SIOS Protection Suite for Linux v9.2.2

Chef Support Document

March 2018

To maintain the quality of our publications, we welcome your comments on the accuracy, clarity, organization, and value of this document.

Address correspondence to:
ip@us.sios.com

Table of Contents

1. Introduction

Starting with v9.0.0 LifeKeeper for Linux provides support for the IT automation tool Chef. The addition of this support in LifeKeeper for Linux (here after referred to as LifeKeeper) allows the easy transfer of resource hierarchy information from a staging cluster to a production cluster. This document includes requirements and basic operations to reconstruct existing LifeKeeper configurations using Chef.

This document assumes you have appropriate knowledge of both LifeKeeper and Chef. Basic configurations and information on detailed technical matters are not included. For more information on Chef use the following link:

https://www.chef.io/chef/

2. Overview

2-1 What is Chef?

Managing servers has traditionally been performed using a manual set of procedures to perform the required tasks. Chef is a tool that allows you to take these manual procedures and automate them into repeatable processes.. This is sometimes referred to as Infrastructure as Code.

2-2 Benefits of Chef Support

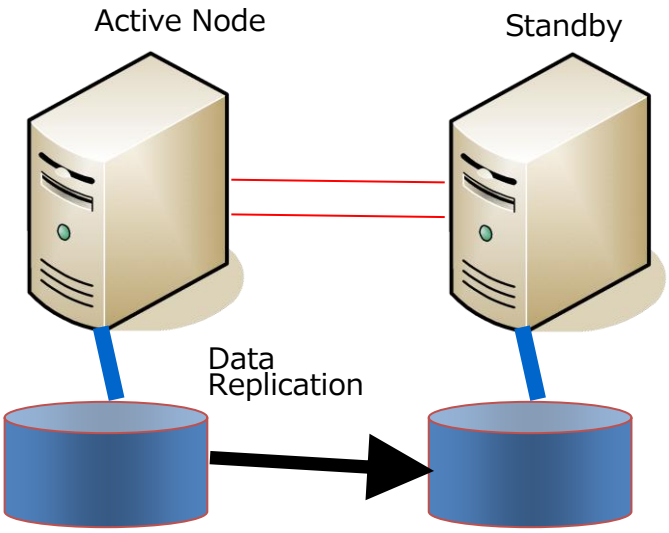
The benefits to supporting Chef in LifeKeeper are as follows:

- The ability to extract Chef attributes from existing clusters makes cluster replication easier.
- Engineering burdens are reduced through automation of existing tasks to create LifeKeeper clusters.
- Error reduction via Chef automation of manual tasks.
- Automation of hardware replacement.Easy transfer from staging to production.

2-3 Supported Operating Systems and LifeKeeper Recovery Kits

The Operating Systems and LifeKeeper Recovery Kits supported via Chef automation are:

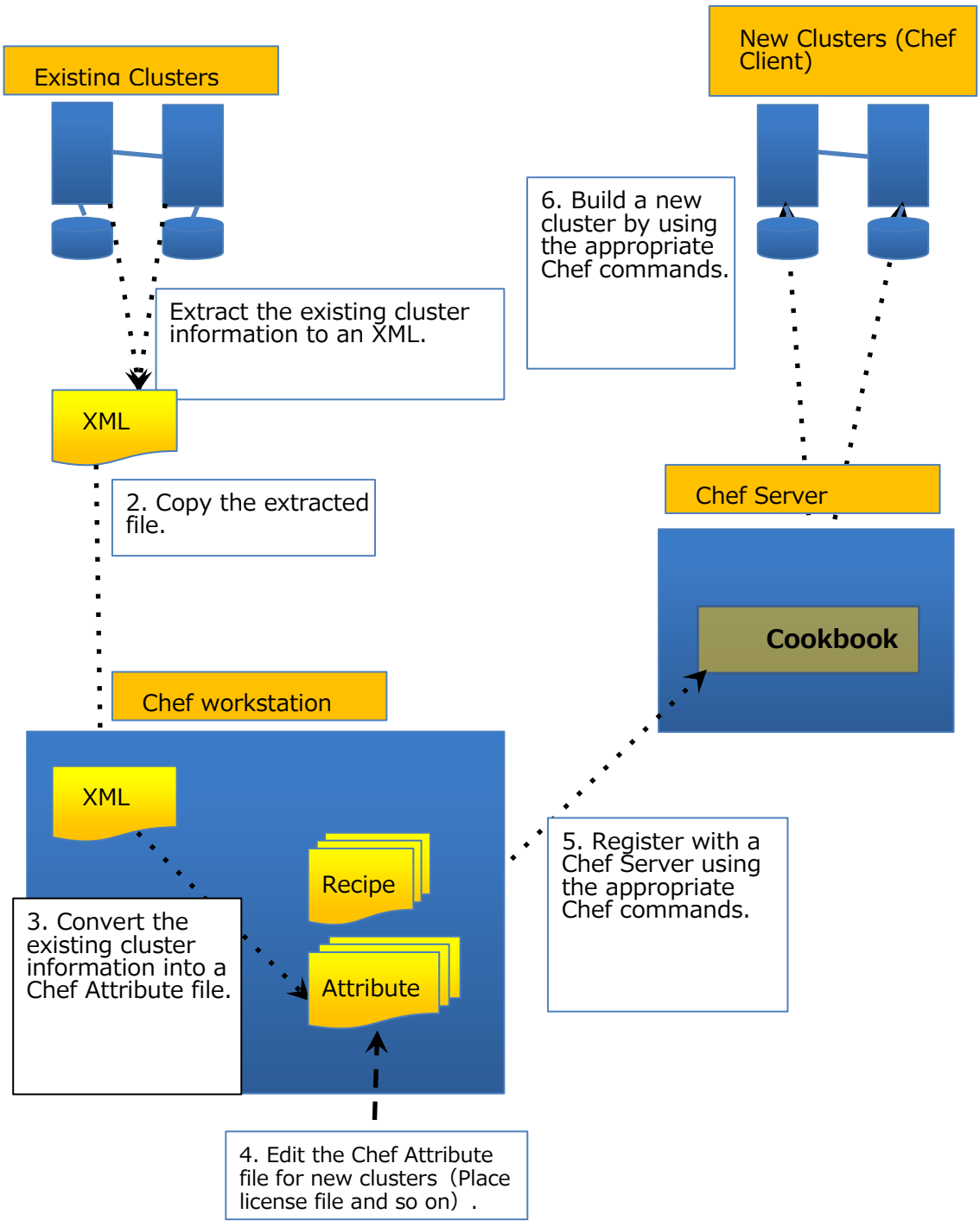
- Operating Systems
  - ➢ Red Hat Enterprise Linux version 5, 6, and 7
  - ➢ Community ENTerprise Operating System (CentOS) version 5, 6, and 7
  - ➢ Oracle Linux version 5, 6, and 7
  - ➢ Note: The OS type and version is supported as both a LifeKeepper cluster node and as a Chef client.
- LifeKeeper Configuration
  - ➢ 2-node Data Replication configuration
  - ➢ Supported Recovery Kits in the 2-node Data Replication configuration:
    - ✧ IP
    - ✧ File System
    - ✧ Apache
    - ✧ MySQL
    - ✧ PostgreSQL



The LifeKeeper installation, communication path creation and resource hierarchy creation are the recipes and attributes for the LifeKeeper Chef cookbook that this release supports. The recipes and attributes will need to be combined into a Chef cookbook.

2-4 Processing Flow

The processing flow with system designations when using LifeKeeper Chef support.



2-5 Processing Flow Overview

1. Extract the existing cluster resource configuration into an xml formatted file.
2. Copy the extracted XML file to the Chef workstation.
3. Convert the extracted XML file into a Chef Parameter “attribute” file. Two “attributes” files will be generated. One for the LifeKeeper communication paths and one for the LifeKeeper resource hierarchies.
4. Copy the two attribute files to the attribute folder under the LifeKeeper cookbook directory on the Chef workstation. Modify the parameters for the embedded host names and IP addresses to match the new environment. Also, copy the rpms used for the installation and licensing of LifeKeeper under the LifeKeeper Chef cookbook directory.
5. Resister the files in the LifeKeeper cookbook directory on the Chef workstation with the Chef Server using the Chef Knife command.
6. Use the appropriate chef-client commands on each new node to construct a LifeKeeper cluster.

3. Chef Procedure Details

3-1 Environment Preparation

1. LifeKeeper HA cluster configuration.
   Setup the LifeKeeper HA cluster to be used to generate the LifeKeeper Chef cookbook. This includes LifeKeeper installation (core and recovery kits), creation of communication paths, and creation of the resource hierarchies. Check basic operations such as switchover and/or failover. LifeKeeper v9.0.0 and later are the only versions supported.

2. Prepare the Chef Server and/or the Chef Work station systems. See the Chef official documentation for configuring the Chef Server and the Chef Work station.
3. Place the Chef support file(recipe file) for LifeKeeper on the Chef workstation

   Mount the LifeKeeper installation image file on the Chef workstation and copy the attribute conversion script.
   See below for the detailed procedure.

   (1) Mount the LifeKeeper installation image file “sps.img” on /mnt.
   Example:

```
# mount sps.img -t iso9660 -o loop /mnt
```

   (2) Check for Chef support directory.
   Example:

```
$ ls /mnt/Chef/
attribute exp2chef.pl nodes recipe TRANS.TBL
```

(3) Copy the LifeKeeper Chef conversion script to the appropriate directory.

Example: Create and copy to the Chef Directory under ~/.

```
$ mkdir ~/Chef
$ cp /mnt/Chef/exp2chef.pl ~/Chef
```

(4) Set up an execution environment for the exp2chef.pl script.

To run the exp2chef.pl script, Perl5 and XML::Simple are required on the Chef workstation. Acquire and install them from the OS distribution image or CPAN.

For example on CentOS 6:

yum install perl-XML-Simple

3-2 Extract the Existing Cluster Information

Run the commands listed below to extract the existing cluster configuration information created in step 1 above. Save this information into a file (note the export command outputs information to stdout). Ensure all resources are In Service on the node before running the export script.

```
#/opt/LifeKeeper/lkadm/bin/lkexportxml
```

Note that if the LifeKeeper hierarchy contains resource types not listed as supported in section 2-3 of this document will result in an error when running lkexportxml.

Usage examples:

Assuming that the output file is resource.xml, and its directory is under ~/Chef.

```
# /opt/LifeKeeper/lkadm/bin/lkexportxml>~/Chef/resource.xml
```

Examples of output result

```
  <?xml version='1.0'?>
  <lifekeeper>
    <node name="node1">
      <commpath remote="node2">
        <baudrate>0</baudrate>

<device>192.168.100.1/192.168.100.2</device>
        <ipaddress>192.168.100.1</ipaddress>
        <priority>1</priority>

<remoteaddress>192.168.100.2</remoteaddress>
        <type>TCP</type>
      </commpath>
                :
                   < Partially omitted >
                :
      <instance order="3" tag="/DATA2">
        <ID>/DATA2</ID>
        <app>gen</app>
        <info>
          <altblock>0</altblock>
          <perm>rw,barrier=0</perm>
          <type>ext4</type>
        </info>
        <init>SEC_ISP</init>
        <state>OSU</state>
        <switchback>INTELLIGENT</switchback>
        <typ>filesys</typ>
      </instance>
    </node>
  </lifekeeper>
```

The file created here is copied to the Chef workstation (The Linux node where the attribute conversion script “exp2chef.pl” was copied in section 3-1 ‘Environment Preparation’ step 3 above).

3-3 Convert the Existing Cluster Information into a Chef File

Log in to the Linux node where the exp2chef.pl was copied in section 3-1 'Environment Preparation t' step 3 above.

Specify the cluster configuration XML file as an argument, and run the exp2chef.pl script.

~/Chef/exp2chef.pl <Cluster information XML file >

Example: The Cluster information XML file is ~/Chef/resource.xml

$ ~/Chef/exp2chef.pl ~/Chef/resource.xml

When this script is run, two attribute files are generated in the same directory as the input XML file. One for the LifeKeeper communication paths and one for the LifeKeeper resource hierarchies.

The output file names generated by exp2chef.pl use the input xml file name minus the xml extension (resource.xml minus the xml extension results in the base name 'resource' for the output scripts). The base name and the attribute file extension 'rb' are combined with the attribute file type, comm for the LifeKeeper communication paths and 'res' for the LifeKeeper resource hierarchies to produce the names of the individual attribute files (e.g. base_name + attribute_file_type + attribute_file_extesion).

Examples:

Cluster information XML file	…	resource.xml
Attribute file for communication paths	…	resource.comm.rb
Attribute file for resource	…	resource.res.rb

hierarchies

Below are examples of the attribute conversion output files.

Attribute file for communication paths

```
    default['LKROOT']="/opt/LifeKeeper"


      default['node1']['commpath']['0'] = {
       "priority" => "1",
       "baudrate" => "0",
       "remoteaddress" => "192.168.100.2",
       "device" =>
  "192.168.100.1/192.168.100.2",
       "remote" => "node2",
       "type" => "TCP",
       "ipaddress" => "192.168.100.1",
      }


      default['node1']['commpath']['1'] = {
       "priority" => "2",
       "baudrate" => "0",
       "remoteaddress" => "192.168.0.2",
       "device" => "192.168.0.1/192.168.0.2",
       "remote" => "node2",
       "type" => "TCP",
       "ipaddress" => "192.168.0.1",
      }


      default['node2']['commpath']['0'] = {


      ……<Partially omitted>……


      default['node2']['commpath']['1'] = {
       "priority" => "2",
       "baudrate" => "0",
       "remoteaddress" => "192.168.0.1",
       "device" => "192.168.0.2/192.168.0.1",
       "remote" => "node1",
       "type" => "TCP",
```

Attribute file for resource hierarchies

```
     default['LKROOT']="/opt/LifeKeeper"


     default['node1']['dependency']['0'] = {
      "parent" => "/DATA1",
      "child" => "datarep-DATA1",
     }


     default['node1']['dependency']['1'] = {
      "parent" => "/DATA2",
      "child" => "datarep-DATA2",
     }


     default['node1']['equivalency']['datarep-
DATA1'] = {
      "priority" => "1",
      "rtag" => "datarep-DATA1",
      "tag" => "datarep-DATA1",
      "type" => "SHARED",
      "remote" => "node2",
      "rpriority" => "10",
     }


     default['node1']['equivalency']['/DATA1']
= {
      "priority" => "1",
     ……<Partially omitted>……
      "perm" => "rw,barrier=0",
      "app" => "gen",
      "init" => "SEC_ISP",
      "state" => "OSU",
      "order" => "3",
      "tag" => "/DATA2",
      "typ" => "filesys",
```

3-4 Preparing for the New Cluster Generation

Prepare the LifeKeeper Chef cookbooks listed below. This is done on the Chef Workstation.

- LifeKeeper installation -Cookbook name: lkinstall
- LifeKeeper communication path creation -Cookbook name:commpath
- LifeKeeper resource hierarchy creation -Cookbook name: resources)

Create the LifeKeeper cookbooks lkinstall, commpath, and resources using the Chef knife command.
See the Chef documentation for more information on using the knife command.

Copy the LifeKeeper packages, License key files, and Chef recipe/attribute files to the LifeKeeper cookbooks created using the knife command.
The required files are listed below.
Copy all files to the specified directories.

■ **LifeKeeper License Key and rpm files for installation of LifeKeeper**

Copy <Source File> to <cookbook path>/lkinstall/files/default

Note: These files are included in the CD image for LifeKeeper (sps.img). The paths listed assume the installation image is mounted on /mnt.

Source file	Note
Common	
/mnt/common/steeleye-perl-*.rpm	
/mnt/common/steeleye-openssl-*.rpm	
/mnt/common/steeleye-openssl-perl-*.rpm	
/mnt/common/steeleye-libgpg-error-*.rpm	
/mnt/common/steeleye-libgcrypt-*.rpm	
/mnt/common/steeleye-libcurl-*.rpm	
/mnt/common/steeleye-curl-*.rpm	
/mnt/common/steeleye-readline-*.rpm	
/mnt/common/steeleye-gnutls-*.rpm	
/mnt/common/steeleye-gnutls-utils-*.rpm	
/mnt/common/steeleye-libxml2-*.rpm	
/mnt/common/steeleye-libxml2-static-*.rpm	
/mnt/common/steeleye-pcre-*.rpm	
/mnt/common/steeleye-perl-addons-*.rpm	
/mnt/common/steeleye-lighttpd-*.rpm	
/mnt/common/steeleye-lighttpd-fastcgi-*.rpm	
/mnt/common/steeleye-lkapi-*.rpm	
/mnt/common/steeleye-lkapi-client-*.rpm	
/mnt/common/steeleye-pdksh-*.rpm	
/mnt/common/steeleye-runit-*.rpm	
/mnt/core/steeleye-lk*.rpm	

LifeKeeper License Key File	Not included in this product. Need to acquire separately.

Source file	Note
Each OS specific	
RedHat Enterprise Linux	
/mnt/RHAS/HADR-RHAS-2.6.32-all.x86_64*.rpm	6.x only
/mnt/RHAS/HADR-RHAS-3.10.0-all.x86_64*.rpm	7.x only
/mnt/RHAS/steeleye-lkRHAS-*.rpm	
CentOS	
/mnt/CentOS/HADR-CentOS-2.6.32-all.x86_64*.rpm	6.x only
/mnt/CentOS/HADR-CentOS-3.10.0-all.x86_64*.rpm	7.x only
/mnt/CentOS/steeleye-lkCentOS-*.rpm	
Oracle Linux	
/mnt/OEL/HADR-OEL-2.6.32-all.x86_64*.rpm	6.x only (Except UEK)
/mnt/OEL/HADR-OEL-3.10.0-all.x86_64*.rpm	7.x only (Except UEK)
/mnt/OEL/steeleye-lkOEL-*.rpm	

●Recovery Kits

Copy only the Recovery Kits that will be installed. These rpm are also found on the LifeKeeper installation media. The source path assume the media has been mounted on /mnt.

Source file	Note
/mnt/kits/steeleye-lkAPA-*.noarch.rpm	Apache ARK
/mnt/kits/steeleye-lkDR-*.noarch.rpm	DataKeeper
/mnt/HADR-generic-*.rpm	
/mnt/kits/steeleye-lkPGSQL-*.noarch.rpm	PostgreSQL ARK
/mnt/kits/steeleye-lkSQL-*.noarch.rpm	MySQL ARK

■Chef Support file

The source and destination locations for each file are different. Use the table below to obtain the source and destination location.

Source file	Copy destination
/mnt/Chef/recipe/lkinstall.rb	<cookbook path>/lkinstall/recipes/default.rb
/mnt/Chef/recipe/commpath.rb	<cookbook path>/commpath/recipes/default.rb
/mnt/Chef/recipe/resources.rb	<cookbook path>/resource/recipes/default.rb
/mnt/Chef/attribute/lkinstall.rb	<cookbook path>/lkinstall/attributes/default.rb

●Chef attribute files generated by running exp2chef.pl

Generated file	Copy to
comm path attribute	<cookbook path>/commpath/attributes/default.rb
resource attribute	<cookbook path>/resources/attributes/default.rb

3-5 Editing Attribute Files

When generated the attribute files contain some information that is unique to the system on which it was generated. Because of this some of the parameter values must be modified before it can be used to generate nodes in a new LifeKeeper cluster. The tables below list the parameters, a short description of the parameter and a notation on if a change to the generated parameter value is required.

■Parameters in the comm path attribute file

Parameter	Description	Change Required
node name	Text string containing the name of new cluster node. The string starts with a null value [''].	Required
priority	Communication path priority	
baudrate	Baud rate for the TTY communication path connection (not used for tcp communication paths).	
remoteaddress	IP address of the remote node. TCP communication paths have 2 end points, the local node and the remote node. This parameter contains the IP address of the remote node. This parameter along with the Ipaddress parameter define the communication path end points.	Required
Device	The device used for tty communication paths.	Required
Remote	Name of the remote node in the LifeKeeper cluster.	Required

Type	Type of communication path connection TTY/TCP	
Ipaddress	IP address of the local node. TCP communication paths have 2 end points, the local node and the remote node. This parameter contains the IP address of the local node. This parameter along with the remoteaddress parameter define the communication path end points.	Required

■**Parameters in the resource attribute file: Dependency section**

Parameter	Description	Change Required
Parent	Resource tag of the Parent resource	Required*
Child	Resource tag of the Child resource	Required

*This parameter is not required if the tag names will not be changed.

■**Parameters in the resource attribute file: Equivalency section**

Parameter	Description	Change Required
priority	Priority	
rtag	Tag name on remote node	Required*
tag	Tag name on local node	Required*
type		
remote	Node name of remote node	Required
rpriority	Priority of remote node	

*This parameter is not required if the tag names are not changed.

■**Parameters in the resource attribute file: Instance section**

The main parameter list for resource instances for a LifeKeeper hierarchy. The parameters are broken down into those that are common to all LifeKeeper resource types and those specific to a LifeKeeper resource instance type.

Parameters common to each instance

Parameter	Description	Change Required
ID	Resource ID	
typ	Resource type	
tag	Resource tag	
switchback	Switchback type	
state	Resource status	

Parameter typ=ip

Parameter	Description	Change Required

primach	Node name	Required
priif	NIC	
mask	net mask	
ipaddr	Virtual IP address of IP resource	Required

Parameter typ=netraid

Parameter	Description	Change Required
ID	Device ID（/dev/sdb, etc）	
num	md number	
async	Sync mode	
mountpoint	Mount point	
bitmap	bitmap file location	
ipaddr	The IP addresses for the replication path end points (local and remote node)	Required*

* The defined endpoints much match define communication path endpoints.

Parameter typ=apache

Parameter	Description	Change Required
root	Location of the httpd.conf file	
path	Location of the httpd daemon executable	

Parameter typ=pgsql

Parameter	Description	Change Required
osexe	Path of the Postgres execution file	
datadir	The protected Postgres data directory	
port	Port for client communication	
socket	Specify the full path to Socket used for Client communication	
clientexe	The pg_ctl path for the Postgres execution file	

dbuser	User name of Postgres data administrator	
exepath	The psql path for the Postgres executables	
logfile	Path for the Postgres log file	
osuser	os user id	

Parameter typ=mysql

Parameter	Description	Change Required
insno	Protection Instance Number	
bindir	Location of MySQL executables	
confdir	The full path name (Except the file name) where MySQL configuration file (my.cnf) is located	
datadir	Data directory for the protected database instance	

■Parameters in the lkinstall attribute file

Edit the attribute file under the LifeKeeper installation cookbook that was created in the preparation step to define the license information.

Example:

default[node][‘license’] = [‘example1.lic’,’example2.lic’]

- node: Specify the node name on which the lifeKeeper license will be installed.
- license: Specify license file(s).

- example1.lic, example2… A comma separated list of license files.

This completes the description of the all the parameters in the attribute files.

3-6 Generating a New Cluster

Below are the general procedures for generating a new cluster. For more information on using the Chef commands such as knife or Chef-client, see the Chef documentation.

1. Upload a cookbook to a server.

   Upload a cookbook from a Chef Workstation to a Chef Server by using the knife

   cookbook upload.

2. Register run list

   Register a recipe to run list by using the knife node run_list add.

3. Install chef-client

   Install the Chef client on the nodes in the new HA cluster using Chef.

   Perform the “knife bootstrap node name” command on the Chef Workstation to install.

4. Execute the recipe

   Perform the chef-client command using the recipe registered at step 2 above via the run list command.

Execute the recipe on the node which will serve as the primary LifeKeeper node

first. When that completes execute the recipe on the LifeKeeper standby node.

Note

- **DataKeeper resource configurations**

In some DataKeeper environments the device used must be added to the

/opt/LifeKeeper/subsys/scsi/resources/DEVNAME/device_pattern file before the DataKeeper resource can be created.
For more information on this, see **Technical Documentation > LifeKeeper > Troubleshooting > Known Issues and Restrictions,** or **DataKeeper > Troubleshooting.**

- **IP resource configurations**

By default the IP resource health monitoring process attempts to determine if the virtual IP address can still send and receive packets outside the local node. This is done through broadcast pings to the subnet or via a ping of a specific IP address. If the device used by the template node is not reachable via the new cluster node, then this information must be updated. For more information, see **IP Recovery Kit Administration Guide >Viewing and Editing IP Configuration Properties.**

5. Start the LifeKeeper GUI and confirm if each resource was generated as expected. At this point, all resource hierarchies are in the OSU state except DataKeeper.

6. Perform an in Service on the source node to start all protected resources.