#
cp -p ${LSITE} ${RSITE}
zoneadm -z ${ZGWHN} boot
#
# ============================================================================
# create the zone ${Z1HN}
# ============================================================================
#
DZ1FS=`zfs list|grep ${Z1HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ1FS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z1HN} -f ${ZPATH1TMP}
zoneadm -z ${Z1HN} clone ${SZHN}
#
cat > ${ZPATH1SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC1} {primary hostname=${Z1HN} ip_address=${VNIC1IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR1}}
security_policy=none
```

```
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z1HN} boot
# ============================================================================
# create the zone ${Z2HN}
# ============================================================================
#
DZ2FS=`zfs list|grep ${Z2HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ2FS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z2HN} -f ${ZPATH2TMP}
zoneadm -z ${Z2HN} clone ${SZHN}
#
cat > ${ZPATH2SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC2} {primary hostname=${Z2HN} ip_address=${VNIC2IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR2}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z2HN} boot
# ============================================================================
# create the zone ${Z3HN}
# ============================================================================
#
DZ3FS=`zfs list|grep ${Z3HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ3FS}" ]; then
```

```
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z3HN} -f ${ZPATH3TMP}
zoneadm -z ${Z3HN} clone ${SZHN}
#
cat > ${ZPATH3SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC3} {primary hostname=${Z3HN} ip_address=${VNIC3IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR3}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z3HN} boot
# ========================================================================
# create the zone ${Z4HN}
# ========================================================================
#
DZ4FS=`zfs list|grep ${Z4HN}|awk -F"/" '{print $NF}'`
if [ -n "${DZ4FS}" ]; then
        echo " Filesystem exists, exiting ..."
        echo " Please delete this filesystem"
        exit 1
else
zonecfg -z ${Z4HN} -f ${ZPATH4TMP}
zoneadm -z ${Z4HN} clone ${SZHN}
#
cat > ${ZPATH4SYSIDCFG} << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
```

```
security_policy=NONE
root_password=${RPW}
network_interface=${MVNIC4} {primary hostname=${Z4HN} ip_address=${VNIC4IP}
netmask=${NM} protocol_ipv6=${IPV6} default_route=${DR4}}
security_policy=none
timeserver=localhost
name_service=none
_EOF_
fi
#
zoneadm -z ${Z4HN} boot
}
```

The `exec_nv_mk_crossbow.sh` Script

To execute and control a function of interest, use this script. If, for example, you want to create zones but not etherstubs, you just comment out the appropriate function. This is a script for Solaris Express Community Edition based systems. For Oracle Solaris 11 Express based systems, use `exec_os_mk_crossbow.sh` instead.

```
#!/bin/sh -x
#
DIR="`/bin/dirname $0`"
export DIR
. ${DIR}/var.init
#
#
. ./nv_mk_crossbow.sh
verify_user
#mketherstub
#mkvnics
#prepare_templatezone
#prepare_gzone
#mk_template_files
#mkzones
```

The `cl_crossbow.sh` Script

This script is a collection of function modules to remove the entire virtual network or just part of its configuration.

```
#!/bin/sh -x
DIR="`/bin/dirname $0`"
export DIR

. ${DIR}/var.init
#
# ============================================================================
# Verify you can only run this script as a root
# ============================================================================
verify_user()
{
# Verify we can only run as root
RUSER=0
EUSER=`id -u`
if [ ${RUSER} -ne ${EUSER} ]; then
    echo "you must be a root user;  Exiting..."
    exit 1
fi
}
#
cleanup_zones()
{
# ============================================================================
# remove the zones 1 - 4
# ============================================================================
for i in ${Z1HN} ${Z2HN} ${Z3HN} ${Z4HN} ${ZGWHN} ${SZHN}
#for i in ${Z1HN} ${Z2HN} ${Z3HN} ${Z4HN} ${ZGWHN}
do
DZNFS=`zfs list|grep "${i}"|awk -F"/" '{print $NF}'`
DZNPL=`zfs list|grep "${i}"|awk -F"/" '{print $1}'`
ZZ=`zoneadm list -i|grep ${i}|awk '{print $1}'`
RV=$?
if [ ${RV} -eq 0 ]; then
               echo "zone ${i} will be removed ..."
               zoneadm -z ${i} halt
               zoneadm -z ${i} uninstall -F
```

```
                zonecfg -z ${i} delete -F
                zfs destroy -rf ${MZNPL}/${MZNFS}/${i}
else
                echo "${i} does not exist"
                exit 1
fi
done
#
zfs destroy -f ${MZNPL}/${MZNFS}
rm -rf /${MZNFS}
}
#
cleanup_vinterfaces()
{
# ============================================================================
# If NIC name is not set manually obtain the name automatically from the
# dladam command.
# ============================================================================
#
ifconfig ${MVNIC8} unplumb
EST0=`dladm show-etherstub|grep -v LINK|grep -i ${MEST0}|awk '{print $1}'`
EST1=`dladm show-etherstub|grep -v LINK|grep -i ${MEST1}|awk '{print $1}'`
EST2=`dladm show-etherstub|grep -v LINK|grep -i ${MEST2}|awk '{print $1}'`
#
if  [ -z "${EST0}" ] &&
        [ -z "${EST1}" ] &&
        [ -z "${EST2}" ]; then
                if [ "${EST0}" != "${MEST0}" ] &&
                        [ "{EST1} != "${MEST1}" ] &&
                        [ "{EST2} != "${MEST2}" ]; then
                echo "Etherstub does not exist"
                exit
                fi
else
                for i in 1 2 3 4 5 6 7 8
                do
                        dladm delete-vnic vnic$i
                done
#
                for i in 0 1 2
                do
```

```
                        dladm delete-etherstub etherstub${i}
                done
fi
#
}
```

The exec_cl_crossbow.sh Script

Use this script to execute the cl_crossbow.sh script.

```
#!/bin/sh -x

DIR="`/bin/dirname $0`"
export DIR
. ${DIR}/var.init
#
#
. ./cl_crossbow.sh
verify_user
cleanup_zones
cleanup_vinterfaces
```

The var.init File

All user-definable variables are initialized within this file.

> **Note**: The following line is commented out in the var.init file:
>
> ```
> #RPW=`cat /etc/shadow |grep root|awk -F":" '{print $2}'`
> ```
>
> Then, the next line contains a hard-coded, encrypted password. The password used to generate the encrypted password was newroot. If you prefer to extract the password from your system, uncomment the commented-out line and then comment out the next line.

```
#!/bin/sh -x
# ======================================================================
umask 022
CFG=/repository/crossbow-build
TIME=$(date +"%F-%H:%M")
LOGFILE=${CFG}/log.${TIME}
MZNPL=rpool
MZNFS=zones
PROF=root/var/svc/profile
```

```
SYSIDCFG=root/etc/sysidcfg
ZNFS=${MZNPL}/${MZNFS}
ZZFS=${MZNPL}/${MZNFS}/
MZN1FS=/${MZNFS}/zone1
MZN2FS=/${MZNFS}/zone2
MZN3FS=/${MZNFS}/zone3
MZN4FS=/${MZNFS}/zone4
MZNGWFS=/${MZNFS}/zgwhost
# ============================================================================
# Define general variables
# ============================================================================
# ============================================================================
# Manually define the VNICS specific variables.
# ============================================================================
MVNIC=vnic
MVNIC1=vnic1
MVNIC2=vnic2
MVNIC3=vnic3
MVNIC4=vnic4
MVNIC5=vnic5
MVNIC6=vnic6
MVNIC7=vnic7
MVNIC8=vnic8
# ============================================================================
# Manually define the Phy. Interface specific variables.
# ============================================================================
VNIC8IP=192.168.103.8
MIFC=rge0
NM=255.255.255.0
DR1=192.168.101.6
DR2=192.168.102.5
DR3=192.168.102.5
DR4=192.168.101.6
DR=NONE
DRGW=192.168.103.8
IPV6=no
# ============================================================================
# Manually define the Etherstubs. Interface specific variables.
# ============================================================================
MEST=etherstub
MEST0=etherstub0
```

```
MEST1=etherstub1
MEST2=etherstub2
# ============================================================================
# Extract root password from shadow specific variables.
# ============================================================================
#RPW=`cat /etc/shadow |grep root|awk -F":" '{print $2}'`
RPW="i.93FnIMC3xc."
# ============================================================================
# Define the template zone specific variables.
# ============================================================================
SZHN=szone
PATHSZ=/${MZNFS}/${SZHN}
PATHSZCFG=${CFG}/${SZHN}
#
# ============================================================================
# Define the gateway zone specific variables.
# ============================================================================
LSITE=${CFG}/site.xml
RSITE=${MZNGWFS}/${PROF}/site.xml
ZGWHN=zgwhost
ZGWHN1=zgwhost1
ZGWHN2=zgwhost2
VNIC5IP=192.168.102.5
VNIC6IP=192.168.101.6
VNIC7IP=192.168.103.7
ZPATHGWHOST=/${MZNFS}/${ZGWHN}
ZPATHGWTMP=${CFG}/${ZGWHN}
ZPATHGWSYSIDCFG=/${MZNFS}/${ZGWHN}/${SYSIDCFG}
ZONEGWBASEPATH=/${MZNFS}/${ZGWHN}
#
# ============================================================================
# Define the zone1 zone specific variables.
# ============================================================================
#
#
#
Z1HN=zone1
VNIC1IP=192.168.101.1
ZPATHZONE1=/${MZNFS}/${Z1HN}
ZPATH1TMP=${CFG}/${Z1HN}
ZPATH1SYSIDCFG=/${MZNFS}/${Z1HN}/${SYSIDCFG}
```

```
ZONE1BASEPATH=/${MZNFS}/${Z1HN}
#
# ============================================================================
# Define the zone2 zone specific variables.
# ============================================================================
#
Z2HN=zone2
VNIC2IP=192.168.102.2
ZPATHZONE2=/${MZNFS}/${Z2HN}
ZPATH2TMP=${CFG}/${Z2HN}
ZPATH2SYSIDCFG=/${MZNFS}/${Z2HN}/${SYSIDCFG}
ZONE2BASEPATH=/${MZNFS}/${Z2HN}
#
# ============================================================================
# Define the zone3 zone specific variables.
# ============================================================================
#
Z3HN=zone3
VNIC3IP=192.168.102.3
ZPATHZONE3=/${MZNFS}/${Z3HN}
ZPATH3TMP=${CFG}/${Z3HN}
ZPATH3SYSIDCFG=/${MZNFS}/${Z3HN}/${SYSIDCFG}
ZONE3BASEPATH=/${MZNFS}/${Z3HN}
# ============================================================================
# Define the zone4 zone specific variables.
# ============================================================================
#
Z4HN=zone4
VNIC4IP=192.168.101.4
ZPATHZONE4=/${MZNFS}/${Z4HN}
ZPATH4TMP=${CFG}/${Z4HN}
ZPATH4SYSIDCFG=/${MZNFS}/${Z4HN}/${SYSIDCFG}
ZONE4BASEPATH=/${MZNFS}/${Z4HN}
#
```

For More Information

Here are some Project Crossbow resources:

- Project Crossbow web site: http://hub.opensolaris.org/bin/view/Project+crossbow/WebHome
- "Networking in Solaris 10" on opensolaris.org: http://hub.opensolaris.org/bin/view/Community+Group+networking/WebHome
- "Upcoming Solaris Features: Crossbow - Part 1: Virtualisation" on c0t0d0s0.org: http://www.c0t0d0s0.org/archives/5355-Upcoming-Solaris-Features-Crossbow-Part-1-Virtualisation.html

Here are additional resources.

- Sun download site: http://www.sun.com/download/
- Oracle University web site: http://www.sun.com/training/
- Discussions, such as Sun forums (http://forums.sun.com/index.jspa) and the BigAdmin Discussions collection (http://www.sun.com/bigadmin/discussions/)
- Sun product documentation at http://docs.sun.com and the Sun Documentation Center (http://www.sun.com/documentation/)
- Sun wikis, such as the Sun BluePrints wiki (http://wikis.sun.com/display/BluePrints/Main)and the BigAdmin wiki (http://wikis.sun.com/display/BigAdmin/Home)
- Support:
  - Sun resources:
    - Register your gear: https://inventory.sun.com/inventory/
    - Sun Services: http://www.sun.com/service/index.jsp
    - SunSolve Online: http://sunsolve.sun.com
  - Community system administration experts: http://www.sun.com/bigadmin/content/communityexperts/

Oracle is committed to developing practices and products that help protect the environment

Setting Up a Virtual Network Automatically
Using Oracle Solaris 11 Express Project Crossbow
December 2010
Author: Aklilu Hadish

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

0109