White Paper

Title: Implementing STONITH Support in LifeKeeper for Linux Clusters

Revision: B

Date: December 2, 2002

Implementing STONITH Support in LifeKeeper for Linux Clusters

Table of Contents

1. Overview

High availability clusters generally require the use of some form of I/O fencing, a technique for ensuring that only the appropriate system in a cluster has access to a given volume of storage. LifeKeeper for Linux uses SCSI reservations as its I/O fencing mechanism in a shared storage cluster.

However, many clustering products, particularly those for Linux, have adopted the use of STONITH (Shoot The Other Node In The Head) devices for I/O fencing. This technique uses an external smart power switch to control the power to the nodes in the cluster and allows the cluster software to instruct the switch via a serial or network connection to power off or reboot a cluster node that appears to have died, thus ensuring that the unhealthy node cannot access or corrupt any shared data.

While we believe that SCSI reservations are a superior technique, particularly for clusters beyond the simple two-node active/passive configuration, there may be situations in which it would be beneficial for LifeKeeper to support the use of STONITH devices as an alternative fencing mechanism. This paper describes an interface for adding STONITH devices to a LifeKeeper cluster. It begins with some background on LifeKeeper's event notification mechanism, which is the framework where the commands to control the smart power switch will be placed. General instructions for implementing and configuring STONITH devices in a LifeKeeper cluster will be given, followed by a specific example using Western Telematic Inc. (WTI) RPS-10 switches in a two-node cluster.

2. Audience

The information in this paper is intended for those who will be installing and configuring the hardware and software in a LifeKeeper cluster.

3. LifeKeeper Event Notification

When LifeKeeper detects failure conditions in a cluster, it provides notification using an event alarming mechanism based on **sendevent**. The framework allows applications to register programs or scripts so that they can receive notification of alarm events, and/or perform recovery or error handling actions. The directory tree under *$LKROOT/events* corresponds to a set of alarm classes with subdirectories representing events within each class (the default value of *$LKROOT* is */opt/LifeKeeper*). When a particular event is detected by LifeKeeper, a **sendevent** command is issued, specifying the particular class and event, along with any other relevant data. The **sendevent** mechanism locates the directory that represents the class and event, and executes any scripts or programs that are in that directory.

For example, the *$LKROOT/events/filesys* directory represents the class of alarms associated with file systems. It may contain subdirectories for each event in that class (badmount, diskfull, noaccess). If one of these events occurs, any script or program in the event directory will run. The "notify" script in the *$LKROOT/events/filesys/diskfull* directory provides an example of how to send email to alert an administrator that a filesystem is reaching full capacity.

More detailed information about LifeKeeper's event notification mechanism can be found in the LifeKeeper online help under the topic Advanced Tasks - LifeKeeper Communications - LifeKeeper Alarming and Recovery. The LifeKeeper **sendevent(5)** man page also documents details of the alarming mechanism.

The *$LKROOT/events/prefailover* class is installed to facilitate node-specific handling of the failure of a system in a LifeKeeper cluster. This class is triggered early in the failure detection process, before any resource recovery begins. The subdirectories or "events" in this class correspond to the remote systems in the LifeKeeper cluster. Each directory name is identical to the name of a remote system in the cluster, and must be manually created to reflect the specific cluster under protection. Scripts or programs may be placed in each directory to perform actions corresponding to the failure of that particular node.

4. Using the "prefailover" Class to Implement STONITH Operations

The prefailover class of alarm is triggered when a node in the cluster detects a complete communication failure with another node in the cluster. If a subdirectory corresponding to the name of the failed node is present in the *$LKROOT/events/prefailover* directory, the alarm mechanism will execute any scripts or programs that reside in the failed node's directory. A script or program can be implemented to send a message to a smart power switch that controls the failed node, to instruct it to reboot or remove power from that node. This prevents the impaired node from corrupting data.

For example, in a two-node cluster where one system is named "node1" and the other is named "node2" the prefailover class directories would be configured as follows:

- On node1, the directory *$LKROOT/events/prefailover/node2* will be created.
- On node2, the directory *$LKROOT/events/prefailover/node1* will be created.

The directory name must exactly match the system name as known to LifeKeeper (which is what is given by the output of **uname -n**). The recommended owner, group, and permissions for the directory are "bin" "bin" and 775.

Each node will be connected to a smart power switch that controls the opposite node. A script or program will be put in *$LKROOT/events/prefailover/<nodeX>* that will send the appropriate message to the controlled node if a failover is detected. The recommended owner, group, and permissions for the scripts are "root" "root" and 0500. The name of the script does not matter to LifeKeeper.

5. An Example Using the WTI RPS-10

The WTI RPS-10 (http://www.wti.com/rps-10.htm) is a remote reboot device that can be used to control AC power to various network devices. A master module can control up to 10 power outlets, and may be used in conjunction with multiple "satellite" modules. For our example, we will be using two master modules in a two-node cluster. Each system will have its power cord plugged into a master RPS-10, and a serial connection to the opposite node's RPS-10. This allows each node to control the power to the other.

The following example script can be used to send a "poweroff" command to the RPS-10.

```
while [ $# != 0 ]
do
      case "$1" in
      -n)
               shift
               MACH=$1
               ;;
      -n*)
               MACH=`echo "$1" | sed "s/^-n//"`
               ;;
      esac
      shift
done

if [ "$MACH" = "" ]
then
      DIR=`dirname $0`
      MACH=`basename $DIR`
fi

echo "STONITH:  class prefailover, removing power from $MACH at `date`"

stty -F /dev/ttyS0 1:0:8bd:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:
:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0

echo ATDT >/dev/ttyS0
MSG="^B^X^X^B^X^X00^M"
echo $MSG >/dev/ttyS0

if [ $? -ne 0 ]
then
      echo "STONITH operation failed."
      exit 1
fi
exit 0
```

When sendevent invokes this script, it supplies the name of the failed node in the -n option on the command line. Any output from the script to stdout or stderr will be automatically directed to the main LifeKeeper log.

This script uses "stty" to initialize the RPS-10 and then sends the specific control string to the serial connection (/dev/ttyS0) to remove power from the failed node.

An alternative implementation is to use the **stonith** utility supplied by the open-source HA "heartbeat" project (http://www.linux-ha.org/heartbeat). This utility implements support for various smart power switches. It is delivered in a separate standalone package, and has a command line interface. Here is an example script using this approach (the **stonith** command performs a reboot of the controlled system rather than a poweroff).

```
while [ $# != 0 ]
do
      case "$1" in
      -n)
               shift
               MACH=$1
               ;;
      -n*)
               MACH=`echo "$1" | sed "s/^-n//"`
               ;;
      esac
      shift
done

if [ "$MACH" = "" ]
then

      DIR=`dirname $0`
      MACH=`basename $DIR`
fi

echo "STONITH:  class prefailover, removing power from $MACH at `date`"
stonith -t rps10 -p "/dev/ttyS0 $MACH 0" $MACH

if [ $? -ne 0 ]
then
      echo "STONITH operation failed."
      exit 1
fi
exit 0
```

6. Conclusion

The "prefailover" alarm class in LifeKeeper can be used to detect failure scenarios where it may be appropriate to initiate STONITH operations in a LifeKeeper cluster. Directories named after remote systems in the cluster can be created in the *$LKROOT/events/prefailover* directory, and a script or program to operate a smart power switch can reboot or power off a node that has failed. The exact implementation of this script or program depends on the particular device that is being used, and is outside the scope of the LifeKeeper product. Sample scripts for the WTI RPS-10 example described in this paper can be found on the SteelEye web site (http://www.steeleye.com/) in the Download area on the "Optional LifeKeeper Extensions" page.