iSCSI High Availability Clustering Using DRBD and Pacemaker on RHEL 8

Brian Hellman, Hayley Swimelar, Matt Kereczman, David Thomas
Version 2.0, 2024-02-01

Table of Contents

Chapter 1. Introduction

This guide outlines the configuration of a highly-available iSCSI storage cluster using DRBD®, Pacemaker, and Corosync. iSCSI provides access to block devices via TCP/IP, which allows storage to be accessed remotely using standard networks.

An iSCSI initiator (client) connects to an iSCSI target (server) and accesses a Logical Unit Number (LUN). Once a LUN is connected, it will function as a normal block device to the client.

This guide was written for RHEL 8, using the cluster stack software as packaged by LINBIT®.

Chapter 2. Assumptions

This guide assumes the following:

2.1. System Configurations

Hostname	LVM Device	Volume Group	Logical Volume	External Interface	External IP	Crossover Interface	Crossover IP
node-a	/dev/sdb	vg_drbd	lv_r0	eth0	192.168.10.201	eth1	172.16.0.201
node-b	/dev/sdb	vg_drbd	lv_r0	eth0	192.168.10.202	eth1	172.16.0.202

You will need a virtual IP for services to run on. For this guide you will use 192.168.10.200

2.2. Firewall Configuration

Refer to your firewall documentation for how to open or allow ports. You will need the following ports open in order for your cluster to function properly.

Component	Protocol	Port
DRBD	TCP	7788
Corosync	UDP	5404, 5405

2.3. SELinux

If you have SELinux enabled, and you are having issues, consult your distribution’s documentation for how to properly configure it, or disable it (not recommended).

Chapter 3. Installation and Configuration

3.1. Registering Nodes and Configuring Package Repositories

You will install DRBD and other cluster stack software that you may need from LINBIT's repositories. To access those repositories you will need to have been set up in LINBIT's system and have access to the LINBIT Customer Portal. If you have not been set up in LINBIT's system, please contact a sales team member: sales@linbit.com.

Once you have access to the Customer Portal, you can register your cluster nodes and configure repository access by using LINBIT's Python command line script. See the "REGISTER NODES" section of the Customer Portal for details about this script.

To download and run the LINBIT configuration script, enter the following commands on all nodes, one node at a time:

```
# curl -O https://my.linbit.com/linbit-manage-node.py
# chmod +x ./linbit-manage-node.py
# ./linbit-manage-node.py
```

Script must be run as superuser.

If the error message `no python interpreter found :-(` is displayed when running `linbit-manage-node.py`, enter the command `dnf install python3` to install Python 3.

The script will prompt you to enter your LINBIT Customer Portal username and password. After validating your credentials, the script will list clusters and nodes (if you have any already registered) that are associated with your account.

Enter which cluster you want to register the current node with. You will then be asked a series of questions regarding which repositories you want to enable.

```
    1) pacemaker-2(Disabled)
    2) drbd-proxy-3.2(Disabled)
    3) drbd-9.0-only(Disabled)
    4) drbd-9.0(Disabled)
    5) drbd-9(Disabled)
```

The "drbd-9" repository includes the latest version, presently 9.1.x. As DRBD 9 is the latest version, it contains newer packages than the other two DRBD repositories. The "drbd-9.0" repository holds 9.0.x packages, for those who want to remain on that branch. The "drbd-9.0-only" contains the DRBD packages only (no LINSTOR).

Be sure to respond **yes** to the questions about installing LINBIT's public key to your keyring and writing the repository configuration file.

After the script completes, you should be able to enter `dnf info kmod-drbd` and see the DNF package manager pulling package information from LINBIT repositories.

Before installing packages, be sure to only pull the cluster stack packages from LINBIT repositories.

3.1.1. Excluding Packages from Mainstream Repositories

To ensure that you only pull cluster packages from LINBIT repositories, add the following exclude line to the system package repository files:

```
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*
```

The x86_64 architecture repositories are used in the following examples. Adjust appropriately if your system architecture is different.

3.1.2. Excluding Packages from RHEL 8 Repositories

The default location for all repositories in RHEL 8 is `/etc/yum.repos.d/redhat.repo`. Add the exclude line to both the `[rhel-8-for-x86_64-baseos-rpms]` and `[rhel-8-for-x86_64-appstream-rpms]` repository sections within the `.repo` file. The modified repository configuration should look like this:

```
# cat /etc/yum.repos.d/redhat.repo

[rhel-8-for-x86_64-baseos-rpms]
name = Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)
...
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*

[rhel-8-for-x86_64-appstream-rpms]
name = Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
...
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*
```

If the *Red Hat High Availability Add-On* is enabled, either add the exclude line to the `[rhel-8-for-x86_64-highavailability-rpms]` section or consider disabling the repository. LINBIT provides most of the packages available in the HA repository.

3.1.3. Excluding Packages from CentOS 8 Repositories

Add the exclude line to both the `[BaseOS]` section of `/etc/yum.repos.d/CentOS-Base.repo` as well as the `[AppStream]` section of `/etc/yum.repos.d/CentOS-AppStream.repo` repository files. The modified repository configuration should look like this:

```
# cat /etc/yum.repos.d/CentOS-Base.repo

[BaseOS]
name=CentOS-$releasever - Base
...
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*
```

```
# cat /etc/yum.repos.d/CentOS-AppStream.repo

[AppStream]
name=CentOS-$releasever - AppStream
...
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*
```

If the `[HighAvailability]` repo is enabled in `/etc/yum.repos.d/CentOS-HA.repo`, add the exclude line to the `[HighAvailability]` section or else consider disabling the repository. LINBIT provides most of the packages available in the HA repository.

3.1.4. Excluding Packages from AlmaLinux 8 Repositories

The default location for all repositories in AlmaLinux 8 is `/etc/yum.repos.d/almalinux.repo`. Add the exclude line to both the `[baseos]` and `[appstream]` sections of this file.

```
# cat /etc/yum.repos.d/almalinux.repo

[baseos]
name=AlmaLinux $releasever - BaseOS
...
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*

[appstream]
name=AlmaLinux $releasever - AppStream
...
exclude=cluster* corosync* drbd kmod-drbd libqb* pacemaker* resource-agents*
...
```

If the `[HighAvailability]` repo is enabled, `enabled=1`, in `/etc/yum.repos.d/almalinux-ha.repo`, add the exclude line to the `[ha]` section or else consider disabling the repository. LINBIT provides most of the packages available in the HA repository.

3.2. Install DRBD

Install DRBD and the DRBD kernel module by entering the following command:

```
# dnf install -y drbd kmod-drbd
```

Because Pacemaker will be responsible for starting the DRBD service, prevent DRBD from starting at boot:

```
# systemctl disable drbd
```

3.3. Configure DRBD

Now that you have installed DRBD, you will need to create your resource configuration file. To do this, create `/etc/drbd.d/r0.res` with the following contents:

```
resource r0 {
        protocol C;
        device    /dev/drbd0;
        disk      /dev/vg_drbd/lv_r0;
        meta-disk internal;
        on node-a {
            address 172.16.0.201:7788;
        }
        on node-b {
            address 172.16.0.202:7788;
        }
}
```

This is a minimally configured resource configuration file. There are many settings you can add and adjust to increase performance. See the *DRBD User Guide* for more information.

Create the resource metadata by issuing the following command:

```
# drbdadm create-md r0
initializing activity log
initializing bitmap (32 KB) to all zero
Writing meta data...
New drbd meta data block successfully created.
success
```

This command should complete without any warnings - if you get messages about data being detected, and choose to proceed, you will lose data.

If you get any error messages when running the above `drbdadm` command, you can verify your DRBD configuration by entering the following command: `drbdadm dump all`.

Bring the device up on both nodes and verify their states by entering the following commands:

```
# drbdadm up r0
# drbdadm status
r0 role:Secondary
  disk:Inconsistent
  node-b role:Secondary
    peer-disk:Inconsistent
```

As the preceding output shows, the resource is connected, but in an inconsistent state. To have your data replicated, you will need to put the resource into a consistent state. There are two options:

1. Do a full sync of the device, which could potentially take a long time depending upon the size of the disk.
2. Skip the initial sync.

Because you know this is a new setup with "just created" metadata and without any existing data on your device, you can use the second option and skip the initial sync. Enter the following commands on **node-a**:

```
# drbdadm new-current-uuid --clear-bitmap r0/0
# drbdadm status
r0 role:Secondary
  disk:UpToDate
  node-b role:Secondary
    peer-disk:UpToDate
```

This section has instructions for installing and configuring the necessary components to achieve a highly-available iSCSI storage cluster. Some of the components that you will install need to come from LINBIT software package repositories. The instructions will guide you on how to register your cluster nodes with LINBIT to access these repositories.

To successfully install the required LINBIT software packages, you will need to enable the LINBIT `pacemaker-2` and the `drbd-9` LINBIT repositories, when prompted by the LINBIT manage node script on both of your nodes.

3.4. Install Pacemaker and Corosync

This section will cover installing Pacemaker and Corosync. You will use Pacemaker as your cluster resource manager (CRM). Corosync acts as a messaging layer, providing information to Pacemaker about the state of your cluster nodes.

Enter the following commands to install the necessary packages and then enable the Pacemaker and Corosync services to start when the system boots:

```
# dnf install -y pacemaker corosync crmsh

# systemctl enable pacemaker
Created symlink /etc/systemd/system/multi-user.target.wants/pacemaker.service to
/usr/lib/systemd/system/pacemaker.service.

# systemctl enable corosync
Created symlink /etc/systemd/system/multi-user.target.wants/corosync.service to
/usr/lib/systemd/system/corosync.service.
```

3.4.1. Configure Corosync

Create and edit the file `/etc/corosync/corosync.conf`. It should look like this:

```
totem {
  version: 2
  secauth: off
  cluster_name: cluster
  transport: knet
  rrp_mode: passive
}

nodelist {
  node {
    ring0_addr: 172.16.0.201
    ring1_addr: 192.168.10.201
    nodeid: 1
    name: node-a
  }
  node {
    ring0_addr: 172.16.0.202
    ring1_addr: 192.168.10.202
    nodeid: 2
    name: node-b
  }
}

quorum {
  provider: corosync_votequorum
  two_node: 1
}

logging {
  to_syslog: yes
}
```

Now that Corosync has been configured, start the Corosync and Pacemaker services:

```
# systemctl start corosync
# systemctl start pacemaker
```

Repeat the preceding Pacemaker and Corosync installation and configuration steps on each cluster node. Verify that everything has been started and is working correctly by entering the following `crm_mon` command. You should get output similar to this:

```
# crm_mon -rf -n1
Cluster Summary:
    * Stack: corosync
    * Current DC: node-a (version 2.0.5.linbit-1.0.el8-ba59be712) - partition with quorum
    * Last updated: Mon Feb 14 00:03:44 2022
    * Last change: Sun Feb 13 04:18:45 2022 by root crmd on node-a
    * 2 nodes configured

Node List:
    * Online: [ node-a node-b ]

Inactive resources:

Migration Summary:
    * Node node-a:
    * Node node-b:
```

3.5. Additional Cluster-Level Configuration

Since you only have two nodes, you will need to tell Pacemaker to ignore quorum. Run the following commands from either cluster node (but not both):

```
# crm configure property no-quorum-policy=ignore
```

Furthermore, for simplicity, you will not be configuring node level fencing (aka STONITH) in this guide. Disable STONITH using the following command:

```
# crm configure property stonith-enabled=false
```

Fencing/STONITH is an important part of HA clustering and should be used whenever possible. Disabling STONITH will lend the cluster vulnerable to split-brains and potential data corruption or loss.

For more information on Fencing and STONITH, you can review the ClusterLabs page on STONITH or contact the experts at LINBIT.

3.6. Selecting and Installing the iSCSI Stack and Dependencies

There is more than one iSCSI implementation you can choose from when building your highly-available iSCSI cluster. This guide will describe two iSCSI implementations: Linux-IO (LIO) and the Generic SCSI Target Subsystem for Linux (SCST). Both LIO and SCST operate within kernel space. They also both support iSCSI persistent reservations, which is an important feature when attaching highly-available initiators such as hypervisor clusters.

LIO is the iSCSI implementation that the Linux kernel community chose as the SCSI target subsystem that is included in the mainline Linux kernel.

SCST is a mature iSCSI implementation that is used in many iSCSI appliances, including LINBIT's VSAN since version 0.17.0.

3.6.1. Option A: LIO

TargetCLI is an interactive shell used to manage the Linux-IO (LIO) target. LIO is the SCSI target that has been included with the Linux kernel since 2.6.38, which makes the installation of its utilities and dependencies relatively simple.

TargetCLI can be installed using the following command on both nodes:

```
# dnf install targetcli
```

3.6.2. Option B: SCST

The SCST project consists of a kernel space core, device handlers, target drivers, and the `scstadmin` user space utility for managing its core components. All of which can be built from source, by following the instructions found on the project's GitHub repository.

By following the installation instructions below, you will install all the necessary components for building and using SCST with Pacemaker.

Enter and run all the commands in this section on both nodes.

ELRepo, the RPM repository for Enterprise Linux packages that are not included in the standard RHEL distribution's repositories, is needed for installing DKMS. Development tools and other dependencies for building SCST's RPMs must also be installed:

```
# dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
# dnf groupinstall -y "Development Tools"
# dnf install -y kernel-devel perl perl-Data-Dumper perl-ExtUtils-MakeMaker rpm-build dkms git
```

After installing build dependencies, you can build and install the SCST packages:

```
# git clone https://github.com/SCST-project/scst
# cd scst/
# make rpm-dkms
# cd ~/
# dnf install -y /usr/src/packages/RPMS/x86_64/scst*
```

Finally, enter the commands below to create the necessary configuration for loading the SCST kernel module, create a systemd unit file for an `iscs-scst` service, and reload systemd services:

```
# echo -n "" > /etc/modules-load.d/scst.conf
# for m in iscsi-scst scst scst_vdisk; do
    echo $m >> /etc/modules-load.d/scst.conf
    modprobe $m
  done

# cat << EOF > /etc/systemd/system/iscsi-scst.service
[Unit]
Description=iSCSI SCST Target Daemon
Documentation=man:iscsi-scstd(8)
After=network.target
Before=scst.service
Conflicts=shutdown.target

[Service]
EnvironmentFile=-/etc/sysconfig/scst
PIDFile=/var/run/iscsi-scstd.pid
ExecStartPre=/sbin/modprobe iscsi-scst
ExecStart=/sbin/iscsi-scstd $ISCSID_OPTIONS

[Install]
WantedBy=multi-user.target
```

```
EOF

# systemctl daemon-reload
# systemctl enable --now iscsi-scst
```

3.7. Creating an Active/Passive iSCSI Configuration

An active/passive iSCSI Target consists of the following cluster resources:

- A DRBD resource to replicate data, which is switched from and to the Primary and Secondary roles as deemed necessary by the cluster resource manager.
- A virtual, floating cluster IP address, allowing initiators to connect to the target no matter which physical node it is running on.
- The iSCSI Target itself.
- Two port blocking rules to prevent connection errors while the target is being brought up or down: one to block the iSCSI target and another to unblock it.
- One or more iSCSI Logical Units (LUs), each corresponding to a Logical Volume in the LVM Volume Group.

The following Pacemaker configuration example assumes that 192.168.10.200 is the virtual IP address to use for a target with the iSCSI Qualified Name (IQN) iqn.2024-01.com.example:drbd0.

The target is to contain one Logical Unit: (LUN 0) mapping to /dev/drbd0.

To start configuring these resources, open the crm shell at the configuration prompt as root (or any non-root user that is part of the haclient group), by issuing the following command:

You only need to run the `crm configure` command and enter the configurations shown below on one of your cluster nodes.

```
# crm configure
```

Now create the DRBD resource primitive by running the following commands:

```
crm(live)configure# primitive p_drbd_r0 ocf:linbit:drbd \
  params drbd_resource="r0" \
  op start timeout=240 \
  op promote timeout=90 \
  op demote timeout=90 \
  op stop timeout=100 \
  op monitor interval="29" role="Master" \
  op monitor interval="31" role="Slave"
```

Running verify after configuring each new Pacemaker resource is recommended: it will alert you if you have syntax errors in your configuration. To edit the configuration in a text editor run edit with an optional resource ID.

Next, create a multi-state resource corresponding to the DRBD resource `r0`:

```
crm(live)configure# ms ms_drbd_r0 p_drbd_r0 \
  meta master-max=1 master-node-max=1 \
  notify=true clone-max=2 clone-node-max=1
```

We will now create a floating Virtual IP for the cluster to be managed by Pacemaker:

```
crm(live)configure# primitive p_iscsi_ip0 ocf:heartbeat:IPaddr2 \
  params ip="192.168.10.200" cidr_netmask="24" \
  op start timeout=20 \
  op stop timeout=20 \
  op monitor interval="10s"
```

iSCSI IQN names must be unique and should follow RFC 3720 standards.

```
crm(live)configure# primitive p_iscsi_target_drbd0 ocf:heartbeat:iSCSITarget \
  params iqn="iqn.2024-01.com.example:drbd0" \
  portals="192.168.10.200:3260" \
  op start timeout=20 \
  op stop timeout=20 \
  op monitor interval=20 timeout=40
```

Now that the iSCSI target and virtual IP are established, we will configure the Logical Units:

```
crm(live)configure# primitive p_iscsi_lun_drbd0 ocf:heartbeat:iSCSILogicalUnit \
  params target_iqn="iqn.2024-01.com.example:drbd0" \
  lun=0 path="/dev/drbd0" \
  op start timeout=20 \
  op stop timeout=20 \
  op monitor interval=20 timeout=40
```

Both the `iSCSITarget` and `iSCSILogicalUnit` have an `implementation` parameter. Omitting the parameter causes the resource agents to determine which implementation to use based on the presence of implementation specific binaries. If you have multiple implementations installed, or simply want to be explicit in your resource definitions, you can set the `implementation` parameter equal to `scst` or `lio-t`.

Configure port blocking and unblocking. This will prevent an iSCSI initiator from receiving a "Connection refused" error during failover before the iSCSI target has successfully started.

```
crm(live)configure# primitive p_iscsi_portblock_on_drbd0 ocf:heartbeat:portblock \
  params ip=192.168.10.200 portno=3260 protocol=tcp action=block \
  op start timeout=20 \
  op stop timeout=20 \
  op monitor timeout=20 interval=20

crm(live)configure# primitive p_iscsi_portblock_off_drbd0 ocf:heartbeat:portblock \
  params ip=192.168.10.200 portno=3260 protocol=tcp action=unblock \
  op start timeout=20 \
  op stop timeout=20 \
  op monitor timeout=20 interval=20
```

To tie all of this together, create a resource group from the resources associated with our iSCSI Target:

```
crm(live)configure# group g_iscsi_drbd0 \
  p_iscsi_portblock_on_drbd0 \
  p_iscsi_ip0 p_iscsi_target_drbd0 p_iscsi_lun_drbd0 \
  p_iscsi_portblock_off_drbd0
```

This group, by default, is *ordered* and *colocated*, which means that the resources contained therein will always run on the same physical node, will be started in the order as specified, and stopped in reverse order.

Finally, we have to make sure that this resource group is started after the DRBD resource has started and set it to run on the same node where DRBD is in the Primary role:

```
crm(live)configure# colocation cl_g_iscsi_drbd0-with-ms_drbd_r0 \
  inf: g_iscsi_drbd0:Started ms_drbd_r0:Master

crm(live)configure# order o_ms_drbd_r0-before-g_iscsi_drbd0 \
  ms_drbd_r0:promote g_iscsi_drbd0:start
```

Now, our configuration is complete, and can be activated:

```
crm(live)configure# commit
```

3.8. Creating an Active/Active iSCSI Configuration

Configuring an active/active iSCSI Target is similar to an active/passive target, with the exception of the following:

- There must be at least two DRBD resources to replicate data, which can run independently on separate nodes.
- One LVM Logical Volume to serve as the backing device for each DRBD resource.
- Each target must have its own floating cluster (virtual) IP address, allowing initiators to connect to the target no matter which physical node the target is running on.
- Each DRBD resource needs to have a resource group, plus order and colocation constraints similar to the single DRBD resource in the active/passive section.

Once these resource groups are configured, you can have them run on separate nodes to achieve an active/active cluster. This can be done by manually migrating the resource groups through the CRM shell, or by configuring the resources to prefer running on a particular node.

Chapter 4. Security Considerations and Data Integrity

There are a few more points worth mentioning, dealing with access control and data integrity.

4.1. Restricting Target Access by Initiator Address

Access to iSCSI targets should be restricted to specific initiators, identified by their iSCSI Qualified Name (IQN). Use the allowed_initiators parameter supported by the iSCSI Target Pacemaker resource agent.

Create a resource to include the allowed_initiators parameter, containing a space-separated list of initiator IQNs allowed to connect to this target. In the example below,access is granted to the initiator IQN iqn.1994-05.com.example:5e6220f26ee.

On a RHEL system, where the `iscsi-initiator-utils` package has been installed, you can find the initiator's IQN by entering:

```
cat /etc/iscsi/initiatorname.iscsi
```

```
primitive p_iscsi_target_drbd0 \
  ocf:heartbeat:iSCSITarget \
  params iqn="iqn.2024-01.com.example:drbd0" \
  allowed_initiators="iqn.1994-05.com.redhat:42615a3677 iqn.1994-05.com.redhat:27cac47b0ae"  \
  op start timeout=20 \
  op stop timeout=20 \
  op monitor interval="10s"
```

When you close the editor, the configuration changes are inserted into the CIB configuration. To commit these changes, enter the following command:

```
crm(live)configure# commit
```

After you commit the changes, the target will immediately reconfigure and enable the access restrictions.

If initiators are connected to the target at the time of re-configuration, and one of the connected initiators is not included in the initiators list for this resource, then those initiators will lose access to the target, possibly resulting in disruption on the initiator node. Use with care.

4.2. Dual-Primary Multipath iSCSI

Do not attempt to use Multipath iSCSI targets with dual-primary DRBD resources. Multipath iSCSI targets do not coordinate with each other and are not cluster aware.

Multipath iSCSI might appear to work with dual-primary DRBD resources under light testing, but will not be able to maintain consistent data under production loads or in the case of some failures, such as the loss of the replication link.

Chapter 5. Using Highly Available iSCSI Targets

This section describes some common usage scenarios for highly available iSCSI Targets.

5.1. Connecting to iSCSI Targets From Linux

The recommended way of connecting to a highly available iSCSI Target from Linux is to use iscsiadm which can be installed by running the following command:

```
# dnf install -y iscsi-initiator-utils
```

After installing the iscsi-initiator-utils package, it is first necessary to start the iSCSI service, iscsi. To do so, issue the following command:

```
# systemctl start iscsi
```

Now you can start a discovery session on your target portal. Assuming your cluster IP address for the target is 192.168.10.200, you can do so via the following command:

```
# iscsiadm -m discovery -p 192.168.10.200 -t sendtargets
```

The output from this command should include the names of all targets you have configured.

```
192.168.10.200:3260,1 iqn.2017-10.com.example:drbd0
```

If a configured target does not appear in this list, check whether your initiator has been blocked from accessing this target via an initiator restriction (see Restricting Target Access by Initiator Address).

Finally, you can log in to the target, which will make all LUNs configured therein available as local SCSI devices:

```
# iscsiadm -m node -p 192.168.10.200 -T iqn.2024-01.com.example:drbd0 --login
```

Chapter 6. Feedback

Any questions or comments about this document are appreciated and encouraged.

For a public discussion about the concepts mentioned in this how-to guide, you are invited to subscribe and post to the drbd-user mailing list. See the drbd-user mailing list for details.

Appendix A: Additional Information and Resources

- LINBIT's GitHub Organization: https://github.com/LINBIT/
- Join the LINBIT Community Forums: https://forums.linbit.com
- The DRBD® and LINSTOR® User Guides: https://linbit.com/user-guides-and-product-documentation/
- The DRBD® and LINSTOR® Mailing Lists: https://lists.linbit.com/
  - drbd-announce: Announcements of new releases and critical bugs found
  - drbd-user: General discussion and community support
  - drbd-dev: Coordination of development

Appendix B: Legalese

B.1. Trademark Notice

B.2. License Information