



SIOS Protection Suite for Linux

v9.0

テクニカルドキュメンテーション

2015年9月

本書およびその内容は SIOS Technology Corp. (旧称 SteelEye® Technology, Inc.) の所有物であり、許可なき使用および複製は禁止されています。SIOS Technology Corp. は本書の内容に関していかなる保証も行いません。また、事前の通知なく本書を改訂し、本書に記載された製品に変更を加える権利を保有しています。SIOS Technology Corp. は、新しい技術、コンポーネント、およびソフトウェアが利用可能になるのに合わせて製品を改善することを方針としています。そのため、SIOS Technology Corp. は事前の通知なく仕様を変更する権利を保留します。

LifeKeeper、SteelEye、および SteelEye DataKeeper は SIOS Technology Corp. の登録商標です。

本書で使用されるその他のブランド名および製品名は、識別のみを目的として使用されており、各社の商標が含まれています。

出版物の品質を維持するために、弊社は本書の正確性、明瞭性、構成、および価値に関するお客様のご意見を歓迎いたします。

以下の宛先に電子メールを送信してください。

ip@us.sios.com

Copyright © 2015

By SIOS Technology Corp.

San Mateo, CA U.S.A.

All rights reserved

目次

Chapter 1: はじめに	1
SIOS Protection Suite for Linux について	1
SPS for Linux の統合コンポーネント	1
ドキュメンテーションとトレーニング	1
ドキュメンテーション	1
トレーニング	2
テクニカルサポート	2
Chapter 2: SIOS LifeKeeper for Linux	3
はじめに	3
保護対象のリソース	3
LifeKeeper Core	4
LifeKeeper Core ソフトウェア	4
File System、Generic Application、IP、および RAW I/O の Recovery Kit ソフトウェア	5
LifeKeeper GUI ソフトウェア	6
LifeKeeper のマニュアルページ	6
設定の概念	6
共通のハードウェアコンポーネント	6
すべての LifeKeeper 設定に共通するコンポーネント	7
システムのグループ化の配置	7
アクティブ - アクティブのグループ化	8
アクティブ - スタンバイのグループ化	9
インテリジェントスイッチバックと自動スイッチバックの違い	10
syslog によるログの記録	11
リソース階層	11
リソースタイプ	11

リソースの状態	12
階層の関係	13
イクイバレンシ情報	13
リソース階層の情報	14
リソース階層の例	15
ステータスの詳細表示	15
リソース階層の情報	17
通信ステータスの情報	18
LifeKeeper のフラグ	19
シャットダウンストラテジー	20
ステータスの簡略表示	20
リソース階層の情報	20
通信ステータスの情報	21
障害検出とリカバリのシナリオ	21
IP ローカルリカバリ	21
ローカルリカバリのシナリオ	21
コマンドラインの操作	22
リソースのエラーリカバリのシナリオ	23
サーバの障害リカバリのシナリオ	25
インストールと設定	27
SPS for Linux のインストール	27
SPS for Linux の設定	27
SPS の設定手順	27
TTY 接続のセットアップ	28
SNMP による LifeKeeper イベント転送	29
SNMP による LifeKeeper イベント転送の概要	29
LifeKeeper イベントテーブル	29
LifeKeeper イベント転送の設定	31
前提条件	31
設定作業	31

設定の確認	32
SNMP イベント転送の無効化	32
SNMP のトラブルシューティング	32
LifeKeeper イベントメール通知	33
LifeKeeper イベントメール通知の概要	33
メールが生成される LifeKeeper のイベント	33
LifeKeeper イベントメール通知の設定	35
前提条件	35
設定作業	35
設定の確認	35
イベントメール通知の無効化	35
メール通知のトラブルシューティング	36
任意の設定作業	36
[Confirm Failover] と [Block Resource Failover] の設定	36
[Set Confirm Failover On]	37
[Confirm Failover]設定を選択するタイミング	39
[Block Resource Failover On]	40
設定の利用条件 / 考慮事項	42
設定例	42
全ての自動フェイルオーバーをすべてブロックする	42
一方方向のフェイルオーバーをブロックする	44
サーバのシャットダウンストラテジーの設定	45
LifeKeeper ハートビートの調整	46
ハートビート設定項目の概要	46
例	47
ハートビートの設定	47
設定上の考慮事項	47
SPS API でカスタム証明書を使用する	48
証明書の使用方法	48

独自の証明書の使用	48
Linux の設定	48
データレプリケーションの設定	51
ネットワーク設定	51
アプリケーションの設定	52
ストレージとアダプタの設定	53
HP のマルチパス I/O 設定	65
日立 HDLM のマルチパス I/O 設定	66
Device Mapper のマルチパス I/O 設定	108
LifeKeeper I-O フェンシングの概要	111
SCSI リザベーション	112
SCSI リザベーションを利用したストレージフェンシング	112
I/O フェンシングのための代替方式	113
リザベーションの無効化	113
非共有ストレージ	114
リザベーションを使用しない I/O フェンシングの設定	114
I/O フェンシング表	114
Quorum/Witness	116
Quorum/Witness Server Support Package for LifeKeeper	116
機能の概要	116
パッケージの要件	116
パッケージのインストールと設定	117
設定可能なコンポーネント	117
使用可能な quorum モード	118
使用可能な witness モード	119
Quorum を喪失したときに利用可能なアクション	119
共有 witness トポロジーのための追加設定	120
2 ノードクラスタに witness ノードを追加する	121
期待される動作 (デフォルトモードを仮定)	122
シナリオ 1	122

シナリオ 2	122
シナリオ 3	122
シナリオ 4	123
STONITH	123
STONITH で IPMI を使用する	124
パッケージの要件	124
VMware vSphere 環境での STONITH	124
パッケージの要件	124
STONITH サーバ	124
監視対象仮想マシン	124
インストールと設定	124
<vm_id>	126
期待される動作	126
Watchdog	126
コンポーネント	126
設定	127
アンインストール	128
リソースポリシー管理	129
概要	129
SIOS Protection Suite	129
ポリシーによるカスタム動作 およびメンテナンスモード動作	129
標準ポリシー	130
メタポリシー	130
リソースレベルのポリシーに関する重要な考慮事項	131
lkpolicy ツール	131
lkpolicy の使用方法の例	131
ローカルおよびリモートサーバとの認証	131
ポリシーのリスト表示	132
現在のポリシーの表示	132
ポリシーの設定	132

ポリシーの削除	133
認証情報の設定	133
認証情報の追加または変更	133
ストア内の認証情報のリスト表示	133
サーバの認証情報の削除	134
追加情報	134
LifeKeeper API	134
ネットワーク設定	134
認証	134
Chapter 2: LifeKeeper 管理	135
概要	135
エラーの検出および通知	135
N-Way リカバリ	135
管理作業	136
サーバプロパティの編集	136
コミュニケーションパスの作成	136
コミュニケーションパスの削除	138
サーバのプロパティ - フェイルオーバー	138
リソース階層の作成	140
LifeKeeper アプリケーションリソース階層	140
Recovery Kit のオプション	141
ファイルシステムリソース階層の作成	141
Generic Application リソース階層の作成	142
Raw デバイスリソース階層の作成	143
リソースのプロパティの編集	144
リソースの優先順位の編集	144
[Up] および [Down] ボタンの使用	145
優先順位の値の編集	146
変更の適用	146
リソース階層の拡張	146

ファイルシステムリソース階層の拡張	147
Generic Application リソース階層の拡張	147
Raw デバイスリソース階層の拡張	148
階層の拡張解除	148
リソース依存関係の作成	149
リソース依存関係の削除	150
すべてのサーバからの階層の削除	151
LifeKeeper User Guide	152
LifeKeeper for Linux の使用	152
GUI	153
GUI の概要 - 全般	153
GUI サーバ	153
GUI クライアント	153
GUI クライアントの終了	153
LifeKeeper GUI ソフトウェアパッケージ	153
メニュー	154
SIOS LifeKeeper for Linux のメニュー	154
リソースのコンテキストメニュー	154
サーバのコンテキストメニュー	155
[File] メニュー	156
[Edit] メニュー - [Resource]	157
[Edit] メニュー - [Server]	157
[View] メニュー	158
[Help] メニュー	159
ツールバー	159
SIOS LifeKeeper for Linux のツールバー	159
GUI のツールバー	159
リソースのコンテキストツールバー	161
サーバのコンテキストツールバー	162
GUI の実行の準備	163

LifeKeeper の GUI - 概要	163
GUI サーバ	163
GUI クライアント	163
GUI クライアントの開始	164
LifeKeeper GUI アプレットの開始	164
アプリケーションクライアントの開始	164
GUI クライアントの終了	164
LifeKeeper の GUI の設定	164
GUI 管理用の LifeKeeper サーバの設定	164
GUI の実行	165
GUI の設定	165
GUI の制限	166
GUI サーバの開始 / 停止	166
LifeKeeper GUI サーバを開始するには	166
トラブルシューティング	167
LifeKeeper GUI サーバを停止するには	167
LifeKeeper GUI サーバのプロセス	167
Java のセキュリティポリシー	168
ポリシーファイルの場所	168
ポリシーファイルの作成と管理	168
ポリシーファイルでの権限の付与	169
ポリシーファイルの例	169
Java プラグイン	170
Java プラグインのダウンロード	170
リモートシステムでの GUI の実行	170
リモートシステムでの GUI の設定	171
リモートシステムでの GUI の実行	171
アプレットのトラブルシューティング	172
LifeKeeper サーバでの GUI の実行	173
GUI アプレットを使用するためのブラウザのセキュリティパラメータ	173

Firefox	173
Internet Explorer	173
ステータスの表	174
プロパティパネル	174
出力パネル	175
メッセージバー	175
GUI の終了	175
共通の作業	175
LifeKeeper の起動	175
LifeKeeper サーバプロセスの起動	176
LifeKeeper の自動再起動の有効化	176
LifeKeeper の停止	176
LifeKeeper の自動再起動の無効化	177
LifeKeeper プロセスの表示	177
LifeKeeper GUI サーバプロセスの表示	178
LifeKeeper の制御プロセスの表示	179
サーバのクラスタへの接続	180
クラスタからの切断	181
接続サーバの表示	181
サーバのステータスの表示	181
サーバのプロパティの表示	182
サーバのログファイルの表示	182
リソースのタグと ID の表示	183
リソースのステータスの表示	183
サーバリソースのステータス	183
グローバルリソースのステータス	184
リソースのプロパティの表示	185
Resource Labels	186
Resource Labels	186
メッセージ履歴の表示	186

メッセージ履歴の解釈	187
リソース階層ツリーの展開と折り畳み	187
[Cluster Connect] ダイアログ	188
[Cluster Disconnect] ダイアログ	189
[Resource Properties] ダイアログ	189
[General] タブ	189
[Relations] タブ	190
[Equivalencies] タブ	190
[Server Properties] ダイアログ	191
[General] タブ	191
[CommPaths] タブ	193
[Resources] タブ	194
オペレータの作業	195
リソースを In Service にする	195
リソースを Out of Service にする	196
高度な作業	196
LCD	196
LifeKeeper 設定 データベース	196
関連トピック	197
LCDI のコマンド	197
シナリオの状況	197
階層の定義	198
LCD の設定データ	200
依存関係の情報	200
リソースのステータス情報	200
サーバ間のイクイバレンシ情報	200
LCD のディレクトリ構造	201
LCD のリソースタイプ	201
LifeKeeper のフラグ	201
リソースのサブディレクトリ	202

リソースの動作	203
/opt/LifeKeeper の LCD のディレクトリ構造	203
LCM	204
通信ステータスの情報	205
LifeKeeper の警報とリカバリ	205
警報 クラス	205
警報 の処理	206
警報 ディレクトリのレイアウト	206
メンテナンス作業	206
LifeKeeper の設定値の変更	206
ファイルシステムの健全性の監視	208
条件の定義	209
フル(またはほぼフル)のファイルシステム	209
アンマウントされた、または不適切にマウントされたファイルシステム	209
LifeKeeper が保護するシステムのメンテナンス	210
リソース階層のメンテナンス	210
フェイルオーバー後の復旧	210
LifeKeeper の削除	211
GnoRPM からの削除	212
コマンドラインからの削除	212
ディストリビューションの有効化パッケージの削除	212
ファイアウォールを使用した状態での LifeKeeper の実行	212
LifeKeeper のコミュニケーションパス	213
LifeKeeper GUI の接続	213
LifeKeeper の IP アドレスリソース	213
LifeKeeper Data Replication	213
ファイアウォールの無効化	214
ファイアウォール経由での LifeKeeper GUI の実行	214
リソース階層の転送	215
テクニカルノート	216

LifeKeeper の機能	216
チューニング	217
LifeKeeper の動作	218
サーバの設定	219
LifeKeeper 8.2.0 以降の GUI 要件	219
[Confirm Failover] と [Block Resource Failover] の設定	219
Confirm Failover On:	220
Block Resource Failover On:	220
条件 / 考慮事項:	220
NFS クライアントのオプション	221
NFS クライアントをマウントするときの考慮事項	221
UDP または TCP の選択	221
/etc/exports の Sync オプション	221
Red Hat EL6 (および Fedora 14) クライアントと Red Hat EL6 NFS サーバの使用	221
Red Hat EL5 NFS クライアントと Red Hat EL6 NFS サーバの使用	222
クラスタの例	222
拡張したマルチクラスタの例	222
トラブルシューティング	223
SPS が開始するフェイルオーバーの一般的な原因	223
サーバレベルでの原因	224
サーバの障害	224
通信障害 / ネットワーク障害	224
スプリットブレイン	225
リソースレベルでの原因	225
アプリケーションの障害	226
ファイルシステム	226
IP アドレスの障害	226
リザベーションコンフリクト	227
SCSI デバイス	227
既知の問題と制限	227

インストール	227
LifeKeeper Core	231
インターネット /IP ライセンス	237
GUI	237
データレプリケーション	240
IPv6	242
Apache	245
Oracle Recovery Kit	245
NFS Server Recovery Kit	246
SAP Recovery Kit	248
LVM Recovery Kit	248
DMMP Recovery Kit	249
DB2 Recovery Kit	250
	250
MD Recovery Kit	251
SAP DB/MaxDB Recovery Kit	252
Sybase ASE Recovery Kit	253
GUI トラブルシューティング	256
ネットワーク関連トラブルシューティング (GUI)	256
Windows プラットフォームでの論理接続の遅延	256
Sun FAQ から:	256
モデムからの実行:	256
プライマリネットワークインターフェースのダウン:	257
ホストへのルートが存在しない例外:	257
不明なホストの例外:	257
Windows から:	258
Linux から:	259
X Window Server に接続できない:	260
コミュニケーションパスの稼働と停止	260
推奨される対策	260

不完全なリソースの作成	261
不完全なリソースの優先順位の変更	261
一貫した状態への階層のリストア	261
階層の設定中に共有ストレージが見つからない	262
LifeKeeper サーバ障害からの復旧	263
推奨される対策:	263
停止できないプロセスからの復旧	264
手動リカバリ時のパニックからの復旧	264
Out-of-Service 階層の復旧	264
リソースタグ名の制限	264
タグ名の長さ	264
有効な特殊文字	264
無効な文字	264
シリアル (TTY) コンソールの警告	265
システムが init 状態 S に遷移しているという警告	265
共有ストレージでスレッドがハングしているというメッセージ	265
説明	265
推奨される対策:	266
Chapter 3: SIOS DataKeeper for Linux	267
はじめに	267
SIOS DataKeeper for Linux によるミラーリング	267
DataKeeper の特長	267
同期ミラーリングと非同期ミラーリングの違い	268
同期ミラーリング	268
非同期ミラーリング	268
SIOS DataKeeper の仕組み	269
同期 (および再同期)	269
標準ミラーの構成	270
N+1 Configuration	270
複数ターゲットの構成	271

SIOS DataKeeper リソース階層	272
ファイルオーバのシナリオ	273
シナリオ 1	273
シナリオ 2	273
シナリオ 3	274
シナリオ 4	274
Chapter A: インストールと設定	277
DataKeeper リソースを設定する前に	277
ハードウェアとソフトウェアの要件	277
ハードウェアの要件	277
ソフトウェアの要件	277
全般的な設定	278
ネットワーク設定	278
データレプリケーションパスの変更	278
ネットワーク帯域幅の要件の特定	279
Linux システム(物理または仮想)での変化率の測定	279
基本変化率の測定	280
詳細変化率の測定	280
収集した詳細変化率データの解析	280
詳細変化率データのグラフ作成	286
SIOS DataKeeper for Linux のリソースタイプ	289
Replicate New File System	290
Replicate Existing File System	290
DataKeeper Resource	290
リソースの設定作業	290
概要	291
DataKeeper リソース階層の作成	291
リソース階層の拡張	293
DataKeeper リソース階層の拡張	294
リソース階層の拡張解除	296

リソース階層の削除	296
DataKeeper リソースを Out of Service にする	297
DataKeeper リソースを In Service にする	297
リソース階層のテスト	297
LifeKeeper の GUI からの手動スイッチオーバーの実行	297
管理	299
SIOS DataKeeper for Linux の管理	299
ミラーのステータスの表示	299
GUI からのミラーの管理	300
ミラーを強制的にオンラインにする	301
一時停止と再開	302
ミラーの一時停止	302
ミラーの再開	302
圧縮レベルの設定	302
コマンドラインからのミラー管理	302
ミラーの操作	302
例:	303
ミラーの設定	303
例:	303
ミラーのサイズ変更	304
ミラーのサイズ変更の推奨手順:	304
ビットマップの管理	304
コマンドラインからのミラーステータスの監視	305
例:	305
サーバの障害	306
再同期	306
全同期の回避	307
方法 1	307
手順	307
方法 2	308

手順	309
DataKeeper で LVM を使用する	309
Fusion-io を使用するクラスタ化	310
DataKeeper のパフォーマンスを最大に発揮させるための Fusion-io のベストプラクティス	310
ネットワーク	311
TCP/IP の調整	311
設定上の推奨項目	312
Multi-Site Cluster	313
SIOS Protection Suite for Linux Multi-Site Cluster	313
SIOS Protection Suite for Linux Multi-Site Cluster	313
Multi-Site Cluster を設定する際の考慮事項	314
ローカルサイトのビットマップの保存先	314
マルチサイトクラスタ設定で避けるべきリソース構成	315
Multi-Site Cluster の設定上の注意点	315
SIOS Protection Suite for Linux Multi-Site Cluster リソース階層の作成	316
Replicate New File System	317
Replicate Existing File System	319
DataKeeper Resource	321
リソース階層の拡張	323
DataKeeper リソース階層の拡張	325
ディザスタリカバリシステムへの階層の拡張	325
リストアおよびリカバリの設定	329
Multi-Site Cluster 環境へのマイグレーション	329
要件	329
始める前に	330
マイグレーションの実行	330
マイグレーションの正常な完了	339
トラブルシューティング	341
Recovery Kit	344
Index	345

Chapter 1: はじめに

SIOS Protection Suite for Linux について

SIOS Protection Suite (SPS) for Linux は、高可用性のクラスタリングと革新的なデータ複製機能をエンタープライズクラスのソリューションに統合したものです。

SPS for Linux の統合コンポーネント

SIOS LifeKeeper は、障害回復性の高いソフトウェアソリューションであり、お使いのサーバのファイルシステム、アプリケーション、およびプロセスの高い可用性を維持します。LifeKeeper には、カスタマイズした耐障害性のハードウェアは不要です。LifeKeeper を使用するには、ネットワーク内にある 2 台以上のシステムをグループ化するだけです。サイト固有の構成データが作成され、自動の障害検出とリカバリが実行されます。

障害が発生した場合、障害が発生したサーバから LifeKeeper が保護しているリソースを指定のバックアップサーバに移行します。実際のスイッチオーバー時に短時間の中断が発生します。ただし、オペレータの介入なしに LifeKeeper がバックアップサーバに動作をリストアします。

SIOS DataKeeper は、LifeKeeper 環境において統合データミラーリング機能を提供します。この機能により、LifeKeeper リソースが共有/非共有ストレージ環境で動作可能になります。

ドキュメンテーションとトレーニング

ドキュメンテーション

SIOS Protection Suite for Linux をインストール、設定、管理、およびトラブルシューティングするための方法を説明する完全なリファレンスです。次のセクションは、SPS for Linux のあらゆる側面を網羅します。

セクション	説明
はじめに	SIOS Protection Suite for Linux 製品の概要を説明します。ソフトウェアパッケージと設定コンセプトが含まれます。
SPS for Linux インストールガイド	SPS 環境のプランニングと設定、SPS のインストールとライセンス、LifeKeeper のグラフィカルユーザインターフェース (GUI) の設定に役立つ情報を提供します。
設定	クラスタ内の各サーバで LifeKeeper ソフトウェアを設定するための詳細情報と手順があります。
管理	サーバのプロパティの編集やリソースの作成などのサーバレベルの作業、およびリソースの編集、拡張、削除などのリソースレベルの作業について説明します。
User's Guide	LifeKeeper GUI で実行できる多数の作業を含めて、 LifeKeeper の GUI に関する詳細情報が 있습니다。 テクニカルノート セクション、および多数の 高度なトピック もあります。

セクション	説明
DataKeeper	SIOS DataKeeper for Linux の計画とインストールの手順、および管理、設定、およびユーザの情報が含まれます。
トラブルシューティング	既知の問題と制限について説明し、SIOS LifeKeeper for Linux のインストール、設定、および使用を行うときに発生する可能性がある問題に対する解決策を説明します。
Recovery Kit	LifeKeeper で特定のアプリケーションを管理および制御するために必要なオプションの Recovery Kit のプランニングおよびインストール手順、管理、設定、およびユーザ情報が含まれます。
エラーコードの検索	SIOS Protection Suite for Linux の使用中に表示される可能性のあるすべてのメッセージの一覧が含まれます。必要に応じて、エラーの原因およびエラー状態を解消するために必要な処置についても説明しています。この全一覧から、受信したエラーコードを検索できます。

トレーニング

SPS トレーニングは、SIOS Technology Corp. または代理店から受講可能です。詳細については、営業担当者にお問い合わせください。

テクニカルサポート

SIOS Technology Corp. と有効なサポート契約を結んだお客様は、[SIOS Technology Corp. のセルフサービスサポートポータル](#)にアクセスできます。

[SIOS Technology Corp. のセルフサービスサポートポータル](#)では、以下のことができます。

- 弊社のソリューションナレッジベースから、問題の解決策と質問に対する回答を検索する。
- 次のメニューを選択して、年中無休の SIOS Technology Corp. のサポートチームにアクセスする。
 - **Log a Case** - 新しいインシデントを報告する。
 - **View Cases** - お客様の未解決と解決済みのインシデントをすべて表示する。
 - **Review Top Solutions** - 弊社のお客様が表示した、最も一般的な問題の解決策の情報を表示する。

セルフサービスポータルを設定してアカウントを有効にする方法については、SIOS Technology Corp. のサポート (support@us.sios.com) にお問い合わせください。

また、SIOS Technology Corp. のサポートには、以下の方法でも連絡できます。

1-877-457-5113 (通話料無料)

1-803-808-4270 (米国以外のお客様)

電子メール: support@us.sios.com

Chapter 2: SIOS LifeKeeper for Linux

はじめに

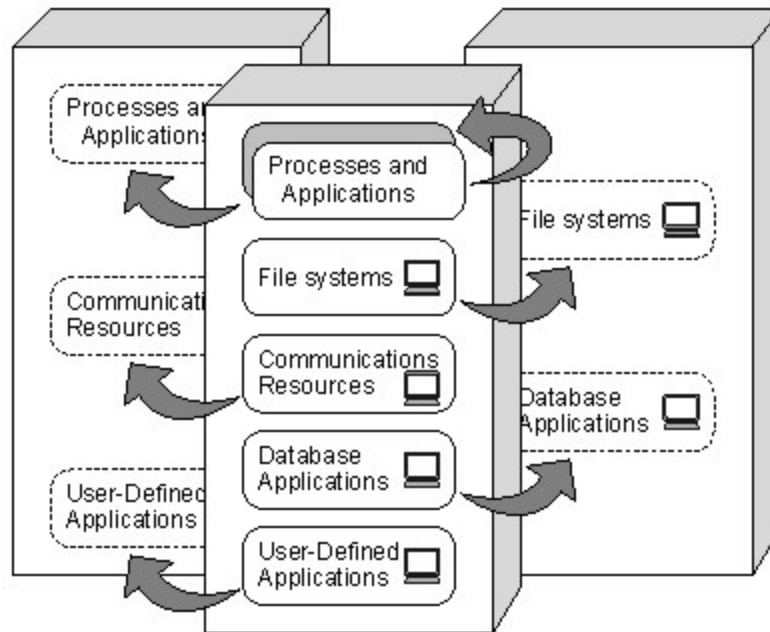
SIOS LifeKeeper for Linux は、さまざまなストレージ構成をサポートし、最大 32 ノードの高可用性クラスタリングを提供します。共有ストレージ(ファイバチャネル SAN、iSCSI)、ネットワーク接続ストレージ(NAS)、ホストベースの複製、HP Continuous Access などのアレイベースの SAN 複製との統合などをサポートします。

保護対象のリソース

LifeKeeper ファミリの製品には、多様なシステムリソースにフェイルオーバー保護を提供できるソフトウェアがあります。以下の図に、LifeKeeper の柔軟性、および自動リカバリを指定できるリソースタイプを示します。

- **ファイルシステム**。LifeKeeper では、ext3、ext4、NFS、vxfs、xfs などのファイルシステムの指定とフェイルオーバーができます。
- **通信リソース**。LifeKeeper には、TCP/IP のような通信リソースの通信 Recovery Kit が用意されています。
- **インフラストラクチャリソース**。LifeKeeper には、NFS、Samba、LVM、WebSphere MQ、ソフトウェア RAID (md) など、Linux インフラストラクチャサービス用のオプションの Recovery Kit が用意されています。
- **Web サーバリソース**。LifeKeeper には、Apache Web サーバリソース用のオプションの Recovery Kit が用意されています。
- **データベースとその他のアプリケーション**。LifeKeeper には、Oracle、MySQL、PostgreSQL などの主な RDBMS 製品、および SAP などのエンタープライズアプリケーション用のオプションの Recovery Kit が用意されています。

LifeKeeper は、多様なリソースタイプについて[複数の回復方法](#)をサポートします。



LifeKeeper Core

LifeKeeper Core は、以下の4つの主要コンポーネントで構成されています。

- LifeKeeper Core ソフトウェア
- File System、Generic Application、Raw I/O、および IP の Recovery Kit ソフトウェア
- LifeKeeper GUI ソフトウェア
- LifeKeeper のマニュアルページ

LifeKeeper Core ソフトウェア

LifeKeeper Core ソフトウェアは、以下のコンポーネントで構成されます。

- [LifeKeeper 構成 データベース \(LCD\)](#) - LCD は、LifeKeeper が保護するリソースの情報を保存します。リソースインスタンス、依存関係、イクイバレンス情報、リカバリの方向、LifeKeeper の動作フラグに関する情報が含まれます。システムの起動後にデータが記憶されているように、データは共有メモリにキャッシュされ、ファイルに保存されます。
- [LCD インターフェース \(LCDI\)](#) - LCDI は、LCD に保存されているデータやデータの変更を要求するクエリを設定データベース (LCD) にクエリを送信します。また、リソースの状態や説明の情報を取得するために、Application Recovery Kit が LCDI を使用することもできます。
- [LifeKeeper Communications Manager \(LCM\)](#) - LCM は、クラスタ内にあるサーバのステータスの特定、および LifeKeeper のプロセス間通信 (ローカルとリモート) に使用されます。クラスタ内のあるサーバ上にあるすべてのコミュニケーションパスで LCM 通信がないことは、サーバに障害が発生したことを示します。

- [LifeKeeper アラームインターフェース](#) - LifeKeeper アラームインターフェースは、イベントを起動するためのインフラストラクチャです。LifeKeeper が保護するリソースに障害が検出された場合、アプリケーションデーモンにより sendevent プログラムが呼び出されます。sendevent プログラムが LCD と通信し、リカバリプロセスが使用可能かどうかを判断します。
- LifeKeeper のローカルリカバリ動作と制御のインターフェース (LRACI) - LRACI はリソースに適切なリカバリスクリプトを判断し、リソースに適切な restore / remove スクリプトを呼び出します。

File System、Generic Application、IP、および RAW I/O の Recovery Kit ソフトウェア

LifeKeeper Core は、サーバ上の指定リソースを保護します。リソースを以下に示します。

- File Systems - LifeKeeper では、共有ストレージデバイス上にあるファイルシステムの指定とフェイルオーバーができます。ファイルシステムは、共有 SCSI バス経由で 2 台のサーバからアクセス可能なディスク上に作成できます。LifeKeeper のファイルシステムリソースは、1 台目のサーバに作成されてから、2 台目のサーバに拡張されます。[ファイルシステムの健全性監視](#) がディスクフルと不適切なマウント (またはアンマウント) のファイルシステム条件を検出します。検出した条件に従って、Recovery Kit が警告メッセージのログ記録、ローカルリカバリの試行、またはファイルシステムリソースのバックアップサーバへのフェイルオーバーを実行できます。

File System Recovery Kit に関連するヘルプトピックとして、ファイルシステムのリソース階層の[作成](#)、[拡張](#)、[ファイルシステムの健全性の監視](#)などがあります。

- Generic Applications - Generic Application Recovery Kit は、リソースタイプに対して事前定義リカバリキットが指定されていない汎用アプリケーションやユーザ定義アプリケーションを保護できます。このキットを使用すると、特定アプリケーションについてカスタマイズした監視スクリプトやリカバリスクリプトを指定できます。

Generic Application Recovery Kit に関連するヘルプトピックとして、汎用アプリケーションのリソース階層の[作成](#)、[拡張](#)などがあります。

- IP Addresses - IP Recovery Kit には、LifeKeeper 環境で、障害が発生したプライマリサーバから「切り替え可能な」IP アドレスをバックアップサーバにリカバリするメカニズムがあります。切り替え可能な IP アドレスとは、サーバ間で切り替えることができる仮想 IP アドレスであり、各サーバのネットワークインターフェースカードに関連付けられている IP アドレスとは別のもので、LifeKeeper で保護されているアプリケーションは切り替え可能な IP アドレスに関連付けられているので、プライマリサーバに障害が発生した場合、切り替え可能な IP アドレスはバックアップサーバに関連付けられます。LifeKeeper で保護されているリソースは、切り替え可能な IP アドレスです。

特定の製品、構成、および管理に関する情報については、リカバリキットに含まれる[IP Recovery Kit Technical Documentation](#)を参照してください。

- RAW I/O - RAW I/O Recovery Kit は、カーネルのバッファリングを迂回するアプリケーションのロー I/O デバイスをサポートします。RAW I/O Recovery Kit では、共有ストレージデバイスにボンディングされた RAW デバイスの指定とフェイルオーバーができます。RAW デバイスは、リソースの作成前に、プライマリノードに設定する必要があります。ローリソースを[作成](#)した後、追加サーバに[拡張](#)できます。

LifeKeeper GUI ソフトウェア

LifeKeeper GUI は、Java テクノロジーを使用して開発されたクライアント/サーバアプリケーションであり、LifeKeeper およびその設定データ用のグラフィカルな管理インターフェースです。LifeKeeper GUI クライアントは、[スタンドアロンの Java アプリケーション](#)、および Web ブラウザから呼び出される [Java アプレット](#) の両方として実装されます。

LifeKeeper のマニュアルページ

LifeKeeper 製品用の LifeKeeper Core のリファレンスマニュアルページです。

設定の概念

LifeKeeper は、2 台以上のサーバを持つグループに対してユーザが定義したリソース階層に基づいて機能します。以下のトピックで、LifeKeeper のフェイルオーバー設定の概念を説明しています。

共通のハードウェアコンポーネント

LifeKeeper のすべての設定には、以下の共通コンポーネントが含まれます。

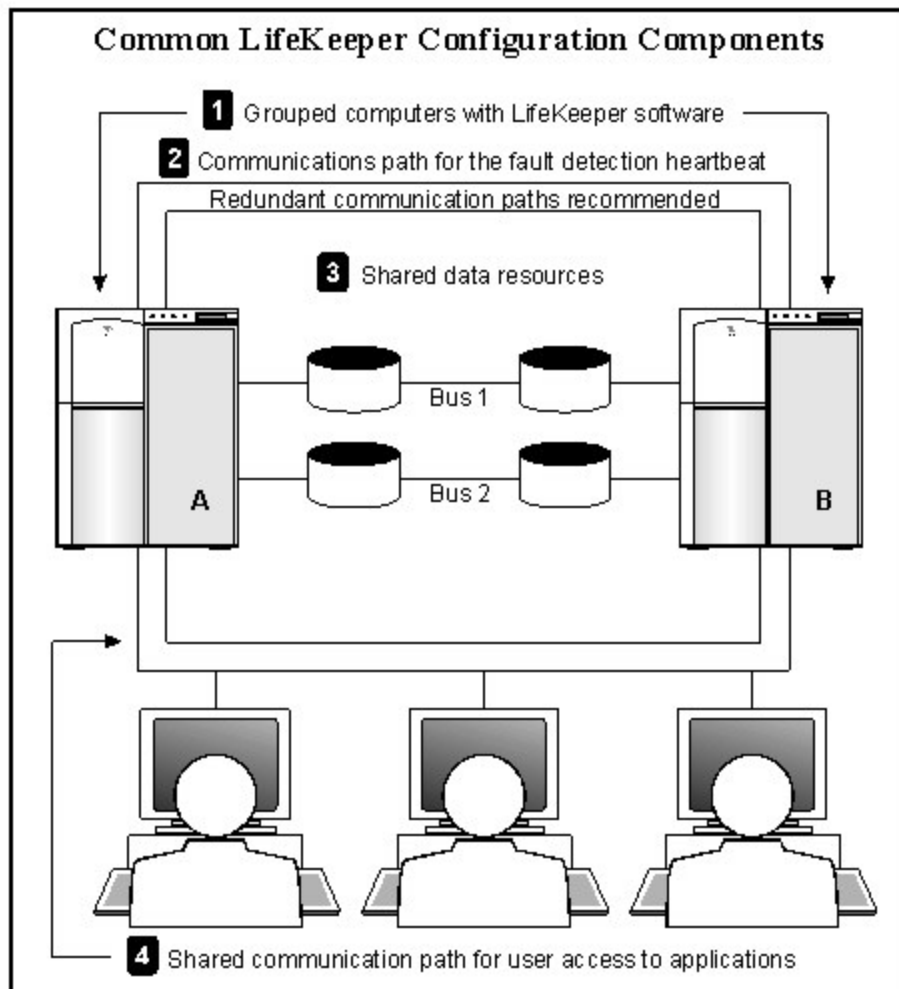
- 1. サーバグループ。** LifeKeeper が提供する障害回復機能は、2 台以上のサーバをクラスタにグループ化することを基礎にしています。サーバは、サポートする Linux のディストリビューションを実行するサポートするプラットフォームであれば、いずれでもかまいません。LifeKeeper には、複数の重なり合うグループにサーバを設定する柔軟性があります。ただし、リカバリ可能なリソースについての重要な要件は、リソースの役割と優先順位が定義されたサーバのグループをリンクすることです。リソースに対するサーバの優先順位は、現在実行中のサーバに障害が発生した場合に、どのサーバがそのリソースを復旧するかを決定するために使用されます。最高の優先順位を示す値は 1 です。特定のリソースについて、最高の優先順位の値 (通常は 1) を持つサーバが通常、そのリソースのプライマリサーバと呼ばれます。その他のサーバは、そのリソースのバックアップサーバとして定義されます。
- 2. コミュニケーションパス。** LifeKeeper のハートビートは、LifeKeeper クラスタ内にあるサーバ間の定期的なメッセージで、主要な障害検出機能です。クラスタ内のすべてのサーバには、単純な通信障害でシステムに障害が発生しないように、冗長なハートビートコミュニケーションパス (comm パス) が必要です。2 つの独立したサブネットを使用する LAN ベース (TCP) の個別な 2 つのコミュニケーションパスが推奨されます (少なくとも 1 つのコミュニケーションパスをプライベートネットワークとして設定してください)。ただし、TCP と TTY のコミュニケーションパスの組み合わせの使用もサポートしています。TCP コミュニケーションパスは、他のシステムの通信にも使用できます。
注記: TTY コミュニケーションパスは、クラスタ内の他のサーバがアクティブかどうかを検出するためにのみ LifeKeeper で使用されます。LifeKeeper の GUI は、TCP/IP を使用して、保護するリソースに関するステータス情報を通信します。TCP コミュニケーションパスが 2 つ設定されている場合、LifeKeeper は、パブリックネットワークのコミュニケーションパスをリソースステータスの通信に使用します。このため、LifeKeeper の GUI が使用しているネットワークがダウンすると、TTY (または他の TCP) コミュニケーションパスが動作可能な場合でも、GUI には他のサーバのステータスが UNKNOWN として表示されます。
- 3. 共有データリソース。** 共有ストレージの構成では、LifeKeeper クラスタ内のサーバは同一セットのディスクに対するアクセスを共有します。プライマリサーバに障害が発生した場合、LifeKeeper は障害が発生したサーバ上にあるディスクのロック解除、および次に使用可能なバックアップサーバのディスクのロックを自

すべての LifeKeeper 設定に共通するコンポーネント

動管理します。

4. **共有通信**。LifeKeeper は TCP/IP アドレスのような通信リソースの切り替えを自動管理できるので、アプリケーションが現在どのサーバでアクティブになっているかには無関係に、ユーザはアプリケーションに接続できます。

すべての LifeKeeper 設定に共通するコンポーネント



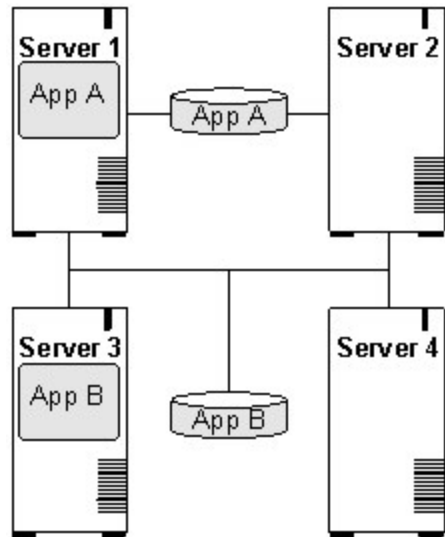
システムのグループ化の配置

リソース階層は、LifeKeeper サーバのクラスタに対して定義されます。ある階層について、各サーバに優先順位が割り当てられます。1 が最高の優先順位です。プライマリ、つまり優先順位が最高のサーバが、それらのリソースの通常動作に使用するコンピュータです。2 番目に高い優先順位を持つサーバがバックアップサーバであり、プライマリサーバに障害が発生した場合に、LifeKeeper がリソースを切り替える先のサーバです。

[アクティブ/アクティブのグループ](#)では、すべてのサーバがプロセスをアクティブに実行します。ただし、他のサーバのリソース階層ではバックアップサーバとしても機能します。[アクティブ/スタンバイのグループ](#)では、プライマリサーバは処理を実行し、バックアップサーバはプライマリサーバに障害が発生した場合に備えてスタンバイするように設定できます。スタンバイシステムは小型でパフォーマンスの低いシステムでもかまいませんが、プライマリサーバに障害が発生した場合にリソースの可用性を確保できるだけの処理能力が必要です。

共有リソースに対する物理的な接続とアクセスにより、グループ化のオプションが決まります。グループ化するサーバには、通信とハートビートパスがインストールされ、動作可能である必要があります。すべてのサーバが共有 SCSI またはファイバチャネルインターフェース経由で、ディスクリソースにアクセスできる必要があります。例えば、以下の図では、サーバ1のリソース AppA にはグループ化オプションが1つのみあります。この構成で AppA データベースへの共有アクセスを持つ他のサーバはサーバ2のみです。

ただし、サーバ3のリソース AppB は、その他3台のいずれれを含むグループにも属するように設定できます。これは、この例の共有 SCSI バスが、構成内の4台すべてのサーバに AppB データベースへのアクセスを提供しているからです。



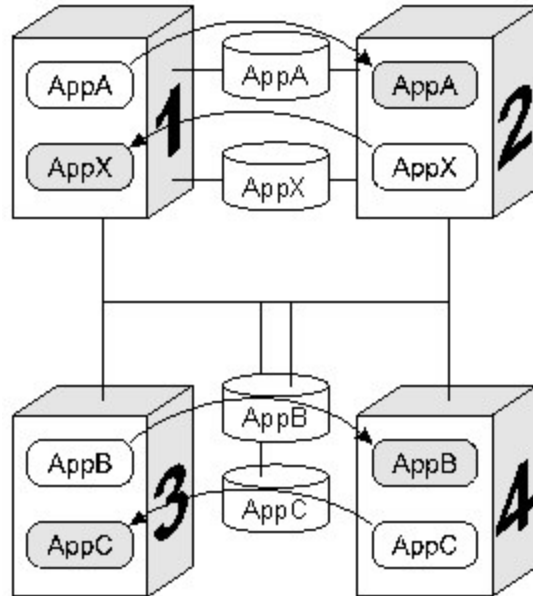
アクティブ - アクティブのグループ化

アクティブ/アクティブペアの設定では、すべてのサーバがプロセスをアクティブに実行します。また、他のサーバのリソース階層ではバックアップサーバとして機能します。

以下の設定例に、2つのアクティブ/アクティブペアのサーバを示します。サーバ1は AppA を処理していますが、サーバ2で実行中の AppX のバックアップサーバとして機能します。この逆も当てはまります。サーバ2は AppX を処理していますが、サーバ1で実行中の AppA のバックアップサーバとして機能します。サーバ3とサーバ4の間には、同じタイプのアクティブ/アクティブの関係があります。

サーバ1とサーバ2の設定と、サーバ3とサーバ4の設定は似ていますが、大きな違いがあります。AppA と AppX のアプリケーションについて、サーバ1とサーバ2のみをグループ化できます。これらのサーバのみが、共有リソースにアクセスできます。

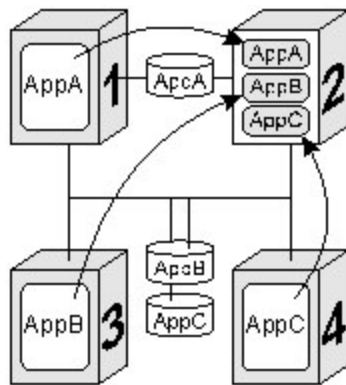
ただし、AppB と AppC は、複数のグループ化オプションを持ちます。これは、4 台のサーバすべてが AppB と AppC の共有リソースにアクセスできるからです。AppB と AppC は、第 3、第 4 のバックアップシステムとしてサーバ 1 やサーバ 2 にフェイルオーバーするように設定することもできます。



注記: LifeKeeper はディスクレベルでロックを適用するので、AppB と AppC のディスクリソースに接続する 4 つのシステムのうち、任意の時点でそれらにアクセスできるのは 1 つのみです。このため、サーバ 3 がアクティブに AppB を処理しているときには、サーバ 1、サーバ 2、およびサーバ 4 は物理的に接続していても AppB のディスクリソースを使用できません。

アクティブ - スタンバイのグループ化

アクティブ / スタンバイのペア設定では、プライマリサーバは処理を実行し、バックアップサーバはプライマリサーバに障害が発生した場合に備えてスタンバイします。スタンバイシステムは小型でパフォーマンスの低いシステムでもかまいませんが、プライマリサーバに障害が発生した場合にリソースの可用性を確保できるだけの処理能力が必要です。



スタンバイサーバは、複数のアクティブサーバにバックアップを提供します。例えば、上の図では、3つのアクティブ/スタンバイのリソースペアでサーバ2がスタンバイサーバです。LifeKeeperのリソース定義が、以下のアクティブ/スタンバイのペアの関係を指定します。

- サーバ1のAppAがサーバ2にフェイルオーバーする。
- サーバ3のAppBがサーバ2にフェイルオーバーする。
- サーバ4のAppCがサーバ2にフェイルオーバーする。

複数のアクティブ/スタンバイグループを持つ設定を検討するときには、以下の3つの重要な設定概念を念頭に置いてください。

- **ディスクの所有権**。複数の異なるアクティブなアプリケーションは、異なる複数のサーバから、同じ共有ディスクまたはLUNにあるディスクパーティションを使用できません。LifeKeeperは、ディスクまたはLUNのレベルでロックを適用します。SCSIロックが適用された場合、共有SCSIバス上にあるシステム1台のみが、ディスクまたはLUNのパーティションにアクセスできます。このため、同一ディスク上の異なるパーティションにアクセスする複数のアプリケーションは、同一サーバ上でアクティブにする必要があります。この例では、サーバ3がAppBのディスクリソースを所有し、サーバ4がAppCのリソースを所有します。
- **処理能力**。サーバ1、サーバ3、およびサーバ4に同時に障害が発生する可能性は非常に低いです。が、複数のリソース関係をサポートするスタンバイサーバを指定するときには、複数の障害が発生した場合にスタンバイサーバが重要な処理のすべてを処理できるように注意する必要があります。
- **LifeKeeperの管理**。この例では、サーバ2がその他3台のサーバをバックアップします。一般的に、LifeKeeperのデータベースを複数の論理グループで同時に管理することは望ましくありません。はじめに、予備システムと1台のアクティブなシステムとの間でリソースを作成し、次に予備システムと別のアクティブなシステムとの間、という手順を繰り返してリソースを作成する必要があります。

インテリジェントスイッチバックと自動スイッチバックの違い

デフォルトでは、リソースのスイッチバック設定はインテリジェントです。これは、そのリソースについてサーバAからサーバBにフェイルオーバーが発生すると、別の障害が発生するか、管理者がリソースを別のサーバにインテリジェントに切り替えるまで、リソースはサーバBに残ります。このため、サーバAがIn Serviceに戻った後も、リソースはサーバBで動作を続行します。この時点では、サーバAはリソースのバックアップとして機能します。

状況によっては、障害が発生したサーバが復旧したときに、リソースをそのサーバに自動でスイッチバックすることが望ましい場合があります。LifeKeeperには、前述したデフォルトのインテリジェントスイッチバック動作に代わる選択肢として、自動スイッチバックオプションがあります。このオプションは、各サーバの個々のリソース階層に設定できます。特定のサーバ上にあるリソース階層に自動スイッチバックを選択し、そのサーバに障害が発生した場合、そのリソース階層はバックアップシステムにフェイルオーバーします。障害が発生したサーバが復旧したときに、リソース階層は元のサーバに自動的にスイッチバックします。

注記:

- 自動スイッチバックのチェックは、LifeKeeperを起動したとき、またはクラスタに新しいサーバを追加したときにのみ実行されます。通常のクラスタ動作中には実行されません。
- LifeKeeperは、優先順位が上位のサーバから下位のサーバへの自動スイッチバックを実行しません。

syslog によるログの記録

LifeKeeper 8.0 から、標準の `syslog` 機能を使用してログの記録が行われます。LifeKeeper では、3 つの `syslog` の実装 (標準の `syslog`、`rsyslog`、および `syslog-ng`) をサポートしています。パッケージのインストール時には、すべての LifeKeeper ログメッセージに対して「local6」機能を使用するように `syslog` が設定されます。すべての LifeKeeper ログメッセージを `/var/log/lifekeeper.log` に送信する LifeKeeper 固有のルーティングを含むように、`syslog` 設定ファイル (`/etc/syslog-ng/syslog-ng.conf` など) が変更されます。(元の設定ファイルは、「~」で終わる同じ名前を使用してバックアップされます。)

この機能は、インストール後に `/opt/LifeKeeper/bin` にある `lklogconfig` ツールを使用して変更することができます。このツールの詳細については、LifeKeeper がインストールされているシステム上の `lklogconfig(8)` マニュアルページを参照してください。

Generic Application のリソーススクリプトが `/opt/LifeKeeper/out/log` に直接メッセージを送ると、LifeKeeper はデフォルトで `/var/log/lifekeeper.log` に ERROR レベルのログメッセージを送ります。`/etc/default/LifeKeeper` に `LOGMGR_LOGLEVEL=LK_INFO` パラメータを追加すれば、ERROR レベルを INFO レベルに変更することができます。

注意: LifeKeeper がサーバから削除されると、LifeKeeper 固有の `syslog` 設定が削除されます。

リソース階層

LifeKeeper の GUI を使用すると、あるサーバにリソース階層を作成し、次にその階層を 1 台以上のバックアップサーバに拡張できます。その後、LifeKeeper により、指定したすべてのサーバに指定階層が自動作成されます。LifeKeeper は、各サーバのデータベースで階層情報を管理します。コマンドラインインターフェースを使用する場合は、各サーバの階層を明示的に指定する必要があります。

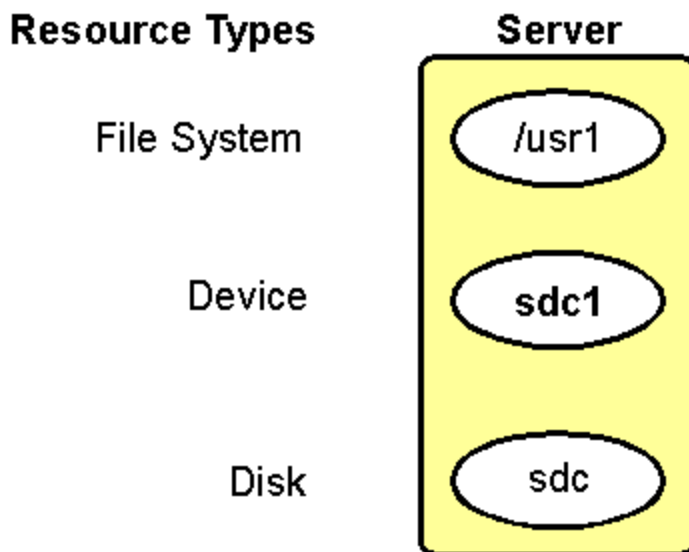
リソース階層の作成後、LifeKeeper が階層内のリソースの停止と開始を管理します。以下の関連トピックで、階層の指定作業の基本情報を説明しています。

リソースタイプ

リソースはハードウェアとソフトウェアのいずれかであり、リソースタイプ別に分類できます。LifeKeeper はファイルシステムと SCSI のリソースタイプに処理を提供し、リカバリキットは通信、RDBMS、その他のアプリケーションのリソースタイプに処理を提供します。

例えば、保護するファイルシステムの階層には、以下のタイプのリソースインスタンスが含まれます。

- **filesystem** - Linux のファイルシステムリソースオブジェクトで、マウントポイントにより識別されます。
- **device** - SCSI ディスクパーティションと仮想ディスクで、デバイスファイル名で識別されます (例: `sdcl`)。
- **disk** - SCSI ディスクまたは RAID システム論理ユニットで、SCSI デバイス名で識別されます (例: `sd`)。



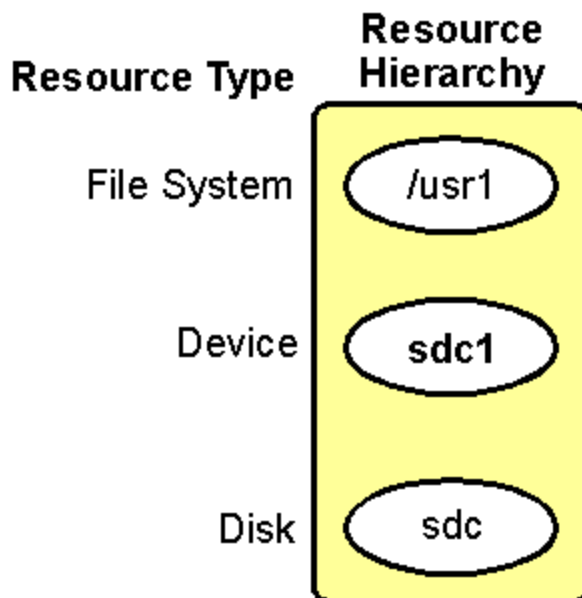
リソースの状態

状態	意味
In Service、保護 (ISP)	リソースが動作可能です。LifeKeeperのローカルリカバリが正常に動作しています。LifeKeeperのサーバ間リカバリと障害検出が動作可能です。
In Service、未保護 (ISU)	リソースは動作可能です。しかし、LifeKeeperのリソースヘルス監視は無効の状態であるため、ローカルリカバリやフェイルオーバーは行われません。 注記：ファイルシステムリソース(filesys)で保護されたファイルシステムの使用容量が90%に(閾値のデフォルト)達した時、リソースステータスをISUにすることでそれを通知します。この場合、監視処理自体は継続されます。ファイルシステムリソースと使用容量の監視では、他のリソースタイプとは違って意味合いでISUステータスを使用しています。ファイルシステムの使用容量が一度閾値を下回れば、リソースステータスはISPに戻ります。
Out of Service、障害 (OSF)	リソースが、障害により Out of Service になっています。リカバリは完了していないか、失敗しました。このリソースについて、LifeKeeperの警告機能は動作不能です。
Out of Service、障害なし (OSU)	リソースは Out of Service ですが、別のサーバからリソースを引き継ぐことができます。

状態	意味
不正 (未定義) 状態 (ILLSTATE)	この状態は、リソースインスタンスについて状態が設定されていない場合に表示されます。通常の場合では、この不正状態が長く続くことはありません。ある状態から別の状態への移行が予測されます。LifeKeeperの情報テーブルがすべて更新される前 (LifeKeeperが初めて起動するときなど) にスイッチオーバーが発生した場合に、この状態になります。

階層の関係

LifeKeeperでは、リソースインスタンス間の関係を作成できます。主な関係は依存関係で、例えばあるリソースインスタンスが動作するために、別のリソースインスタンスに依存します。リソースインスタンスと依存関係の組み合わせが、リソース階層です。



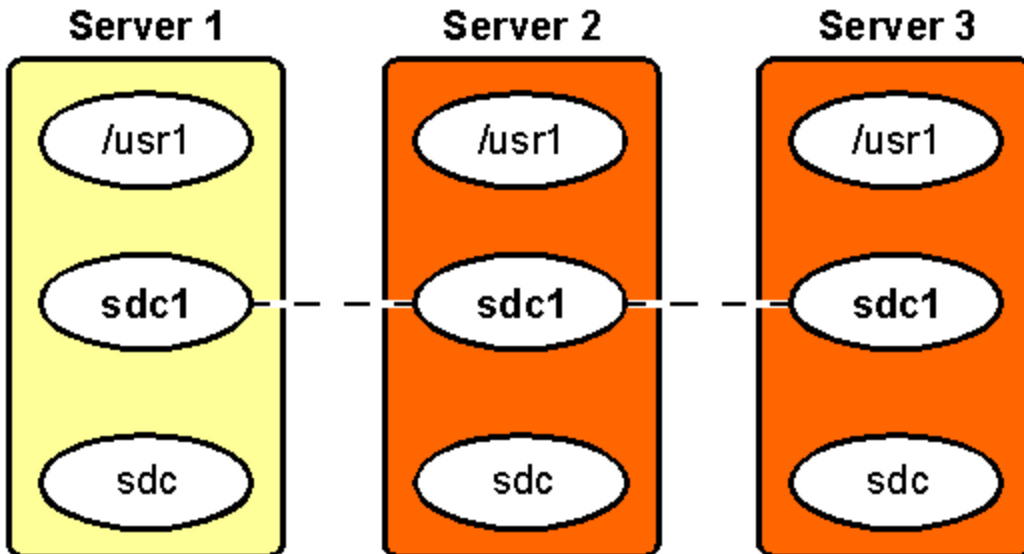
例えば、`/usr1`の動作はディスクサブシステムに依存するので、`/usr1`と、ディスクサブシステムを表すインスタンスとの間に順序付きの階層の関係を作成できます。

リソース階層により指定された依存関係は、リソースインスタンスを In Service と Out of Service にする適切な順序を LifeKeeper に示します。このリソース階層の例では、`disk` と `device` のインスタンスを正常に In Service にするまで、LifeKeeper は `/usr1` リソースを In Service にすることができません。

イクイバレンシ情報

LifeKeeper リソース階層を作成して拡張すると、そのリソース階層はプライマリサーバとセカンダリサーバの両方に存在します。ほとんどのリソースインスタンスは、1台のサーバでのみ同時にアクティブにできます。このようなリソースについて、LifeKeeper は「イクイバレンシ情報」という第2の種類関係を定義します。これにより、リソースがあるサーバで In Service になると、イクイバレンシ情報が定義されている他のサーバでは Out of Service になります。

以下の例に、各サーバのディスクパーティションのリソースインスタンス間のイクイバレンシ情報を示します。この例では、各リソースインスタンスが類似のイクイバレンシを持ちます。



リソース階層の情報

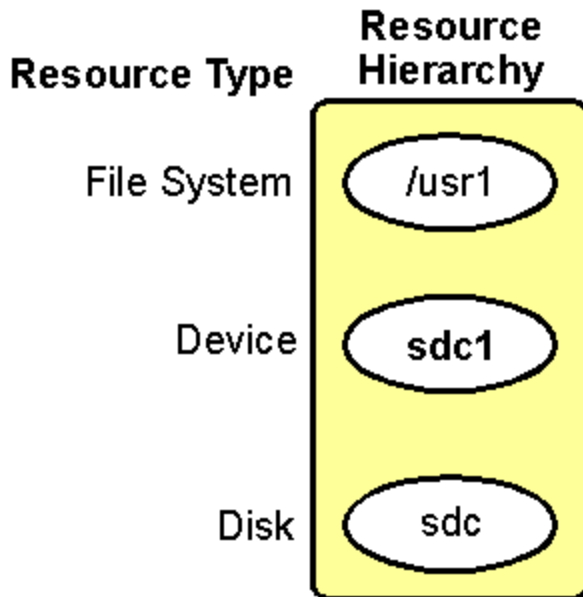
各リソースのステータスは、[ステータスの詳細表示](#)と[ステータスの簡略表示](#)で表示されます。root リソースを表す LifeKeeper のタグ名は、[TAG] 列の左端から開始され、階層内のリソースのタグ名は適切にインデントされてリソース間の依存関係を表します。

以下の例は、ステータスの簡略表示のリソース階層セクションから取ったものです(デバイスとディスクの ID は、表示領域に収まるように切り詰められています)。

LOCAL	TAG	ID	STATE	PRIO	PRIMARY
svr1	app3910-on-svr1	app4238	ISP	1	svr2
svr1	filesys4083	/jrl1	ISP	1	svr2
svr1	device2126	000...300-1	ISP	1	svr2
svr1	disk2083	000...300	ISP	1	svr2

階層の図については[リソース階層の例](#)のトピックを参照してください。詳細については、[ステータスの詳細表示](#)と[ステータスの簡略表示](#)のトピックの「リソース階層の情報」セクションを参照してください。

リソース階層の例



ステータスの詳細表示

このトピックでは、**lcdstatus** コマンドの出力例を使用してステータスの詳細表示で提供される情報のカテゴリについて説明します。この情報を表示する方法の詳細については、LCD (1M) のマニュアルページを参照してください。コマンドラインに、**man lcdstatus** または **man LCD** を入力できます。LifeKeeper の GUI で使用できるステータス情報については、[サーバーのステータスの表示](#) または [リソースのステータスの表示](#) を参照してください。

ステータスの詳細表示の例:

シャットダウンストラテジー

```
Resource hierarchies for machine "wileecoyote":
ROOT of RESOURCE HIERARCHY
apache-home.fred: id=apache-home.fred app=webserver type=apache state=ISP
initialize=(AUTORES_ISP) automatic restore to IN-SERVICE by LifeKeeper
info=/home/fred /usr/sbin/httpd
reason=restore action has succeeded
depends on resources: ipeth0-172.17.104.25,ipeth0-172.17.106.10,ipeth0-172.17.106.105
Local priority = 1
SHARED equivalency with "apache-home.fred" on "roadrunner", priority = 10
```

ステータスの詳細表示

FAILOVER ALLOWED

ipeth0-172.17.104.25: id=IP-172.17.104.25 app=comm type=ip state=ISP
initialize=(AUTORES_ISP) automatic restore to IN-SERVICE by LifeKeeper
info=wileecoyote eth0 172.17.104.25 fffffc00
reason=restore action has succeeded

these resources are dependent: apache-home.fred

Local priority = 1

SHARED equivalency with "ipeth0-172.17.104.25" on "roadrunner", priority = 10

FAILOVER ALLOWED

ipeth0-172.17.106.10: id=IP-172.17.106.10 app=comm type=ip state=ISP
initialize=(AUTORES_ISP) automatic restore to IN-SERVICE by LifeKeeper
info=wileecoyote eth0 172.17.106.10 fffffc00
reason=restore action has succeeded

these resources are dependent: apache-home.fred

Local priority = 1

SHARED equivalency with "ipeth0-172.17.106.10" on "roadrunner", priority = 10

FAILOVER ALLOWED

ipeth0-172.17.106.105: id=IP-172.17.106.105 app=comm type=ip state=ISP
initialize=(AUTORES_ISP) automatic restore to IN-SERVICE by LifeKeeper
info=wileecoyote eth0 172.17.106.105 fffffc00
reason=restore action has succeeded

These resources are dependent: apache-home.fred

Local priority = 1

SHARED equivalency with "ipeth0-172.17.106.105" on "roadrunner", priority = 10

FAILOVER ALLOWED

通信ステータスの情報

The following LifeKeeper servers are known:

machine=wileecoyote state=ALIVE

machine=roadrunner state=DEAD (eventslcm detected failure at Wed Jun 7 15:45:14 EDT 2000)

The following LifeKeeper network connections exist:

```
to machine=roadrunner type=TCP addresses=192.168.1.1/192.168.105.19
state="DEAD" priority=2 #comm_downs=0
```

[LifeKeeper のフラグ](#)

The following LifeKeeper flags are on:

shutdown_switchover

[シャットダウンストラテジー](#)

The shutdown strategy is set to: switchover.

リソース階層の情報

LifeKeeper は、リソースのステータスを root リソースから表示します。表示には、リソースのすべての依存関係についての情報が含まれます。

複数のリソースに共通する要素は、最初の root リソースの下に 1 回のみ表示されます。各リソース記述の第 1 行には、リソースタグとその後続くコロン (:) が表示されます (例: device13557:)。階層内でリソースの記述に使用できる情報要素を以下に示します。

- **id** - LifeKeeper が使用する一意のリソース識別文字列。
- **app** - アプリケーションのタイプを示します。例えば、サンプルリソースは *Web* サーバアプリケーションです。
- **type** - リソースのクラスタイプを示します。例えば、サンプルリソースは *Apache* アプリケーションです。
- **state** - リソースの現在の状態。
 - ISP — ローカルで In Service であり、保護されています。
 - ISU — In Service であり、保護されていません。
 - OSF — Out of Service であり、障害が発生しています。
 - OSU — Out of Service であり、障害はありません。
- **initialize** - リソースの初期化方法を指定します。例えば、LifeKeeper はアプリケーションのリソースをリストアしますが、ホストアダプタは LifeKeeper なしで初期化します。
- **info** - オブジェクトの `remove` と `restore` のスクリプトが使用する、オブジェクトに固有の情報があります。
- **reason** - 存在する場合、リソースが現在の状態にある原因を示します。例えば、あるアプリケーションが OSU の状態になった原因は、別のサーバでそのアプリケーションが In Service (ISP または ISU) になったからです。共有リソースは、グループ内の 1 台のサーバでのみ同時にアクティブにできます。
- **depends on resources** - 存在する場合、このリソースが依存するリソースのタグ名がリストされます。
- **these resources are dependent** - このオブジェクトに直接依存するすべての親リソースのタグ名が示されます。

- **Local priority** - このリソースについて、ターゲット サーバのフェイルオーバーの優先順位の値を示します。
- **SHARED equivalency** - このリソースが同等として定義されたりリモートリソースのリソースタグとサーバ名、およびこのリソースについてのフェイルオーバーの優先順位の値を示します。
- **FAILOVER ALLOWED** - 存在する場合、上の行で同等と指定されたりリモートサーバで LifeKeeper が動作可能であること、およびアプリケーションが障害に対して保護されていることを示します。FAILOVER INHIBITED は、LifeKeeper がシャットダウンされているかリモートサーバが停止していることにより、アプリケーションが保護されていないことを示します。

通信ステータスの情報

ステータス表示のこのセクションには、LifeKeeper が認識しているサーバとその現在の状態、および各コミュニケーションパスの情報 がリストされます。

これらの通信情報の要素は、ステータス表示にあります。

- **state** - コミュニケーションパスのステータス。通信ステータスの値は以下の値をとります。
 - ALIVE - 通常の動作中。
 - DEAD - 通常の動作をしていません。
- **priority** - コミュニケーションパスに割り当てられた優先順位の値。この項目は TCP パスについてのみ表示されます。
- **#comm_downs** - ポートに障害が発生してフェイルオーバーが発生した回数。パスの障害によりフェイルオーバーが発生するのは、障害発生時に「ALIVE」のコミュニケーションパスが他にない場合のみです。

さらに、ステータス表示では、TTY コミュニケーションパスについてのみ維持されている以下の統計値を提供できます。

- **wrpid** - 個々の TTY コミュニケーションパスが、一意の読み取りプロセスと書き込みプロセスを持ちます。wrpid フィールドには、書き込みプロセスのプロセス ID があります。書き込みプロセスは、以下の 2 つの条件のうちいずれかが発生するまでスリープ状態です。
 - ハートビートタイマの期限が切れ、書き込みプロセスにメッセージを送信させる。
 - ローカルプロセスが、LifeKeeper のメンテナンスメッセージを他のサーバに送信するように書き込みプロセスに要求する。書き込みプロセスは、関連付けられた TTY ポートを使用して、メッセージを他のシステムの TTY ポート上にある読み取りプロセスに送信します。
- **rdpid** - 個々の TTY コミュニケーションパスが、一意の読み取りプロセスと書き込みプロセスを持ちます。rdpid フィールドには、読み取りプロセスのプロセス ID があります。読み取りプロセスは、以下の 2 つの条件のうちいずれかが発生するまでスリープ状態です。
 - ハートビートタイマの期限が切れ、定義済みのハートビート間隔が期限切れになったかどうかを読み取りプロセスが判断する必要がある場合。期限切れの場合、読み取りプロセスはコミュニケーションパスに DEAD 状態のマークを付けます。これにより、ALIVE とマークされた他のコミュニケーションパスがない場合はフェイルオーバーイベントが開始されます。
 - リモートシステムの書き込みプロセスが LifeKeeper のメンテナンスメッセージを送信し、読み取りプロセスがメッセージの受信に必要なプロトコルを実行します。

- **#NAKs** - 書き込みプロセスがnegative acknowledgment (NAK)を受信した回数。NAKメッセージは、他のシステム上にある読み取りプロセスが、書き込みプロセスが送信したメッセージを受け取らず、書き込みプロセスがメッセージパケットを再送信する必要があったことを意味します。#NAKsの統計値は、回線ノイズに起因して、長期間にわたって集計できます。ただし、急激に数値が増加した場合、通信サブシステムで診断手順を実行する必要があります。
- **#chksumerr** - サーバ間のチェックサムメッセージが一致しなかった回数。この統計値は、回線ノイズに起因して、長期間にわたって集計できます。ただし、急激に数値が増加した場合、通信サブシステムで診断手順を実行する必要があります。
- **#incmpltmes** - 受信メッセージパケットが予測サイズに一致しなかった回数。不一致の回数が多い場合、コミュニケーションパスに関連付けられたハードウェアポートで診断手順の実行が必要な可能性があります。
- **#noreply** - 書き込みプロセスが肯定応答の待機中にタイムアウトし、メッセージを再送信しなければならなかった回数。肯定応答がない場合、サーバの過負荷、またはサーバの障害を意味することがあります。
- **#pacresent** - 読み取りプロセスが同一パケットを受診した回数。これは、送信サーバの書き込みプロセスがタイムアウトし、同一メッセージを再送信する場合に発生することがあります。
- **#pacoutseq** - 読み取りプロセスが、順序が不正のパケットを受診した回数。このフィールドの値が大きい場合、メッセージパケットの脱落を示すことがあり、通信サブシステムで診断手順の実行が必要な可能性があります。
- **#maxretrys** - 特定のメッセージについて、再送信の最大回数を超えたときに増加する指標 (NAKとnoreplyのメッセージ)。#maxretrysフィールドの値が大きい場合、通信サブシステムで診断手順を実行する必要があります。

LifeKeeper のフラグ

ステータスの詳細表示の後部近くに、システムのフラグセットがあります。共通タイプは、プロセスのロックが動作を完了するまで他のプロセスを確実に待機させるために使用するLCDのロックフラグです。LCDのロックの標準フォーマットは以下のとおりです。

```
!action!processID!time!machine:id.
```

一般的なLCDのロックフラグの例を示します。

- `!action!02833!701236710!server1:filesys`。ファイルシステム階層を作成すると、このフォーマットでステータス表示にフラグが生成されます。`filesys`の指定は、他のアプリケーションリソース階層では別のリソースタイプである場合も、一般的なアプリケーションやユーザ定義アプリケーションでは`app`である場合もあります。
- 他の代表的なフラグとして、`!nofailover!machine`、`!notamode!machine`、`shutdown_switchover`などがあります。`!nofailover!machine`と`!notamode!machine`のフラグは、LifeKeeperが作成と削除を行う内部の一時フラグで、サーバのフェイルオーバーを制御します。`shutdown_switchover`フラグは、このサーバのシャットダウンストラテジーが`switchover`に設定されたことを示し、サーバのシャットダウンによりスイッチオーバーが発生します。使用可能なフラグの詳細については、依存関係の作成方法については、LCDI-flag (1M)を参照してください。

シャットダウンストラテジー

ステータスの詳細表示の最後の項目は、このシステム用に選択された LifeKeeper のシャットダウンストラテジーを示します。詳細については、[サーバのシャットダウンストラテジーの設定](#)を参照してください。

ステータスの簡略表示

このトピックでは、`lcdstatus -e` コマンドの出力例を使用して、ステータスの簡略表示で提供される情報のカテゴリについて説明します。この情報を表示する方法の詳細については、LCD (1M) のマニュアルページを参照してください。コマンドラインに、`man lcdstatus` または `man LCD` を入力できます。LifeKeeper の GUI で使用できるステータス情報については、[サーバのステータスの表示](#) または [リソースのステータスの表示](#) を参照してください。

ステータスの簡略表示の例:

[リソース階層の情報](#)

BACKUP	TAG	ID	STATE	PRIO	PRIMARY
svr1	appfs3910-on-svr1	appfs4238	ISP	1	svr2
svr1	filesys4083	/jrl1	ISP	1	svr2
svr1	device2126	000...300-1	ISP	1	svr2
svr1	disk2083	000...300	ISP	1	svr2

[通信ステータスの情報](#)

MACHINE	NETWORK	ADDRESSES/DEVICE	STATE	PRIO
svr1	TCP	100.10.1.20/100.11.1.21	ALIVE	1
svr1	TTY	/dev/ttyS0	ALIVE	--

リソース階層の情報

LifeKeeper は、各リソースのステータスを表示します。root リソースを表す LifeKeeper のタグ名は、[TAG] 列の左端から開始され、階層内のリソースのタグ名は適切にインデントされてリソース間の依存関係を表します。

BACKUP 列は、フェイルオーバの優先順序内で、このステータス表示の対象システムの次にあるシステムを示します。指定したリソースについて、ターゲットシステムが優先順位の最も低いシステムである場合、そのリソースの **BACKUP** 列にはダッシュ (----) が表示されます。

- **TAG 列** - リソースの root タグがあります。
- **ID 列** - 各リソースの識別文字列があります。
- **STATE 列** - 各リソースの現在の状態があります ([リソースの状態](#)を参照)。

- **PRIO 列** -各リソースについて、ローカルサーバのフェイルオーバーの優先順位の値があります。
- **PRIMARY 列** -各リソースについて、優先順位が最高のサーバ名があります。

通信ステータスの情報

表示のこのセクションには、ターゲットシステムで定義された各コミュニケーションパスのリストがあります。各パスについて、以下の情報が表示されます。

- **MACHINE** - コミュニケーションパスのリモートサーバ名。
- **NETWORK** - コミュニケーションパスのタイプ (TCP または TTY)。
- **ADDRESSES/DEVICE** - コミュニケーションパスの IP アドレスまたはデバイス名のペア。
- **STATE** - コミュニケーションパスの状態 (ALIVE または DEAD)。
- **PRIO** - TCP パスの場合、パスに割り当てられた優先順位。TTY パスの場合、優先順位が割り当てられていないので、この列にはダッシュ (---) が表示されます。

障害検出とリカバリのシナリオ

障害検出とリカバリを実行するために、LifeKeeper のさまざまなコンポーネントがどのように連携しているかを調べるには、3つのタイプのリカバリシナリオを説明する以下のトピックを参照してください。

IP ローカルリカバリ

SIOS では、バックアップインターフェースが必要な場合、すべての LifeKeeper リリースに含まれる標準 Linux の NIC ボンディングメカニズムを使用してボンディングしたインターフェースを使用することを推奨しています。LifeKeeper のリリース 7.4.0 から、ボンディングしたインターフェースがサポートする唯一の方法になりました。7.4.0 以前のリリースでは、後述の IP キットのバックアップインターフェース機能を使用できます。

IP ローカルリカバリ機能を使用すると、IP Recovery Kit が障害を検出したときに、LifeKeeper は、保護している IP アドレスを、設定されているインターフェースから同一サーバ上の別のインターフェースに移動できます。ローカルリカバリはオプションのバックアップ方式を提供するので、サーバで特定のインターフェースに障害が発生した場合、保護している IP アドレスをバックアップインターフェースで動作可能にできます。このため、アプリケーション/リソース階層全体がバックアップサーバにフェイルオーバーすることを防ぐことができます。

ローカルリカバリのシナリオ

IP ローカルリカバリを使用すると、サーバ上で LifeKeeper が保護する各 IP アドレスについて、バックアップネットワークインターフェースを 1 つ指定できます。バックアップインターフェースが正しく動作するためには、プライマリインターフェースと同じ物理ネットワークに接続する必要があります。システム管理者は、有効なインターフェースが選択されていることを確認する必要があります。バックアップインターフェースをあるサーバに指定し、クラスタ内の他のサーバには指定しないことには、正当性があります。選択されたあるサーバ上のバックアップインターフェースは、他のサーバ上のバックアップの選択に影響を与えません。

IP Recovery Kit によって IP アドレスの障害が検出されると、結果として生じる障害によって IP ローカルリカバリスクリプトが起動されます。LifeKeeper は最初に、その IP アドレスを現在のネットワークインターフェース上で In Service に戻そうとします。この動作に失敗した場合、LifeKeeper はリソースインスタンスをチェックして、使用可

コマンドラインの操作

可能なバックアップインターフェースの有無を調べます。使用可能なバックアップインターフェースがある場合、IP アドレスをバックアップインターフェースに移動しようとします。ローカルリカバリの試行がすべて失敗した場合、LifeKeeper は IP アドレスとすべての依存リソースをバックアップサーバにフェイルオーバーします。

バックアップインターフェース名は、IP リソースインスタンスの情報フィールドに指定できます。情報フィールドの値はスペースで区切り、プライマリサーバ名、ネットワークインターフェース名、IP アドレス、ネットマスク、バックアップインターフェース名の順に指定します。例を示します。

```
ServerA eth0 172.17.106.10 fffffc00 eth1
```

バックアップインターフェースを設定しない場合、5 番目のフィールド値を **none** に設定してください。

保護している IP アドレスがバックアップインターフェースに移動すると、2 番目と 5 番目のフィールド値が入れ替えられ、元のバックアップインターフェースがプライマリになり、元のプライマリインターフェースがバックアップになります。この結果、LifeKeeper の起動時、スイッチオーバー時、およびフェイルオーバー時には、LifeKeeper は常に最後に設定されたインターフェースで IP アドレスを In Service にしようとします。

コマンドラインの操作

LifeKeeper for Linux v3.01 以降では、既存の IP リソースインスタンスにバックアップインターフェースを追加したり削除したりする機能は、コマンドラインユーティリティとして提供されています。この機能は、lkipbu ユーティリティが提供します。コマンドと構文は以下のとおりです。

```
lkipbu [-d machine] -{a|r} -t tag -f interface
```

このインスタンスについてバックアップインターフェースがすでに定義済みの場合、または不正なインターフェース名が指定された場合、add 動作 (-a オプションで指定) は失敗します。指定したインターフェースが、この DataKeeper の現在のバックアップインターフェースでない場合、削除動作 (-r オプションで指定) は失敗します。

コマンドライン操作で、IP アドレスをバックアップインターフェースに手動で移動することもできます。この操作は、以下の構文で -m オプションにより指定します。

```
lkipbu [-d machine] -m -t tag
```

このインスタンスについてバックアップインターフェースが設定されていない場合、この操作は失敗します。指定したリソースインスタンスが現在 In Service である場合、現在のインスタンスから IP アドレスを設定解除する ipaction remove 動作、および IP アドレスをバックアップインターフェースに設定する ipaction restore 動作を使用して、移動が実行されます。移動後、execute_broadcast_ping の機能を使用して新しいインターフェース上にあるアドレスの動作が確認され、正常に動作している場合は、IP リソースインスタンスの INFO フィールドにあるインターフェースの値が入れ替えられます。このコマンドの実行時に、指定した IP リソースインスタンスが Out of Service である場合、INFO フィールドのプライマリとバックアップのインターフェースの値が単純に入れ替えられます。

lkipbu ユーティリティには、指定した IP リソースインスタンスについて現在指定されているプライマリとバックアップのインターフェース、およびプライマリインターフェース上のリソースの状態 (動作中または停止) を取得するオプションもあります。この操作は、以下の構文で -s オプションにより指定します。

```
lkipbu [-d machine] -s -t tag
```

出力は、以下のようになります。

```
IP address: 172.17.106.10
```

Netmask: 255.255.252.0

Primary interface: eth0 (up)

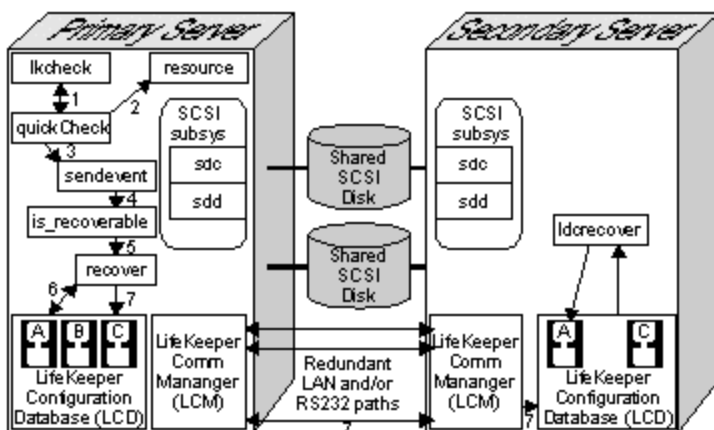
Backup interface: eth1

詳細については、Ikipbu(8)のマニュアルページを参照してください。

リソースのエラーリカバリのシナリオ

LifeKeeper は、LifeKeeper が保護するリソースのステータスと健全性をチェックするリアルタイムデーモンモニタ **Ikcheck** を装備しています。In Service の各リソースについて、**Ikcheck** が定期的にそのリソースタイプの **quickCheck** スクリプトを呼び出します。**quickCheck** スクリプトがリソースのクイック健全性チェックを実行し、リソースが障害のある状態にあると判断すると、**quickCheck** スクリプトはイベント通知メカニズム **sendevent** を呼び出します。

以下の図に、**Ikcheck** がプロセスを開始したときのリカバリプロセスの作業を示します。



1. **Ikcheck** が実行されます。デフォルトでは、**Ikcheck** プロセスは2分ごとに実行されます。**Ikcheck** が動作すると、システムで In Service の各リソースについて適切は **quickCheck** スクリプトを呼び出します。
2. **quickCheck** スクリプトがリソースをチェックします。**quickCheck** スクリプトが実行するチェックの内容は、各リソースタイプによって異なります。通常、スクリプトは、リソースのクライアントの動作をシミュレートして、予測した応答を受信するかどうかを確認することにより、目的の作業を実行するためにリソースが使用可能かどうかを単純に確認します。
3. **quickCheck** スクリプトが **sendevent** を呼び出します。**quickCheck** スクリプトが、リソースが障害のある状態にあると判断した場合、**sendevent** を呼び出して、適切なクラスとタイプを持つイベントを開始します。
4. リカバリ手順の検索。システムイベント通知メカニズム **sendevent** は、はじめに、イベントタイプまたはコンポーネントについて、LCD がリソースまたはリカバリを持つかどうかを判断しようとしています。この判断を行うた

めに、is_recoverable プロセスは LCD のリソース階層をスキャンして、イベントに対応するリソースインスタンス(この例では filesys の名前)を検索します。

次の手順の動作は、スキャンでリソースレベルのリカバリ手順が検出されたかどうかによって異なります。

- 検出されない場合。リカバリ手順が見つからない場合、is_recoverable は **sendevent** に戻り、**sendevent** は基本イベント通知を続行します。
 - 検出された場合。スキャンでリソースが検出された場合、is_recoverable はリカバリプロセスをバックグラウンドに運びます。is_recoverable プロセスが戻り、**sendevent** が基本イベント通知を続行します。推奨フラグ「-A」を基本警告イベント応答スクリプトに渡し、LifeKeeper がリカバリを実行することを示します。
5. リカバリプロセスが開始されます。リカバリが続行していると仮定して、is_recoverable はリカバリプロセスを開始し、はじめにローカルリカバリを試行します。
 6. ローカルリカバリが試行されます。インスタンスが検出された場合、リカバリプロセスは、LCD 内のリソース階層にアクセスし、階層ツリーからイベントに回答する方法を知っているリソースを検索して、ローカルリカバリを試行します。各リソースタイプについて、イベントクラスにちなむ名前を持つサブディレクトリ(そのイベントタイプのリカバリスクリプトを持つ)を含むリカバリサブディレクトリを検索します。

リカバリプロセスが、リソース階層で障害が発生しているリソースから上方向に最も離れたリソースに関連付けられている、リカバリスクリプトを実行します。リカバリスクリプトが正常に完了した場合、リカバリは停止します。スクリプトが失敗した場合、次のリソースに関連付けられたスクリプトが実行され、リカバリスクリプトが正常に完了するか、障害が発生したインスタンスに関連付けられたリカバリスクリプトが試行されるまで続行されます。

ローカルリカバリが正常に完了した場合、リカバリは停止します。

7. サーバ間のリカバリが開始されます。ローカルリカバリに失敗した場合、イベントはサーバ間のリカバリにエスカレートします。
8. リカバリが続行されます。ローカルリカバリに失敗しているため、リカバリプロセスは失敗したインスタンスを *Out-of-Service-FAILED* (OSF) 状態にマークし、この失敗したリソースに依存するすべてのリソースを *Out-of-Service-UNIMPAIRED* (OSU) 状態にマークします。リカバリプロセスは次に、障害が発生したリソース、または障害が発生したリソースに依存するリソースが、他のシステム上にあるリソースとイクイバレンシー情報を持っているかどうかを判断し、優先順位が最高の動作可能なサーバを選択します。同時にアクティブにできるイクイバレンシー情報を持つリソースは 1 つのみです。

イクイバレンシー情報が存在しない場合、リカバリプロセスは停止します。

イクイバレンシー情報が検出されて選択された場合、LifeKeeper はサーバ間のリカバリを開始します。リカバリプロセスが LCM 経由で、イクイバレンシー情報を持つリソースを持つ選択されたバックアップシステムの LCD プロセスにメッセージを送信します。これは、LifeKeeper がサーバ間のリカバリを試行することを意味します。

9. **lcdrecover** プロセスが転送を調整します。バックアップサーバの LCD プロセスが **lcdrecover** プロセスを運び、同等リソースの転送を調整します。
10. バックアップサーバのアクティブ化。**lcdrecover** プロセスが同等のリソースを検出し、そのリソースが Out of Service のリソースに依存しているかどうかを判断します。**lcdrecover** が、必要な各リソースについて restore スクリプト(リソースリカバリ動作スクリプトの一部)を実行し、リソースを In Service にします。

バックアップサーバでリソースをリストアすることにより、プライマリシステムからより多くの共有リソースを転送することが必要になる場合があります。プライマリシステムとの間で、プライマリサーバ上でのサービスから削除する必要があるリソースを示すメッセージが送受信され、次に選択したバックアップサーバで In Service になり、重要なアプリケーションのすべての機能が提供されます。この動作は、転送する追加の共有リソースがなくなり、バックアップで必要なすべてのリソースインスタンスがリストアされるまで続行されます。

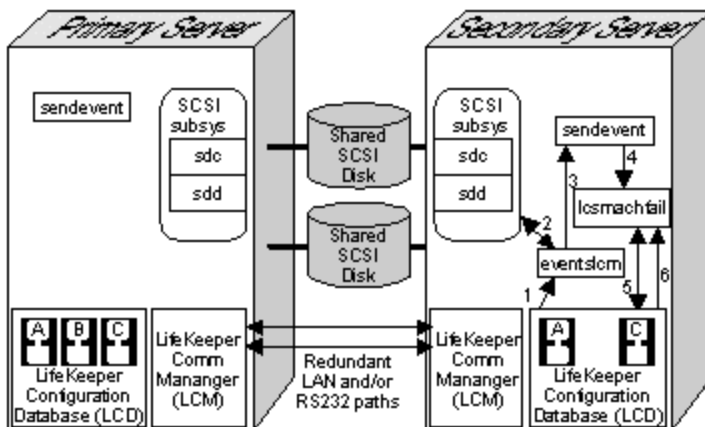
サーバの障害リカバリのシナリオ

LifeKeeper Communications Manager (LCM) には、2つの機能があります。

- メッセージング。LCM は、LifeKeeper がリカバリ、設定、または監査の実行を行うときに送信するメッセージの経路として機能します。
- 障害検出。また、LCM には、サーバに障害が発生しているかどうかを検出する役割もあります。

LifeKeeper には、構成内の各サーバに、ペアのサーバが動作していることを定期的に通知する組み込みのハートビート信号があります。あるサーバが、いずれかのコミュニケーションパス経由でハートビートメッセージを受信しなかった場合、LifeKeeper はそのパスを DEAD としてマークします。

以下の図に、LCM ハートビートメカニズムがサーバの障害を検出したときのリカバリ作業を示します。



以下の手順では、上の図で、LifeKeeper があるサーバのすべての通信接続を DEAD としてマークした場合のリカバリシナリオを説明します。

1. LCM が **eventslcm** を起動します。LifeKeeper がすべてのコミュニケーションパスを DEAD としてマークすると、LCM は **eventslcm** プロセスを開始します。

eventslcm プロセスを停止する活動は1つのみです。

- コミュニケーションパスが、アクティブである。いずれかのコミュニケーションパスがハートビート信号の送信を再開した場合、LCM は **eventslcm** プロセスを停止します。

通信障害に起因するフェイルオーバーやシステムの障害を防止するために、各ペアのサーバ間に、物理的に独立した冗長なコミュニケーションパスを2つ以上設定することが重要です。

2. `sendevent` へのメッセージ送信。`eventslicm` は、`sendevent` をイベントタイプ `machfail` で呼び出してシステム障害アラームを送信します。
3. `sendevent` がフェイルオーバーリカバリを開始します。`sendevent` プログラムが、LifeKeeper がシステム障害イベントを処理できることを判断し、LifeKeeper フェイルオーバーリカバリプロセス `lcmachfail` を実行します。
4. `lcmachfail` のチェック。`lcmachfail` プロセスがはじめに、応答していないサーバがシャットダウンしていないことを確認します。システム障害の発生前に、他のシステムが正常にシャットダウンしている場合、フェイルオーバーは禁止されます。次に、`lcmachfail` は、障害が発生したシステムと同等の共有を持つリソースをすべて特定します。これが、リカバリの関与ポイントです。
5. `lcmachfail` がリソースをリストアします。`lcmachfail` が、障害が発生したシステムと同等のイクイバレンシ情報を持つ、バックアップサーバ上のリソースをすべて特定します。また、バックアップサーバが該当するリソースが構成されている、優先順位が最高のアクティブなサーバであるかどうかを判断します。すべてのバックアップサーバがこのチェックを実行するので、1台のサーバのみが階層のリカバリを試行します。このチェックに合格した個々の同等リソースについて、`lcmachfail` が、関連付けられたリストアッププログラムを呼び出します。次に、`lcmachfail` は、リストアしたリソースに依存する各リソースもリストアします。これは、バックアップサーバ上の階層全体がサービス中になるまで続行されます。

インストールと設定

SPS for Linux のインストール

SPS for Linux ソフトウェアのインストールに関する詳細な手順については、SPS for Linux インストールガイドを参照してください。その他の情報については、SPS for Linux リリースノートを参照してください。

SPS for Linux の設定

SPS 環境がインストールされると、クラスタ内の各サーバ上で SPS ソフトウェアを設定することができます。以下の **SPS 設定手順** トピックに記載の手順に従ってください。ここでは、詳細情報を含む各トピックへのリンクが記載されています。

SPS の設定手順

SPS インストールガイドの説明に従って SPS 環境をインストールした場合、クラスタ内の各サーバで SPS ソフトウェアを起動し、設定することができます。

詳細を説明するトピックへのリンクを含む以下の手順を実行してください。以下の手順は、クラスタ内の各サーバで実行します。

1. 次のコマンドを root として実行して LifeKeeper を起動します。

```
/etc/init.d/lifekeeper start
```

このコマンドによって、管理対象のサーバ上のまだ起動していないすべての LifeKeeper デーモンプロセスを起動します。

LifeKeeper の起動および停止の詳細については、[LifeKeeper の起動](#) および [LifeKeeper の停止](#) を参照してください。

2. [TTY 通信接続をセットアップ](#) します。LifeKeeper のハートビート用に TTY 通信 パスを利用する場合は、ハートビート用の物理的な接続をセットアップする必要があります。
3. GUI を設定します。GUI の設定には多くのタスクが含まれます。[GUI の実行準備](#) の [LifeKeeper GUI - 概要](#) トピックから始めてください。詳細な手順については、[GUI の実行準備](#) に網羅されたリンクの順番に従ってください。

注記: LifeKeeper GUI を初めて実行すると、QuickStart ボタンが表示され、これを押すと LifeKeeper のリソースの設定を案内する手順とリンクを含むウィンドウが開きます。QuickStart Configuration Assistant は [\[Help\] メニュー](#) からいつでもアクセスできます。

4. [コミュニケーションパスを作成](#) します。LifeKeeper の保護を有効にする前に、コミュニケーションパス (ハートビート) の定義を作成する必要があります。
5. 以下の設定作業を任意で実行します。
 - [サーバのシャットダウンストラテジーの設定](#)

- [手動フェイルオーバー確認オプションの設定](#)
 - [LifeKeeper ハートビートの調整](#)
 - [SNMP イベント転送の設定](#)
 - [イベントメール通知の設定](#)
 - クラスタで [STONITH](#) デバイスを使用する場合は、STONITH デバイスを制御するスクリプトを作成し、LifeKeeper の適切なイベントディレクトリに配置します。
6. SPS でアプリケーションを保護する準備ができました。以降の手順は、オプションの SPS Recovery Kit のいずれかを使用するかどうかによって異なります。
- SPS Recovery Kit を使用する場合、リソース階層の作成と拡張の手順については Recovery Kit に関連するドキュメンテーションを参照してください。
 - 関連する Recovery Kit がないアプリケーションを使用する場合、2 通りの選択肢があります。
 - シンプルなアプリケーションの場合、アプリケーションと LifeKeeper との間のインターフェースの作成方法を慎重に検討してください。LifeKeeper Core に含まれる [Generic Application Recovery Kit](#) を使用して保護することもできます。

TTY 接続のセットアップ

LifeKeeper のハートビート用に TTY コミュニケーションパスを利用する場合は、ハートビート用の物理的な接続をセットアップする必要があります。単一の通信障害による誤ったフェイルオーバーを抑止するためには、複数のコミュニケーションパスが必要です。2 つ以上の LAN ベース (TCP) のコミュニケーションパスも使用する必要があります。

使用する各サーバのシリアルポートにシリアルハートビート用の TTY ケーブルを接続します。

1. 次のコマンドを実行してシリアルパスをテストします。

```
/opt/LifeKeeper/bin/portio -r -p port -b baud
```

ここで、

- **baud** は、シリアルパス用に選択したボーレート (通常は 9600)
- **port** は、サーバ 1 でテスト中のシリアルポート。例えば、`/dev/ttyS0`

これでサーバ 1 は、サーバ 2 からの入力を待っている状態です。

2. サーバ 2 で `portio` コマンドを実行します。ペアの 2 番目のシステムで次のコマンドを実行します。

```
echo Helloworld | /opt/LifeKeeper/bin/portio -p port -b baud
```

ここで、

- **baud** は、サーバ 1 に合わせて選択した同じボーレート
- **port** は、サーバ 2 でテスト中のシリアルポート。例えば、`/dev/ttyS0`。

3. コンソールを確認します。コミュニケーションパスが正常に動作する場合、サーバ1のコンソールには「HelloWorld」が表示されます。表示されない場合は、診断、修正作業を終えてからLifeKeeperの設定を続けてください。

SNMP による LifeKeeper イベント転送

SNMP による LifeKeeper イベント転送の概要

SNMP (Simple Network Management Protocol) は、ネットワークを管理するための、デバイスに依存しないフレームワークです。ネットワーク上のデバイスは、デバイスのベンダーが提供する MIB (Management Information Base) 変数によって記述されます。ネットワーク内の各ノード上では SNMP エージェントが実行され、ネットワークマネージャノードと通信を行います。ネットワークマネージャは、エージェントに対するクエリで MIB 変数の値を取得、設定することにより、エージェントノードを監視、制御します。エージェントは、トラップと呼ばれるメッセージを非同期に生成して例外イベントの発生をマネージャに知らせることもできます。SNMP (Simple Network Management Protocol) を使用してネットワークを監視および管理するアプリケーションは多数提供されています。

LifeKeeper のイベント通知機能では、特定のイベントが起きたときに通知を受信するアプリケーションを登録することができます (sendevent(5) マニュアルページを参照)。LifeKeeper は、LifeKeeper の動作を監視するサードパーティのネットワーク管理コンソールに向けて LifeKeeper の重要なイベントに関する SNMP トラップ通知を送信するように簡単に設定できます。

SNMP トラップを受信するリモート管理コンソールは、最初にそのシステムの管理用ソフトウェアを使用して設定する必要があります。LifeKeeper は、外部の SNMP の設定機能を提供していません。リモート管理サーバは、通常、LifeKeeper クラスターの外側に配置されます (つまり LifeKeeper のノードではありません)。

LifeKeeper イベントテーブル

以下の表では、LifeKeeper のイベントのリストと関連付けられているトラップ番号を示しています。オブジェクト ID (OID) は、プレフィックスとそれに続く個別のトラップ番号から、次のフォーマットで構成されます。

```
prefix.0.specific trap number
```

プレフィックスは「.1.3.6.1.4.1.7359」であり、これは MIB ツリーで **iso.org.dod.internet.private.enterprises.7359** に展開されます (7359 は、SteelEye (SIOS Technology) の企業番号です。LifeKeeper を表す「1」をこれに続けます)。例えば、「LifeKeeper Startup Complete」イベントは次の OID を生成します: **.1.3.6.1.4.1.7359.1.0.100**。

LifeKeeper イベント/説明	トラップ番号	オブジェクト ID
LifeKeeper Startup Complete LifeKeeper が起動したノードから送信されます	100	.1.3.6.1.4.1.7359.1.0.100
LifeKeeper Shutdown Initiated LifeKeeper のシャットダウンを開始したノードから送信されます	101	.1.3.6.1.4.1.7359.1.0.101

LifeKeeper Shutdown Complete LifeKeeper のシャットダウンを完了したノードから送信されます	102	.1.3.6.1.4.1.7359.1.0.102
LifeKeeper Manual Switchover Initiated on Server 手動スイッチオーバーを要求したノードから送信されます	110	.1.3.6.1.4.1.7359.1.0.110
LifeKeeper Manual Switchover Complete - recovered list 手動スイッチオーバーを完了したノードから送信されます	111	.1.3.6.1.4.1.7359.1.0.111
LifeKeeper Manual Switchover Complete - failed list 手動スイッチオーバーに失敗したクラスタ内の各ノードから送信されます	112	.1.3.6.1.4.1.7359.1.0.112
LifeKeeper Node Failure Detected for Server クラスタ内のノードに障害が発生したときに、クラスタ内の各ノードから送信されます	120	.1.3.6.1.4.1.7359.1.0.120
LifeKeeper Node Recovery Complete for Server - recovered list 障害ノードからのリソースをリカバリしたクラスタ内の各ノードから送信されます	121	.1.3.6.1.4.1.7359.1.0.121
LifeKeeper Node Recovery Complete for Server - failed list 障害ノードからのリソースのリカバリに失敗したクラスタ内の各ノードから送信されます	122	.1.3.6.1.4.1.7359.1.0.122
LifeKeeper Resource Recovery Initiated リソースをリカバリしているノードから送信されます。リカバリが完了したか失敗したかを示す 131 または 132 トラップが必ずこれに続きます。	130	.1.3.6.1.4.1.7359.1.0.130
LifeKeeper Resource Recovery Failed リカバリを試みたリソースを稼働できなかったときに、トラップ 130 を送信したノードから送信されます	131*	.1.3.6.1.4.1.7359.1.0.131
LifeKeeper Resource Recovery Complete リソースのリカバリが完了したときに、トラップ 130 を送信したノードから送信されます	132	.1.3.6.1.4.1.7359.1.0.132
LifeKeeper Communications Path Up ノードへのコミュニケーションパスが確立されました	140	.1.3.6.1.4.1.7359.1.0.140
LifeKeeper Communications Path Down ノードへのコミュニケーションパスがダウンしました	141	.1.3.6.1.4.1.7359.1.0.141
トラップ PDU に追加情報を含めるために以下の変数が使用され ます。		

Trap message	すべての のトラップ	.1.3.6.1.4.1.7359.1.1
Resource Tag	130	.1.3.6.1.4.1.7359.1.2
Resource Tag	131	.1.3.6.1.4.1.7359.1.2
Resource Tag	132	.1.3.6.1.4.1.7359.1.2
List of recovered resources	111	.1.3.6.1.4.1.7359.1.3
List of recovered resources	121	.1.3.6.1.4.1.7359.1.3
List of failed resources	112	.1.3.6.1.4.1.7359.1.4
List of failed resources	122	.1.3.6.1.4.1.7359.1.4

* 複数のバックアップサーバでリカバリに失敗すると、このトラップは複数回表示されることがあります。

LifeKeeper イベント転送の設定

前提条件

SNMP のイベント転送機能は、LifeKeeper Core の機能の一部として含まれているので、LifeKeeper の追加パッケージをインストールする必要はありません。ただし、LifeKeeper イベントのトラップ通知を生成する LifeKeeper の各ノードに SNMP ソフトウェアがインストールされている必要があります。LifeKeeper は、この SNMP トラップユーティリティを使ってトラップを生成します。このユーティリティは、ほとんどの Linux ディストリビューションで `snmp-utils` パッケージによって提供されています (SuSE では `snmp` と呼ばれます)。

以前のバージョン (4.1 以前) の `snmp` の実装では、`defCommunity` デイレクティブがサポートされていないため、トラップは「public」コミュニティストリングを使用して送信されます。

LifeKeeper のノードで SNMP エージェント `snmpd` を起動しておく必要はありません。

ネットワーク管理コンソール上のトラップハンドラおよびトラップメッセージに対するハンドラの応答に関する設定は、LifeKeeper の本機能が提供する範囲ではありません。必要な手順については、お使いのシステム管理ツールが提供するドキュメンテーションを参照してください。

設定作業

LifeKeeper SNMP イベント転送を設定するには、以下の作業を実施します。SNMP トラップを生成する LifeKeeper クラスターの各ノードにおいて、最後の手順以外のすべてを繰り返す必要があります。

1. 上述の `snmptrap` ユーティリティが利用できることを確認します。
2. SNMP トラップを受信するネットワーク管理ノードを指定します。指定するには、コマンドラインを使用するか、`/etc/default/LifeKeeper` ファイルを編集します。DNS の問題に影響されないように、ドメイン名ではなく IP アドレスを指定してください。

設定の確認

- コマンドラインからは、`lk_configsnmp` を使用します (詳細については、`lk_configsnmp (1M)` のマニュアルページを参照)。このユーティリティでは、IP アドレスのみ使用できます。
 - または、デフォルトファイル `/etc/default/LifeKeeper` を編集して IP アドレスを追加します。 `LK_TRAP_MGR=` エントリを見つけて「=」の右側に IP アドレスを入力します (「=」の前後にはスペースを入れません)。
3. `defCommunity` をサポートとしない SNMP 実装の以前のバージョンをお使いの場合は、このステップを飛ばしてください。トラップは「public」コミュニティストリングを使用して送信されます。新しいバージョンの場合は次の手順を実行します。

`/usr/share/snmp/snmp.conf` にデフォルトのコミュニティを指定してください。このファイルが存在しない場合は、十分な制限付きの権限で作成します。ディレクティブ `defCommunity` を値と共に追加します。これにより、トラップの送信時に SNMP バージョン 2c のコミュニティストリングが指定されます。例えば、以下のような行を追加します。

```
defCommunity myCommunityString
```

この設定ファイルの詳細については、`snmp.conf` マニュアルページを参照してください。

4. リモート管理コンソール上で、LifeKeeper のイベントから送られて来るトラップ OID を検出し応答するために必要な設定手順をすべて実行します。管理ノードが Linux サーバの場合、この機能の検証を開始するために最低限必要なことは、`-f -lo` オプション (メッセージを `stdout` に出力) を指定して `snmptrapd` デモンを開始することです。

設定の確認

設定が正常に動作することを確認するには、LifeKeeper の処理を実行します (例えば、LifeKeeper を開始または停止する、または LifeKeeper GUI を使用して、あるリソースを手動で in service にするなど)。管理コンソールでトラップメッセージを受信していることを確認します。トラップを受信していない場合は、管理システムの適切なログファイルを調査し、管理ソフトウェアが提供する標準のトラブルシューティング手順を実行してください。LifeKeeper のログを調べると、トラップメッセージの送信に問題があるかどうかを判断することができます。詳細については、[SNMP のトラブルシューティング](#) を参照してください。

SNMP イベント転送の無効化

LifeKeeper による SNMP トラップの生成を無効にするには、ファイル `/etc/default/LifeKeeper` の `LK_TRAP_MGR` 環境変数から IP アドレスの割り当てを削除します。削除するには、コマンドラインで「disable」オプションを指定して `lk_configsnmp` ユーティリティを実行します (例については `lk_configsnmp (1M)` マニュアルページを参照)。または、`/etc/default/LifeKeeper` を編集して、`LK_TRAP_MGR` のエントリを `LK_TRAP_MGR=` に変更します (または行全体を削除する)。この手順は、トラップメッセージの送信を無効にしたい各ノードで実行する必要があります。

SNMP のトラブルシューティング

SNMP によるイベント転送に関連して予想される問題とその解決策を以下に説明します。具体的なエラーメッセージについては、[LifeKeeper メッセージカタログ](#) を参照してください。

問題: LifeKeeper から SNMP のトラップメッセージが送信されない。

解決策: snmptrap ユーティリティがインストールされていることを確認します (通常は /bin/bin にあります)。インストールされていない場合は、適切な SNMP パッケージをインストールします ([前提条件](#)を参照)。別の場所にインストールされている場合は、ファイル /etc/default/LifeKeeper の PATH 変数に適切なパスを追加します。

問題: SNMP のエラーメッセージがログに記録されない。LifeKeeper サーバから SNMP のトラップメッセージが送信されていないように見える。

解決策: トラップを受信するネットワーク管理サーバの IP アドレスが LK_TRAP_MGR に設定されていることを確認します。コマンドラインで、lk_configsnmp を「-query」オプション付きで使用して設定を確認します (lk_configsnmp (1M) マニュアルページの例を参照してください)。または、ファイル /etc/default/LifeKeeper の LK_TRAP_MGR のエントリを確認します。この変数は、SNMP トラップメッセージを生成する LifeKeeper の各ノードで設定する必要があります。

LifeKeeper イベントメール通知

LifeKeeper イベントメール通知の概要

LifeKeeper イベントメール通知は、特定のイベントが LifeKeeper クラスタで発生したときに 1 人以上のユーザがメールによる通知を受信する仕組みです。LifeKeeper のイベント通知機能では、特定のイベントが起きたときに通知を受信するアプリケーションを登録することができます (sendevent (5) マニュアルページを参照)。LifeKeeper は、LifeKeeper の動作を監視したいユーザのグループに向けて LifeKeeper の重要なイベントに関するメール通知を送信するように簡単に設定できます。さらに、/var/log/lifekeeper.log ファイルまたは LifeKeeper GUI の [サーバログファイルの表示](#) 機能を使用すると、送信された各メール通知のログを参照することができます。メッセージは通常 NOTIFY ログに入ります。ログの内容をコマンドラインで表示する方法の詳細については、lk_log (8) マニュアルページを参照してください。

デフォルトでは、LifeKeeper イベントメール通知は無効になっています。この機能を有効にするには、/etc/default/LifeKeeper で指定する LK_NOTIFY_ALIAS 環境変数を設定する必要があります。LK_NOTIFY_ALIAS 環境変数には、メールアドレスまたはエイリアスを 1 つまたは複数個 (カンマ区切り) 設定できます。LK_NOTIFY_ALIAS を設定するには、コマンドラインから lk_confignotify alias (lk_confignotifyalias (1M) マニュアルページで例を参照してください) を実行してイベントが発生したときにメールを受信するアドレスまたはアドレスリストを指定するか、デフォルトファイル /etc/default/LifeKeeper を編集してメールアドレスまたはアドレスリストを追加します。LK_NOTIFY_ALIAS= エントリを見つけて、アドレスまたはカンマ区切りのアドレスリストを入力します。選択した LifeKeeper イベントについてメールを送信する必要があるクラスタのすべてのノードで以上の手順を繰り返します。

メール通知を無効にするには、引数 -disable を付けて lk_confignotifyalias (lk_confignotifyalias (1M) マニュアルページで例を参照してください) を実行するか、デフォルトファイル /etc/default/LifeKeeper を編集して LK_NOTIFY_ALIAS の設定を削除します (この行を LK_NOTIFY_ALIAS= に変更)。

メールが生成される LifeKeeper のイベント

以下の LifeKeeper イベントが発生するとメール通知が生成されます (LK_NOTIFY_ALIAS が設定されている場合)。

メールが生成される LifeKeeper のイベント

LifeKeeper のイベント	イベントの説明
LifeKeeper Startup Complete	LifeKeeper が起動したノードから送信されます。
LifeKeeper Shutdown Initiated	LifeKeeper のシャットダウンを開始したノードから送信されます。
LifeKeeper Shutdown Complete	LifeKeeper のシャットダウンを完了したノードから送信されます。
LifeKeeper Manual Switchover Initiated on Server	手動スイッチオーバーを要求されたノードから送信されます。
LifeKeeper Manual Switchover Complete - recovered list	手動スイッチオーバーが完了したノードから、リカバリに成功したリソースのリストと共に送信されます。
LifeKeeper Manual Switchover Complete - failed list	手動スイッチオーバーが完了したノードから、切り替えに失敗したリソースのリストと共に送信されます。
LifeKeeper Node Failure Detected	クラスタ内のノードに障害が発生したときに、クラスタ内の各ノードから送信されます。
LifeKeeper Node Recovery Complete for Server - recovered list	障害ノードからのリソースをリカバリしたクラスタ内の各ノードから、リカバリに成功したリソースのリストと共に送信されます。
LifeKeeper Node Recovery Complete for Server - failed list	障害ノードからのリソースのリカバリに失敗したクラスタ内の各ノードから、リカバリに失敗したリソースのリストと共に送信されます。
LifeKeeper Resource Recovery Initiated	リソースをリカバリしているノードから送信されます。このメールに続いて、リカバリが完了したか失敗したかを示すメッセージ(「Resource Recovery Complete」または「Resource Recovery Failed」)が必ず送信されます。
LifeKeeper Resource Recovery Complete	リソースのリカバリが成功した時点で、「LifeKeeper Resource Recovery Initiated」メッセージを送信したノードから、リカバリに成功したリソースのリストと共に送信されます。
LifeKeeper Resource Recovery Failed	リソースが In Service の状態になることができない場合に、「LifeKeeper Resource Recovery Initiated」メッセージを送信したノードから、リカバリに成功したリソースのリストと共に送信されます。
LifeKeeper Communications Path Up	ノードへのコミュニケーションパスが確立されました。
LifeKeeper Communications Path Down	ノードへのコミュニケーションパスがダウンしました。

LifeKeeper イベントメール通知の設定

前提条件

イベントメール通知機能は、LifeKeeper のコア機能の一部として含まれており、LifeKeeper の追加パッケージをインストールする必要はありません。ただし、LifeKeeper イベントのメール通知を生成する LifeKeeper の各ノードに電子メールソフトウェアがインストールされている必要があります。LifeKeeper は、mailx パッケージによってインストールされるメールユーティリティを使用して通知を送信します。

メールの設定は、LifeKeeper の本機能が提供する範囲ではありません。デフォルトでは、LifeKeeper イベントメール通知は無効になっています。

設定作業

LifeKeeper イベントメール通知を設定するには、以下の作業を実施します。

1. 上述のメールユーティリティが利用できることを確認します。
2. LifeKeeper のイベントのメール通知を受信するユーザ(1人以上)を特定し、LifeKeeper のデフォルトファイル `/etc/default/LifeKeeper` の `LK_NOTIFY_ALIAS` を設定します。これを行うには、コマンドラインを使用するか、ファイル `/etc/default/LifeKeeper` を編集して通知を受信するメールアドレスまたはエイリアスを指定します。
 - コマンドラインからは、`lk_confignotif্যালias` を使用します (詳細については、`lk_confignotif্যালias (1M)` のマニュアルページを参照してください)。このユーティリティでは、カンマ区切りのメールアドレスまたはエイリアスのみ使用できます。
 - または、デフォルトファイル `/etc/default/LifeKeeper` を編集してメールアドレスまたはエイリアスを追加します。`LK_NOTIFY_ALIAS=` エントリを見つけて「=」の右側にメールアドレスまたはエイリアス(1つまたはカンマ区切りのリスト)を入力します(「=」の前後にはスペースを入れません)。

設定の確認

設定が正常に動作することを確認するには、LifeKeeper の処理を実行します (例えば、LifeKeeper を [開始](#) または [停止](#) する、または LifeKeeper GUI を使用して、あるリソースを手動で In Service の状態にするなど)。ファイル `/etc/default/LifeKeeper` の `LK_NOTIFY_ALIAS` で指定したユーザがメールを受信していること、LifeKeeper のログファイルにメッセージが記録されていることを確認します。メールを受信していない場合は、メール障害に対する通常のトラブルシューティング手順を実行してください。LifeKeeper のログを調べると、メール送信に問題があるかどうかを判断することができます。詳細については、[メール通知のトラブルシューティング](#) を参照してください。

イベントメール通知の無効化

LifeKeeper によるメール通知の生成を無効にするには、ファイル `/etc/default/LifeKeeper` の `LK_NOTIFY_ALIAS` 環境変数からメールアドレスとエイリアスの割り当てを削除するだけです。コマンドラインで `lk_confignotif্যালias` ユーティリティを「`--disable`」オプションを付けて実行します (`lk_confignotif্যালias`

(1M) マニュアルページの例を参照してください)。または、`/etc/default/LifeKeeper` を編集して、`LK_NOTIFY_ALIAS` のエントリを `LK_NOTIFY_ALIAS =` に変更します。この手順は、メール送信を無効にしたい各ノードで実行する必要があります。

メール通知のトラブルシューティング

LifeKeeper イベントのメール通知に関連して予想される問題とその解決策を以下に説明します。具体的なエラーメッセージについては、[LifeKeeper メッセージカタログ](#) を参照してください。

問題: LifeKeeper からのメールを受信しない。

解決策: メールユーティリティがインストールされていることを確認します (通常は `/bin/mail` にあります)。インストールされていない場合は、`mailx` パッケージをインストールします。別の場所にインストールされている場合は、ファイル `/etc/default/LifeKeeper` の `PATH` 変数にメールユーティリティのパスを追加します。

問題: LifeKeeper からのメールを受信しない。

解決策: メール設定を確認し、配信用のキューにメールメッセージが滞留していないことを確認します。メール設定の問題が原因でメッセージが滞留することがあります。`LK_NOTIFY_ALIAS` で指定しているメールアドレスが有効なアドレスであり、カンマで区切られていることを確認します。

問題: ログファイルに「mail returned」というエラーメッセージがある。

解決策: メールコマンドがエラー「X」を返す場合、LifeKeeper イベントがメールを生成、送信する際に問題 (「node failure」など) が発生しています。メール設定を確認し、`LK_NOTIFY_ALIAS` に含まれるメールアドレスが有効であり、アドレスのリストがカンマで区切られていることを確認します。また、`LK_NOTIFY_ALIAS` で指定しているメールアドレスのフォーマットを使用してコマンドラインからそれらのアドレスにメールを送信できることを確認します。

問題: メッセージや成功または失敗がログに何も記録されず、ノードのフェイルなどの LifeKeeper イベントが発生したときもメールを受信するはずのユーザがメールを受信しない。

解決策: `LK_NOTIFY_ALIAS` にメールアドレスが設定されており、複数の場合はカンマで区切られていることを確認します。コマンドラインで、`lk_confignotifyalias` を「`-query`」オプション付きで使用して設定を確認します (`lk_confignotifyalias` (1M) マニュアルページの例を参照してください)。または、ファイル `/etc/default/LifeKeeper` の `LK_NOTIFY_ALIAS` で確認します。この変数は、メール通知メッセージを生成する LifeKeeper の各ノードで設定する必要があります。また、[LifeKeeper イベントメール通知の概要](#) で、その LifeKeeper イベントがメールメッセージを生成するかどうかを確認します (すべてのイベントがメールメッセージを生成するわけではありません)。

任意の設定作業

[Confirm Failover] と [Block Resource Failover] の設定

LifeKeeper の通常の動作として、ノード障害やリソース障害によるバックアップノードへの切り替えは自動的に行

われるようになっています。しかし、利用する環境によっては、障害が検出されたシステムのフェイルオーバー/リカバリをLifeKeeperが実行する前にシステム管理者の手動による確認を必須とすることが望ましいこともあります。そのような場合には、この **[Confirm Failover]** や **[Block Resource Failover]** 設定の利用することができます。この機能を利用することによって、リソース障害やノード障害が発生した際に、自動的に行われるフェイルオーバーをブロックしたり、フェイルオーバーが行われるまでの待ち時間を設けたりすることができます。

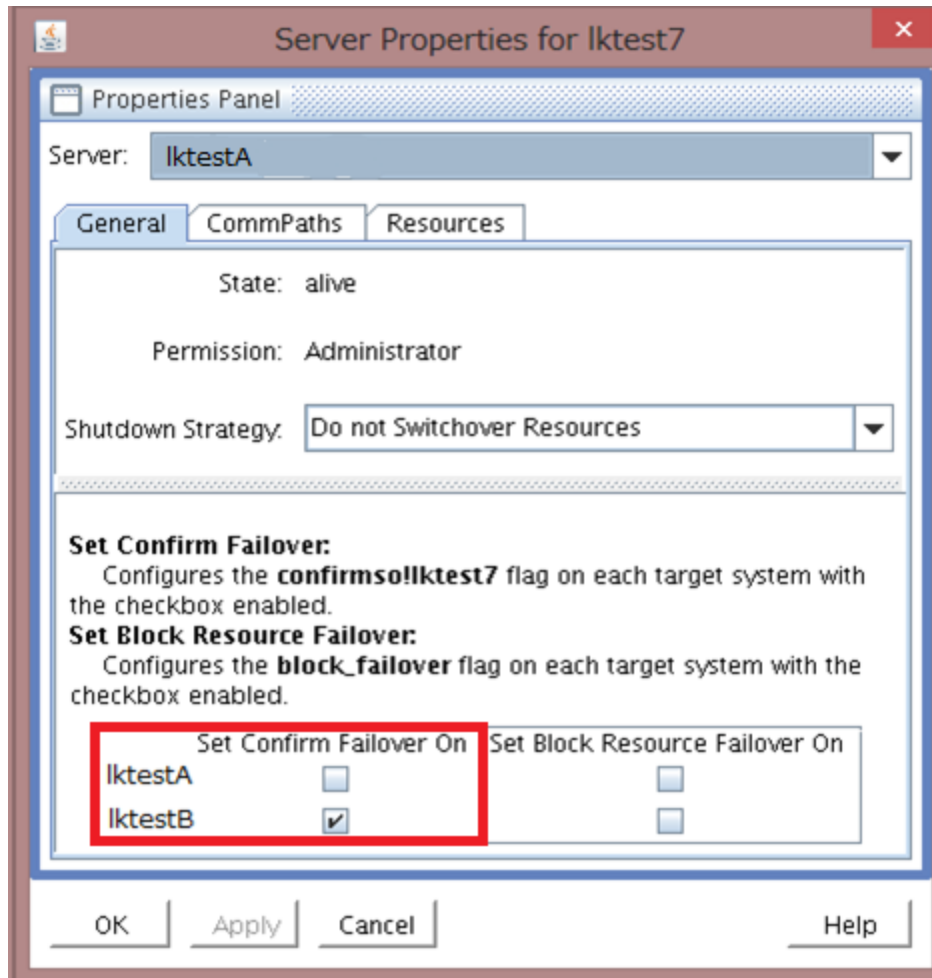
以下の説明、例、および考慮事項をよく読んで理解してから、お使いのSPS環境で **[Confirm Failover]** または **[Block Resource Failover]** を設定してください。これらの設定は、コマンドライン、または **LifeKeeper** の **GUI** の **[Properties]** パネルから使用できます。

[Set Confirm Failover On]

[Confirm Failover]を設定すると、LifeKeeper クラスターのノード障害によるフェイルオーバーが発生した時(注記: ノード障害は、システムへのすべてのLifeKeeperコミュニケーションパス障害によって識別されます。)、バックアップノードへの切り替えの実行に対して待ち時間を設けることができますようになります(後述のCONFIRMSODEF変数を参照して下さい)。また、その待ち時間の中にバックアップノードへの切り替えを実行するかしないかをユーザーが決定できるようになります。

注記: Set Confirm Failover On設定は、ノード障害が発生した場合にのみ有効です。一つ以上のコミュニケーションパスがアクティブであるリソース障害に対しては有効ではありません。

[Confirm Failover]設定をGUIで有効にする場合は、サーバプロパティのGeneralタブ画面を使用します。以下はサーバプロパティのGeneral画面の例です。赤で囲まれた部分が**[Confirm Failover]**を設定する画面です。



注意 : この設定は、SPS の管理者権限を持つユーザのみが使用できます。

この画面の例では、lktestAという名前のホストから見た設定となります。上図の赤で囲まれた部分が【**Confirm Failover**】の設定部分で、HAクラスタを構成するノードの名前が縦に表示されています。この例では、lktestAの対向ノードがlktestBとなっています。

図の例ではlktestAを選択して設定状況を表示しており、lktestBの横にチェックが入った状態となっています。この場合、lktestB上にConfirm Failover フラグが作成され、lktestAからlktestBへのフェイルオーバーが行われる際に、フェイルオーバー実施までのタイムアウトと実施可否の確認が有効になります。このフラグの有無によって、【**Confirm Failover**】の動作の実施がコントロールされます。

この時のフラグの作成状況については、コマンドで確認することができます。例にあるlktestAというホスト上でlktestBの欄にチェックを入れた場合には、lktestBに【**Confirm Failover**】フラグが作成されます。(例の通りの操作の場合 lktestAにはフラグは作成されません)。具体的には以下のような出力になります。

```
[root@lktestB~]# flg_list
```

```
confirmso!lktestA
```

「confirmso!lctestA」はflg_listコマンドの結果が出力され、**[Confirm Failover]**フラグがlctestBノード上に設定されます。confirmsoフラグがセットされている状態で、フェイルオーバーが発生するとLifeKeeperのログファイルには以下のようなログが記録されます。

```
INFO:lcd.recover:::004113:chk_man_interv: Flag confirmso!hostname is set, issuing confirmso event and waiting for switchover instruction.
```

```
NOTIFY:event.confirmso:::010464:LifeKeeper: FAILOVER RECOVERY OF MACHINE lctestA requires manual confirmation! Execute '/opt/LifeKeeper/bin/lk_confirmso -y -s hostname' to allow this failover, or execute '/opt/LifeKeeper/bin/lk_confirmso -n -s lctestA' to prevent it. If no instruction is provided, LifeKeeper will timeout in 600 seconds and the failover will be allowed to proceed.
```

このメッセージが出力されている間に、次のいずれかのコマンドを実行するとフェイルオーバーの可否をコントロールすることができます。

フェイルオーバーを続行する場合

```
#/opt/LifeKeeper/bin/lk_confirmso -y -s ホスト名
```

フェイルオーバーをブロックする場合

```
#/opt/LifeKeeper/bin/lk_confirmso -n -s ホスト名
```

コマンドを実行する際に指定するホスト名はConfirm Failoverフラグに書かれているホスト名、この例では、lctestAとなります。ログに具体的なコマンドの実行例が書かれていますので、内容を元にコマンドを実行してください。

設定された待ち時間を超えた場合、LifeKeeperのデフォルトの設定ではバックアップノードへのフェイルオーバー(あるいは、フェイルオーバーのブロック)を自動的に行うようになっています。タイムアウトを迎えた場合以下のようなログが記録されます。

```
lcdrecover[xxxx]: INFO:lcd.recover:::004408:chk_man_interv: Timed out waiting for instruction, using default CONFIRMSODEF value 0.
```

待ち時間を経過した際の動作は、/etc/default/LifeKeeperファイルの「CONFIRMSODEF=1 or 0」の設定で制御されます。デフォルトでは「0」が設定されており、待ち時間を超えた場合にはフェイルオーバーが継続されます。この値を「1」に変更した場合には、待ち時間を経過した場合にフェイルオーバーしないようにすることができます。

フェイルオーバー時の待ち時間を変更したい場合には、/etc/default/LifeKeeperファイルの「CONFIRMSOTO=秒」の値を変更してください。上記CONFIRMSODEFの値によって決定されたフェイルオーバーの実行、あるいはブロックをする前に、この変数はユーザからの手動確認を待つための秒数を指定します。この設定をこれらの変数の影響を反映するために、LifeKeeperやOSを再起動する必要はありません。CONFIRMSOTOに0秒を設定すると待ち時間無しで、CONFIRMSODEFの設定に基づいた動作をさせることができます。

[Confirm Failover]設定を選択するタイミング

この設定は、コミュニケーションパスが冗長化されていない環境で場合のディザスタリカバリやWAN構成で使用されます。

- 通常のサイト(非マルチサイトクラスタおよび非XenServer)では、あるサーバで**[Properties]**ページを開き、**[Confirm Failover]**フラグをオンに設定するサーバを選択してください。
- マルチサイト WAN の構成の場合:フェイルオーバーの手動確認を**[Confirm Failover flag]**の設定で、有効にしてください。

- マルチサイト LAN の構成の場合:フェイルオーバーの手動確認を有効にするために、**[Confirm Failover flag]**を設定しないでください。
- マルチサイトクラスタ環境では、非ディザスタシステムからDRシステムを選択し、**[Set Confirm Failover flag]** チェックボックスをオンにします。クラスタ内の非ディザスタサーバのそれぞれについて、指定したシステムでのリソース障害に起因するフェイルオーバーをブロックします。**[Properties]** パネルを開いてこの設定を選択する必要があります。

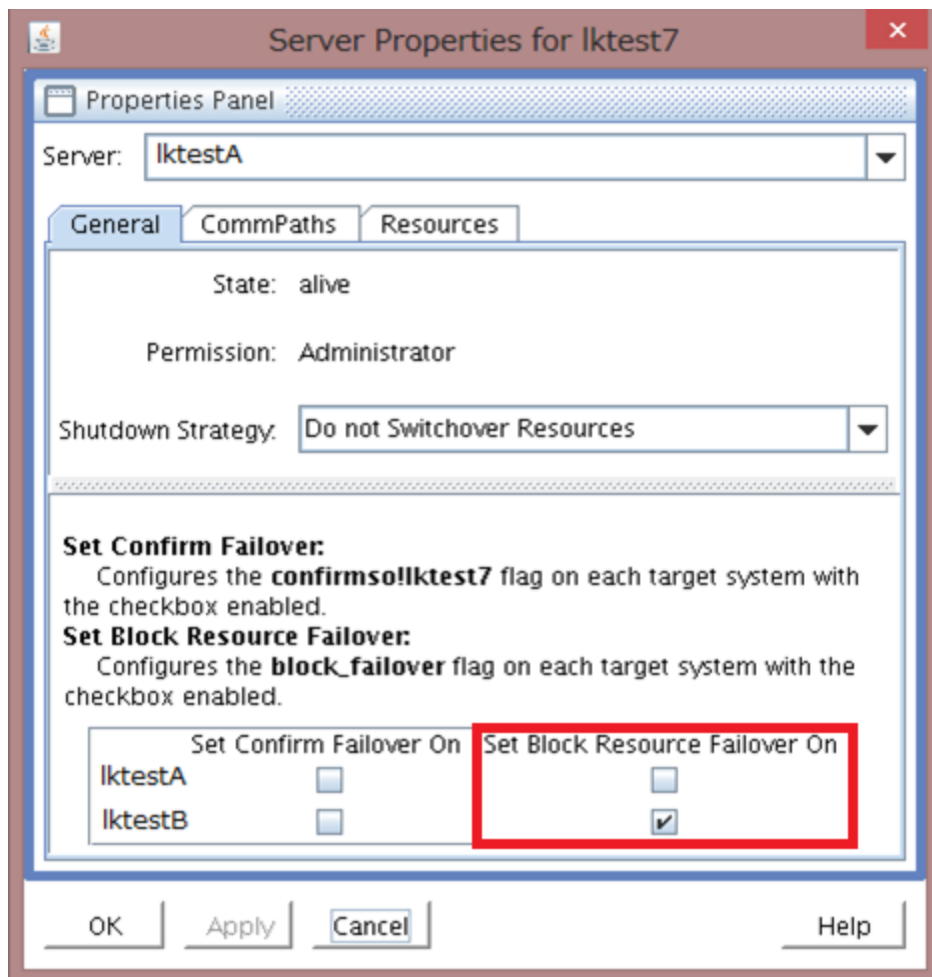
[Block Resource Failover On]

[Block Resource Failover] 設定は、指定したシステムでのリソース障害に起因するフェイルオーバーをブロックします。

注記: Block Resource Failover設定は、ノード障害が発生した場合のフェイルオーバー動作には影響しません。この設定は、ローカルリソースの回復に失敗し、クラスタで他ノードへリソースを転送する場合のみフェイルオーバーをブロックします。

デフォルトでは、リソース障害を検知した時、ローカルシステムでの障害リソースのリカバリ(ローカルリカバリ)を試行し、ローカルリカバリが失敗した場合、または有効になっていない場合は、リソースが定義されている、優先順位が次に最も高いスタンバイノードにフェイルオーバーしようとします。Block Resource Failoverの設定は、この時のフェイルオーバーをブロックします。

Block Resource Failoverの設定をGUIで有効にする場合は、サーバプロパティのGeneralタブ画面を使用します。以下はサーバプロパティのGeneral画面の例です。赤で囲まれた部分が**[Block Resource Failover On]**を設定する画面です。



注意 : この設定は、SPS の管理者権限を持つユーザのみが使用できます。

この画面の例は、lktestAという名前のホストから見た設定となります。上図の赤で囲まれた部分が**[Confirm Failover]**の設定部分で、HAクラスタを構成するノードの名前が縦に表示されています。この例では、lktestAの対向ノードがlktestBとなっています。

この場合、lktestB上にBlock Resource Failover フラグが作成されます。lktestBでflg_listコマンドを実行すると、「block_failover」というフラグが作成されていることを確認することができます。出力例は以下の通りです。

```
[root@lktestB~]# flg_list
```

```
block_failover
```

この結果lktestB上で、リソース障害が発生した場合には、他ノード (**lktestA**) へのフェイルオーバーがブロックされます。

block_failoverフラグは、それがセットされているノード上で発生したリソース障害によるフェイルオーバーをブロックします。この設定によってフェイルオーバーがブロックされた場合には、以下のようなログが記録されます。

ERROR:lcd.recover:::004787:Failover is blocked by current settings. MANUAL INTERVENTION IS REQUIRED

設定の利用条件 / 考慮事項

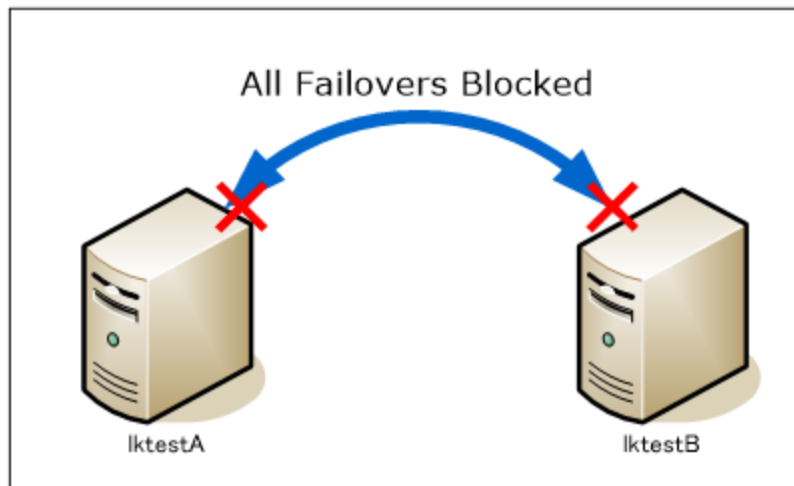
マルチサイト設定では、設定に含まれるすべてのサーバについて、フェイルオーバーのブロックを有効にしないでください。

マルチサイトクラスタ構成での重要な考慮事項:マルチサイトクラスタ構成のサーバについては、**[Set Block Resource Failover On]** 列のチェックボックスをオンにしないでください。confirmFailOn4.png

設定例

いくつかの具体的な設定例を解説します。

全ての自動フェイルオーバーをすべてブロックする



この例ではlktestAとlktestBどちらでノード障害、リソース障害が発生してもフェイルオーバーしないようにします。この設定にはConfirm Failover設定とBlock Resource failover設定を利用します。設定例は次の通りです。

1. lktestAサーバを選択し、**[Server Properties]** を表示してください。

[General] タブで、lktestBの行にある **[Set Confirm Failover On]** ボックスと、lktestAとlktestB両方の行にある**[Set Block Resource Failover on]**ボックスをオンにしてください。

全ての自動フェイルオーバーをすべてブロックする

GUIのlktestAのサーバプロパティ設定例

	Set Confirm Failover On	Set Block Resource Failover On
lktestA	(チェックなし)	✓
lktestB	✓	✓

※ GUIでサーバプロパティをみる場合、ノード名は、プロパティパネルの上部近くに表示されます。

2. lktestBサーバを選択し、**[Server Properties]**を表示してください。

[General] タブで、lktestAの行にある **[Set Confirm Failover On]** ボックスをオンにしてください。

GUIのlktestBのサーバプロパティ設定例

	Set Confirm Failover On	Set Block Resource Failover On
lktestB	(チェックなし)	✓
lktestA	✓	✓

※ GUIでは表示している自ノードのホスト名が列の上部に表示されます

GUIで設定した後、lktestBでflg_listコマンドを使用してフラグ設定を確認すると、「confirmso!lktestA」フラグが作成されていることが確認できます。

ここまでの結果それぞれのノードにconfirmso!hostnameとblock_failoverフラグがセットされていることを確認してください。confirmsoフラグについては、lktestA上のlktestBからのフェイルオーバーをブロックするために、フェイルオーバー確認が実行されることになっているhostnameが、フラグ名の内容の一部として記載されることを確認して下さい。lktestAは、lktestA上でlktestBをconfirmsoフラグ名の内容に記載する必要があります。本設定例ではそれぞれのノードで以下のようにフラグが作成されます。

	Confirm Failover フラグ	Block Resource Failoverフラグ
lktestA側	confirmso!lktestB	block_failover
lktestB側	confirmso!lktestA	block_failover

3. 両ノードの/etc/default/LifeKeeperのCONFIRMSOTOとCONFIRMSODEFの設定値を以下のように設定してください。(LifeKeeperやOSの再起動は必要ありません。)

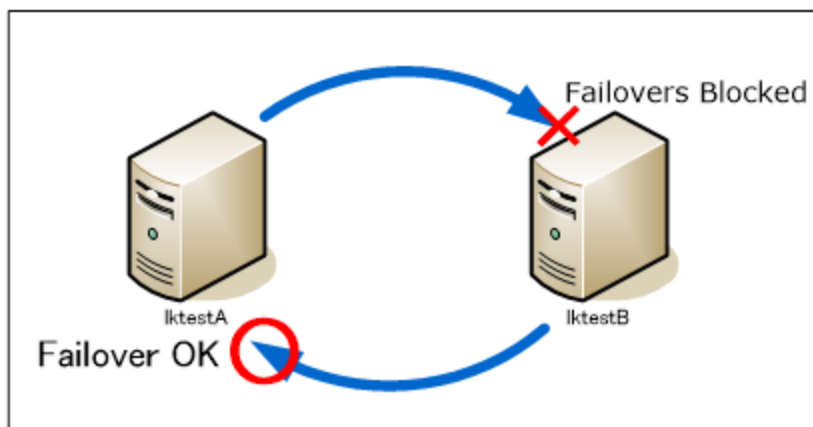
CONFIRMSODEF = 1

CONFIRMSOTO = 0

CONFIRMSOTOの設定については待ち時間を設ける場合には時間を秒で指定することも可能です。ポイントはCONFIRMSODEFの設定が0(フェイルオーバーする)から1(フェイルオーバーしない)になっていることです。

上記の設定により、オペレータが介在することなく、ノード障害はすぐにブロックされます。

一方向のフェイルオーバーをブロックする



この例ではlktestAで障害が検知された場合にはノード障害、リソース障害どちらの場合でもlktestBへのフェイルオーバーをブロックします。逆に、lktestBで障害が発生した場合にはノード障害、リソース障害どちらの場合でもServerAへフェイルオーバーします。

1. lktestAを選択し、**[Server Properties]**を表示してください。
2. **[General]** タブで、lktestAの行にある **[Set Confirm Failover On]** ボックスと、lktestAの行にある**[Set Block Resource Failover On]**ボックスをオンにしてください。GUIでの設定状態は以下のようになります。

GUIのlktestAのサーバプロパティ設定例

	Set Confirm Failover On	Set Block Resource Failover On
lktestA	(チェックなし)	✓
lktestB	✓	(チェックなし)

3. lktestBサーバを選択し、**[Server Properties]**を表示してください。

[General] タブで、lktestAの行にある **[Set Confirm Failover On]** ボックスをオフ(チェックなし)にしてください。GUIでの設定状態は以下のようになります。

GUIのlktestBのサーバプロパティ設定例

	Set Confirm Failover On	Set Block Resource Failover On
lktestA	(チェックなし)	(チェックなし)
lktestB	(チェックなし)	✓

※ GUIでは表示している自ノードのホスト名が列の上部に表示されます。

この設定で、lktestB上に「confirmso!lktestA」フラグ、lktestA上に「Block failover」フラグが設定されていることを確認してください (lktestAには、confirmsoフラグは設定されません)。

	Confirm Failover フラグ	Block Resource Failoverフラグ
lktestA	なし	block_failover
lktestB	confirmso!lktestA	なし

- lktestBの/etc/default/LifeKeeperを以下のように設定してください。LifeKeeperやOSの再起動は必要ありません。

CONFIRMSODEF = 1

CONFIRMSOTO = 0

ここまでの設定によって、lktestAサーバからlktestBのサーバへのフェイルオーバーは許可されますが、lktestBからlktestAへのフェイルオーバーは許可されません。

サーバのシャットダウンストラテジーの設定

シャットダウンストラテジーは、サーバがシャットダウンするときにバックアップサーバにリソースをスイッチオーバーするかどうかを制御するLifeKeeperの設定オプションです。以下のオプションがあります。

Do Not Switch Over Resources (デフォルト)	LifeKeeperは、正常なシャットダウンではバックアップサーバのリソースを起動しません。
Switch Over Resources	LifeKeeperは、正常なシャットダウンでバックアップサーバのリソースを起動します。

シャットダウンストラテジーは、デフォルトでは「Do Not Switch Over Resources」に設定されています。クラスタ内の各サーバでどちらのストラテジーを使用するかを決定し、必要に応じてシャットダウンストラテジーを「Switch Over Resources」に変更してください。

クラスタ内の各サーバで次のようにします。

1. [\[Edit\]メニュー](#)で **[Server]** を選択し、次に **[Properties]** をクリックします。
2. 修正するサーバを選択します。
3. **[Server Properties]** ダイアログの [\[General\] タブ](#) で、**[Shutdown Strategy]** を選択します。

注記: シャットダウンストラテジーが有効に機能するには、正常なシャットダウン時に LifeKeeper のプロセスが起動している必要があります。

注記: いくつかの Amazon EC2 の設定において、シャットダウンストラテジーを "Do not Switchover Resources" に設定した場合に問題が起こることがあります。詳細は、トラブルシューティング > [既知の問題と制限](#) をご確認ください。

LifeKeeper ハートビートの調整

ハートビート設定項目の概要

LifeKeeper のハートビートは、各サーバが「生存」していることを確認するためにコミュニケーションパスを通じて LifeKeeper のサーバ間で送受信される信号です。ハートビートに関しては、LifeKeeper が障害を検知する速さを決定する要素が2つあります。

- 間隔: ハートビートの間の秒数。
- ハートビート回数: コミュニケーションパスが切断していると LifeKeeper が判定するまでに許容されるハートビートの失敗回数。

これらのハートビートの値は、LifeKeeper デフォルトファイル /etc/default/LifeKeeper 内の以下の2つの設定項目で指定します。デフォルト値を使用した場合よりも早期にサーバの障害を検知したい場合は、設定項目を変更することができます。

- LCMHBEATTIME (間隔)
- LCMNUMHBEATS (ハートビート回数)

次の表は、TCP および TTY 経由のハートビートの設定項目についてのデフォルト値と最小値の一覧です。TTY コミュニケーションパスは、媒体として通信速度が遅いため、間隔を2秒未満にすることはできません。

設定項目	デフォルト値	最小値
LCMHBEATTIME	5	1 (TCP) 2 (TTY)
LCMNUMHBEATS	3	2 (TCP、TTY)

重要な注記: どちらの設定項目も、クラスタ内のすべてのサーバで必ず同じ値にする必要があります。

例

LifeKeeper のクラスタで両方の間隔がデフォルト値に設定されていると仮定します。LifeKeeper は、サーバ間で 5 秒ごとにハートビートを送信します。通信障害によって 2 回のハートビートが途絶し、3 回目のハートビートで再開した場合、LifeKeeper はアクションを実行しません。コミュニケーションパスの切断継続時間がハートビート 3 回分になった場合は、LifeKeeper はそのコミュニケーションパスを切断と判定します。ただし、他方の冗長的なコミュニケーションパスも切断と判定されるまではフェイルオーバを開始しません。

ハートビートの設定

設定項目とその値を追加するには、`/etc/default/LifeKeeper` ファイルを手動で編集する必要があります。通常、デフォルトファイルにはこれらの設定項目のエントリが含まれていません。設定する値を含む次の行を最終行以降に追加してください。

```
LCMHBEATTIME=x
LCMNUMHBEATS=y
```

最小値を下回る値を設定した場合、LifeKeeper はその値を無視して代わりに最小値を採用します。

設定上の考慮事項

- 間隔を 5 秒未満に設定すると、ネットワークの中断による誤ったフェイルオーバを発生させるリスクが高くなるため、5 秒未満に設定する場合は、コミュニケーションパスをプライベートネットワーク上で構成してください。
- 検証によると、ハートビート回数を 2 未満にした場合に誤ったフェイルオーバの発生リスクが高まります。このため、この値は 2 以上に制限されています。
- 誤ったフェイルオーバを回避するため、間隔およびハートビート回数の値はどちらもクラスタ内のすべてのサーバで必ず同じ値にする必要があります。このため、これらの値を編集する前に両方のサーバで LifeKeeper を停止しておく必要があります。LifeKeeper の稼働開始後、アプリケーションを保護している状態でハートビートの設定項目を編集する場合は、コマンド `/etc/init.d/lifekeeper stop-daemons` を使用できます。このコマンドは LifeKeeper を停止しますが、保護下のアプリケーションは停止しません。
- LCMHBEATTIME および LCMNUMHBEATS の値に上限値はありません。ただし、非常に大きい数字に値を設定すると、LifeKeeper の障害検知能力は著しく損なわれます。例えば、両方の値を 25 に設定した場合、サーバ障害を検知するまで LifeKeeper は 625 秒間 (10 分間以上) 待機します。これはサーバをリブートしてクラスタに再参加させるのに十分な時間です。

注記: TTY および TCP コミュニケーションパスの両方を使用する場合、各設定項目の値は両方のコミュニケーションパスに適用されます。唯一の例外は、TTY コミュニケーションパスの最小値である 2 未満の値が間隔に設定された場合です。

例えば、障害をできるだけ早く検知するために、LifeKeeper で許容される最小値を指定したとします。

```
LCMHBEATTIME=1
LCMNUMHBEATS=2
```

このとき LifeKeeper は TCP コミュニケーションパスの間隔に 1 秒を採用し、TTY の間隔には 2 秒を採用します。サーバ障害が発生すると、LifeKeeper は間隔の短い TCP の障害 (1 秒間隔の 2 回のハートビート後) を先に検知します。ただし、TTY の障害 (2 秒間隔の 2 回のハートビート後) を検知するまでは何もしません。

SPS API でカスタム証明書を使用する

SIOS Protection Suite (SPS) API のリリース 7.5 以降では、異なるシステムとの通信に SSL/TLS が使用されます。現在、この API は一部のみ使用され、内部使用のみとして予約されていますが、将来のリリースではお客様とサードパーティが使用できるように公開される可能性があります。デフォルトでは、ノード間で一定の身元確認が可能なデフォルト証明書が SPS と共にインストールされます。このドキュメントでは、デフォルト証明書を組織独自の認証局 (CA) が作成した証明書に置き換える方法を説明します。

注記: 通常の SPS 通信では、これらの証明書は使用されません。

証明書の使用方法

データ転送を保護するために SPS サーバ間の通信に SSL/TLS が使用されている場合、システムがそれ自体を識別するために証明書を提供します。また、システムは、CA 証明書を使用して、SSL 接続経由で提示された証明書を検証します。

以下の 3 種類の証明書が使用されます。

- /opt/LifeKeeper/etc/certs/LK4LinuxValidNode.pem (server certificate)
- /opt/LifeKeeper/etc/certs/LK4LinuxValidClient.pem (client certificate)
- /opt/LifeKeeper/etc/certs/LKCA.pem (certificate authority)

最初の 2 つの証明書は、サーバが実行する検証に合格するために CA 証明書による署名が必要です。証明書の共通名は検証されません。証明書は CA によって署名されるのみということに注意してください。

独自の証明書の使用

運用環境によっては、デフォルト証明書を組織内部の CA または商用 CA が作成した証明書に置き換える必要がある場合があります。そのような場合は、上記の 3 種類の証明書を、同じ証明書ファイル名を持つ新しい証明書に置き換えます。これらの証明書は PEM 形式です。LK4LinuxValidNode.pem および LK4LinuxValidClient.pem はそれぞれ、キーと証明書の両方を含んでいます。LK4LinuxValidNode.pem 証明書は、サーバタイプの証明書です。LK4LinuxValidClient.pem は、クライアントタイプの証明書です。

デフォルトの証明書を置換した場合、変更を反映するために SPS を再起動する必要があります。証明書の設定を間違えると、steeleye-lighttpd デーモンが起動に失敗し、LifeKeeper のログファイルにエラーが記録されます。問題が発生した場合、このログファイルを参照すると実行すべき完全なコマンドを見ることができます。

Linux の設定

オペレーティングシステム	必要なすべてのパッケージをインストールするためには、オペレーティングシステムはデフォルトでインストールしてください。最小構成のオペレーティングシステムでは必要なすべてのパッケージが含まれないため、LifeKeeper で使用することはできません。
--------------	---

LifeKeeper クラスターの可用性を最大限に引き出すには、システムで使用するカーネルのバージョンが非常に重要です。次の表に、LifeKeeper 認定テストに合格した、サポート対象のディストリビューション、バージョン、およびカーネルを示します。

注記:SPS 8.1 以降、Red Hat Enterprise Linux システムとサポートされた Red Hat Enterprise Linux 互換 ディストリビューション (CentOS と OEL) で、カーネルのアップグレードを実行する際、インストールイメージから setup スクリプト (./setup) を再実行する必要はなくなりました。カーネルが適切な Red Hat package (rpm ファイル) からインストールされている限り、モジュールはアップグレードしたカーネルで特別な操作を必要とせず、自動的に使用可能になります。SUSE Linux Enterprise Server に対する SPS カーネルモジュールの要件はありません。

ディストリビューションバージョン	サポート対象のバージョン	サポート対象のカーネル
Red Hat Enterprise Linux および Red Hat Enterprise Linux Advanced Platform AMD64/EM64T	5	2.6.18-8.el5
	5.1	2.6.18-8.1.1.el5 (デフォルトカーネル)
	5.2	2.6.18-53.el5
	5.3	2.6.18-92.el5
	5.4	2.6.18-128.el5
	5.5	2.6.18-164.el5
	5.6	2.6.18-194.el5
	5.7	2.6.18-238.el5
	5.8	2.6.18-274.el5
	5.9	2.6.18-308.el5
	5.10	2.6.18-348.el5
Red Hat Enterprise Linux for AMD64/EM64T (*6.0 は非推奨)	6.0*	2.6.32-71.el6
	6.1	2.6.32-131.17.1.el6
	6.2	2.6.32-220.el6
	6.3	2.6.32-279.el6
	6.4	2.6.32-358.el6
	6.5	2.6.32-431.el6
	6.6	2.6.32-504.el6
6.7	2.6.32-573.el6	
Red Hat Enterprise Linux for AMD64/EM64T	7	3.10.0-123.el7
	7.1	3.10.0-229.el7
SUSE SLES 11 for x86_64	SP1	2.6.27.19-5
	SP2	2.6.32.12-0.7
	SP3	3.0.42-0.7.3
	SP4	3.0.76-0.11.1 3.0.101-0.63.1
カーネルのアップデート	5	2.6.18-8.el5
	5.1	2.6.18-53.0.0.0.1.el5
	5.2	2.6.18-92.0.0.0.1.el5
	5.3	2.6.18-128.0.0.0.1.el5
	5.4	2.6.18-164.0.0.0.1.el5
	5.5	2.6.18-194.0.0.0.1.el5
	5.6	2.6.18-238.0.0.0.1.el5
	5.7	2.6.18-274.0.0.0.1.el5
5.8	2.6.18-308.0.0.0.1.el5	

デバイスの動的な追加	<p>LifeKeeper が起動する前に、Linux 側ですべてのデバイスの設定を完了しておく必要があります。LifeKeeper の起動後に LifeKeeper の保護対象のデバイスを設定する場合、そのデバイスを共有する各サーバで LifeKeeper を停止して再起動する必要があります。これにより、デバイスを検知および検証する機能によって設定が確認され、LifeKeeper がデバイスにアクセスできるようになります。</p>
LUN のサポート	<p>Linux の SCSI ドライバには、論理ユニット (LUN) の検索対象とするデバイスを制御するいくつかのパラメータがあります。</p> <ul style="list-style-type: none"> • LUN をサポートしないデバイスのリスト - このリストのデバイスは LUN をサポートしないことが既知であるため、SCSI ドライバはこれらのデバイスに対して LUN を検索することを許可しません。 • LUN をサポートするデバイスのリスト - このリストのデバイスは LUN をサポートすることが既知であるため、必ず LUN を検索します。 • Probe all LUNs on each SCSI device - デバイスがどちらのリストにも存在しない場合、検索するかどうかを指定します。このパラメータは、make config を使用して SCSI モジュールセクションで設定します。 <p>(SUSE を含む) ほとんどのディストリビューションでは、Probe all LUNs 設定はデフォルトで有効になっていますが、Red Hat ではデフォルトで無効に設定されています。LifeKeeper の構成でデータ保護を目的として通常使用される外部 RAID コントローラには、多くの場合、複数の LUN (論理ユニット) が設定されます。LUN のサポートを有効にするには、このフィールドを選択してカーネルを再構築する必要があります。</p> <p>カーネルやモジュールを再構築せずに Probe all LUNs を有効にするには、変数 <code>max_scsi_luns</code> を 255 に設定します (これによって最大 255 個の LUN をスキャンするようになります)。SCSI ドライバがモジュールになっているカーネル (Red Hat など) で <code>max_scsi_luns</code> を設定するには、<code>/etc/modules.conf</code> に以下のエントリを追加し、初期 RAM ディスクを再構築して、再起動してからその RAM ディスクを読み込みます。</p> <pre>options scsi_mod max_scsi_luns=255</pre> <p>SCSI ドライバがコンパイルされるカーネル (SUSE など) で <code>max_scsi_luns</code> を設定するには、<code>/etc/lilo.conf</code> に以下のエントリを追加します。</p> <pre>append="max_scsi_luns=255"</pre> <p>注記: 255 個の LUN をスキャンすると、デバイスによってはブートのパフォーマンスに悪影響を与える可能性があります (特に、BLIST_SPARSELUN が指定されたデバイス)。Dell PV650F というアレイではそのような状況が発生しました。このパフォーマンスの問題を回避するには、アレイ上で設定した LUN の最大数 (16 または 32 など) を <code>max_scsi_luns</code> に設定します。例えば、以下ようになります。</p> <pre>append="max_scsi_luns=16"</pre>

データレプリケーションの設定

項目	説明																													
SIOS DataKeeper の機能 / ディストリビューションマトリクス	<p>SIOS DataKeeper は、Linux カーネルバージョン 2.6 以降をサポートします。一部の DataKeeper 機能には、追加でカーネルの最低要件があります。</p> <p>次の表は、DataKeeper の各機能をサポートする Linux ディストリビューションを「X」で示しています。</p> <table border="1"> <thead> <tr> <th rowspan="2">DataKeeper の機能</th> <th colspan="2">RED HAT</th> <th colspan="2">SUSE</th> </tr> <tr> <th>RHEL 5+</th> <th>RHEL 6</th> <th>SLES 10</th> <th>SLES 11</th> </tr> </thead> <tbody> <tr> <td>複数ターゲットサポート (カーネル 2.6.7+)</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> </tr> <tr> <td>ビットマップインテントログ (カーネル 2.6.16+)</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> </tr> <tr> <td>非同期 (WAN) レプリケーション (カーネル 2.6.16+)</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> </tr> <tr> <td>ビットマップマージ (2.6.27+)</td> <td>X*</td> <td>X</td> <td></td> <td>X</td> </tr> </tbody> </table> <p>*RHEL 5.4 以降が該当します。ビットマップマージのコードは、Red Hat EL5 Update 4 カーネルにバックポートされました。</p>	DataKeeper の機能	RED HAT		SUSE		RHEL 5+	RHEL 6	SLES 10	SLES 11	複数ターゲットサポート (カーネル 2.6.7+)	X	X	X	X	ビットマップインテントログ (カーネル 2.6.16+)	X	X	X	X	非同期 (WAN) レプリケーション (カーネル 2.6.16+)	X	X	X	X	ビットマップマージ (2.6.27+)	X*	X		X
DataKeeper の機能	RED HAT		SUSE																											
	RHEL 5+	RHEL 6	SLES 10	SLES 11																										
複数ターゲットサポート (カーネル 2.6.7+)	X	X	X	X																										
ビットマップインテントログ (カーネル 2.6.16+)	X	X	X	X																										
非同期 (WAN) レプリケーション (カーネル 2.6.16+)	X	X	X	X																										
ビットマップマージ (2.6.27+)	X*	X		X																										
SIOS DataKeeper ドキュメンテーション	<p>SIOS DataKeeper のドキュメンテーションは、SIOS Technology Corp. の Web サイトにある「SIOS Protection Suite テクニカルドキュメンテーション」の中に収録されています。</p>																													

ネットワーク設定

項目	説明
ルーティングテーブルに対する IP Recovery Kit の影響	<p>LifeKeeper が保護する IP アドレスは、論理インターフェースとして、Linux 上で実装されます。Linux 上で論理インターフェースを設定すると、その論理インターフェースに関連付けられたサブネットへのルートが自動的にルーティングテーブルに追加されます。例えば、物理インターフェースによってそのサブネットへのルートがすでに存在する場合も同様です。この追加により、同じサブネットに対して複数のルーティングテーブルエントリが作成される可能性があります。</p> <p>接続元のアドレスを検査して確認するアプリケーションの場合、複数のルーティングテーブルエントリがあると、LifeKeeper システムが (LifeKeeper がインストールされていない) 他のシステム上のそのようなアプリケーションに接続しようとしたときに問題が発生することがあります。複数のルーティングテーブルエントリによって、物理インターフェースからではなく論理インターフェースから接続が張られているように見えます。</p>

アプリケーションの設定

項目	説明
IP サブネットマスク	LifeKeeper 保護下の IP 設定では、物理 インターフェースの IP アドレスと、LifeKeeper が保護するエイリアス IP アドレスのサブネットを同じにする場合、2つのアドレスのサブネット マスクを同じにする必要があります。サブネット マスクの設定を間違えると、LifeKeeper GUI のクライアントとサーバ間の接続に遅延や障害が発生します。
EEpro100 ドライバの初期化	Intel Ethernet インターフェースを搭載するシステムでは、eepro100ドライバの初期化の問題を解決するために、Intel e100ドライバをインストールする必要があります。eepro100ドライバを使用すると、ブート時にインターフェースが起動したときに以下のエラーが発生し、インターフェースをシャットダウンするまでエラーを出し続けることがあります。 eth0: card reports no Rx buffers eth0: card reports no resources

アプリケーションの設定

項目	説明
データベース初期設定ファイル	データベースの初期設定ファイルは、共有デバイス上に置いてローカルファイルシステムの指定場所にシンボリックリンクを作成するか、または個別のシステム上に保持して変更を適用する必要がある場合に手動で両方のシステムを更新するかのいずれかに必要があります。
Oracle のローカルマウントポイント	Oracle のローカル環境は、 <i>internal</i> として接続するか、 <i>sysdba</i> として接続するかによって異なります。データベースを LifeKeeper の保護下に置く場合、「connect / as sysdba」を使用してローカルマウントポイント上にデータベースを作成する必要があります。
Apache のアップデート	Linux オペレーティングシステムのアップグレードの一環として、SPS が保護する Apache アプリケーションをアップグレードするには、起動時にデフォルトサーバインスタンスを無効にする必要があります。 設定ファイル (<i>httpd.conf</i>) がデフォルトのディレクトリ (<i>/etc/httpd/conf</i>) にある場合、Red Hat のアップグレードにより設定ファイルが上書きされます。したがって、アップグレードする前にファイルのコピーを作成し、アップグレードした後にファイルをリストアする必要があります。 また、 <i>Apache Web Server Recovery Kit 管理ガイド</i> の「Apache Web Server の設定に関する考慮事項」セクションも参照してください。

ストレージとアダプタの設定

項目	説明
<p>マルチパス I/O 冗長コントローラ</p>	<p>マルチパス I/O のソリューションには数種類あり、すでに利用可能なものや Linux 環境向けに開発中のものなどがあります。SIOS Technology Corp. は、多くのサーバベンダ、アダプタベンダ、およびドライバ開発者と積極的に協力することで、LifeKeeper とマルチパス I/O ソリューションとの協調動作を実現しています。データの整合性を保護するために LifeKeeper が使用する SCSI リザベーションは、特殊な要件を必要とするため、マルチパス I/O ソリューションの最初の実装では多くの場合要件が満たされません。</p> <p>ディスクアレイサポートに関する以下の技術情報を参照し、個別のアレイがマルチパスおよび特定のマルチパスソリューションでサポートされているかを判断してください。マルチパスおよび特定のマルチパスソリューションと共に動作する LifeKeeper のサポート対象として一覧に指定されていないアレイは、サポート対象ではないと考えてください。</p>

項目	説明
<p>マルチパス構成での大量のI/O</p>	<p>マルチパス構成では、バスの操作中に大量のI/Oを実行すると、システムが応答しなくなったように見えることがあります。マルチパスのソフトウェアがLUNのアクセスをあるパスから別のパスに移動する場合、処理中のI/Oも新しいパスに移動させる必要があります。このI/Oの経路変更は、そのI/Oの応答時間の遅延を発生させます。この間にさらにI/Oが発行されると、それらはシステム内のキューとなり、システムはプロセス用のメモリを使い果たしてしまう可能性があります。非常に高負荷のI/Oの下では、これらの遅延と低メモリ状態によってシステムが無応答になり、LifeKeeperがこれをサーバのダウンとして検知し、フェイルオーバーを開始することがあります。</p> <p>この問題が発生する頻度には多くの要因が影響を及ぼします。</p> <ul style="list-style-type: none"> プロセッサの速度は、I/Oがキューに保持される速さに影響します。高速なプロセッサでは、障害が検知される頻度が高くなります。 システムメモリの搭載量は、システムが無応答になるまでにキューに保持できるI/Oの数に影響します。メモリが多いシステムでは、障害が検知される頻度が低くなります。 使用するLUNの数は、キューに保持できるI/Oの量に影響します。 I/Oの特性は、キューに保持されるI/Oの量に影響します。問題が発生したテストケースでは、ディスクにデータを無制限に書き込んでいました。ほとんどのアプリケーションは、データの読み取りと書き込みの両方を行うはずですが、フェイルオーバーを待って読み取りがブロックされることで書き込みも抑制され、結果的にI/O速度が減少して障害検知の頻度が低くなります。 <p>例えば、RDACを使用したIBM DS4000のマルチパス構成のテストでは、DS4000へのI/Oスループットを毎秒190MB以上にしてパス障害をシミュレーションした場合にLifeKeeperは約12回に1回サーバの障害を(誤)検出しました。このテストでは、サーバとしてIBM x345 (デュアルXeon 2.8GHzプロセッサとメモリ2GBを搭載)を使用し、DS4400に接続して使用サーバ当たり8ボリューム(LUN)にしました。フェイルオーバーを抑止するために、LifeKeeperのLCMNUMHBEATSパラメータ (</p>
<p>SIOS Protection Suite for Linux テクニカル Page 54</p>	<p>を</p>

項目	説明
<p>大規模ストレージ構成の場合のスイッチオーバーに関する特別な考慮事項</p>	<p>いくつかの大規模ストレージ構成 (例えば、複数の論理ボリュームグループがあり、各ボリュームグループ内に 10 以上の LUN を持つ構成) では、LifeKeeper は障害を検出したときにデフォルトの 300 秒のタイムアウト時間内に sendevent を完了することができない場合があります。その結果、バックアップシステムへのスイッチオーバーが失敗します。サービス状態にならないリソースが生じ、LifeKeeper のログにエラーメッセージが記録されます。</p> <p>大規模ストレージ構成では、 /etc/default/LifeKeeper ファイルの SCSIERROR を「event」から「halt」に変更することを推奨します。これにより LifeKeeper は SCSI エラーの発生時に「halt」を実行します。LifeKeeper はバックアップシステムへのフェイルオーバーに成功するようになります。</p>
<p>HP 3PAR StoreServ 7200 FC</p>	<p>HP 3PAR StoreServ 7200 は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>QLogic QMH2572 8Gb FC HBA for HP BladeSystem c-Class(ファームウェアバージョン 5.06.02 (90d5) を使用した)HP 3PAR StoreServ 7200 (ファームウェア (HP 3PAR OS) バージョン 3.1.2), ドライババージョン 8.03.07.05.06.2-k (RHEL にバンドル) + DMMP (device-mapper-1.02.66-6, device-mapper-multipath-0.4.9-46.el6)</p> <p>テストは RHEL 6.2 (x86_64) を使用した SPS for Linux v8.1.1 で実施されました。</p> <p>注記: 3PAR StoreServ 7200 は、デフォルトのパスチェッカとのリザベーション競合を返します。この競合を回避するには、 /etc/default/LifeKeeper 内に次のパラメータを設定 (追加) してください。</p> <p>DMMP_REGISTRATION_TYPE=hba</p>

項目	説明
<p>HP 3PAR StoreServ 7400 FC</p>	<p>HP 3PAR StoreServ 7400は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>HP 3PAR StoreServ 7400 (ファームウェア (HP 3PAR OS) バージョン3.1.2) + HP DL380p Gen8 + Emulex LightPulse Fibre Channel SCSI HBA (ドライババージョン 8.3.5.45.4p) + DMMP (device-mapper-1.02.66-6, device-mapper-multipath-0.4.9-46.el6)</p> <p>テストはRHEL 6.2 (x86_64) を使用したSPS for Linux v8.1.1 で実施されました。</p> <p>注記: 3PAR StoreServ 7400 は、デフォルトのパスチェッカとのリザベーション競合を返します。この競合を回避するには、<code>/etc/default/LifeKeeper</code> 内に次のパラメータを設定 (追加) してください。</p> <p>DMMP_REGISTRATION_TYPE=hba</p> <p>およびユーザーフレンドリーデバイスマッピングはサポートされません。"multipath.conf" に次のパラメータを設定してください。</p> <p>"user_friendly_names no"</p>
<p>HP 3PAR StoreServ 7400 iSCSI (DMMP Recovery Kit を使用したマルチパス構成)</p>	<p>HP 3PAR StoreServ 7400は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>HP 3PAR StoreServ 7400 (ファームウェア (HP 3PAR OS) version 3.1.3) + HP Ethernet 10Gb 2-port 560SFP+ Adapter (Networkdriver ixgbe-3.22.0.2) + iSCSI (iscsi-initiator-utils-6.2.0.873-10.el6.x86_64)、DMMP (device-mapper-1.02.79-8.el6、device-mapper-multipath-0.4.9-72.el6)。</p> <p>注記: 3PAR StoreServ 7400 iSCSI は、リザベーション競合を返す場合があります。この競合を回避するため、<code>/etc/default/LifeKeeper</code> 内に次のパラメータを設定 (追加) してください。</p> <p>DMMP_REGISTER_IGNORE=TRUE</p>

項目	説明
<p>HP 3PAR StoreServ 7400 iSCSI(Quorum/Witness Kit 使用)</p>	<p>HP 3PAR StoreServ 7400は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>iSCSI (iscsi-initiator-utils-6.2.0.872-21.el6.x86_64) + DMMP (device-mapper-multipath-0.4.9-41 + device-mapper-1.02.62-3) + nx_nic v4.0.588.</p> <p>DMMP と DMMP Recovery Kit を RHEL 6.1 で利用する場合には、Quorum/Witness Server Kit と STONITH 機能を利用する必要があります。SCSI リザベーションを無効にするため、"/etc/default/LifeKeeper"に RESERVATIONS=none を追記する必要があります。</p> <p>サーバは IPMI 2.0 に準拠したインタフェースを備えている必要があります。</p>
<p>HP 3PAR StoreServ 10800 FC</p>	<p>HP 3PAR StoreServ 10800 FC は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>ファームウェア (HP 3PAR OS) バージョン3.1.2 + HP DL380p Gen8 + Emulex LightPulse Fibre Channel HBA (ドライババージョン 8.3.5.45.4p) + DMMP (device-mapper-1.02.66-6, device-mapper-multipath-0.4.9-46 el6)</p> <p>テストはRHEL 6.2 (x86_64)を使用したSPS for Linux v8.1.2 で実施されました。</p> <p>注記: 3PAR StoreServ 10800 FC は、デフォルトのパスチェックとのリザベーション競合を返します。この競合を回避するには、/etc/default/LifeKeeper 内に次のパラメータを設定 (追加) してください。</p> <p>DMMP_REGISTRATION_TYPE=hba</p> <p>およびユーザーフレンドリーデバイスマッピングはサポートされません。"multipath.conf" に次のパラメータを設定してください。</p> <p>"user_friendly_names no"</p>

項目	説明
HP MSA1040/2040fc	<p>HP MSA 2040 Storage FC は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>HP MSA 2040 Storage FC (ファームウェア GL101R002) + HP SN1000Q 16Gb 2P FC HBA QW972A (ファームウェアバージョン 6.07.02, ドライババージョン 8.04.00.12.06.0-k2 (RHELにバンドル)) + DMMP (device-mapper-1.02.74-10, device-mapper-multipath-0.4.9-56).</p> <p>テストは RHEL 6.3 (X86_64) を使用した LifeKeeper for Linux v8.1.2 で実施されました。</p>
HP P9500/XP	<p>SIOS LifeKeeper for Linux v7.2 以降を使用する場合について、Hewlett-Packard 社により認定。テストに使用されたモデルは HP P9500/XP です。以下の環境の LifeKeeper で動作することが認定されています。</p> <ul style="list-style-type: none"> • Red Hat Enterprise for 32-bit, x64 (64-bit; Opteron および Intel EMT64) <p>RHEL 5.3, RHEL 5.4, RHEL 5.5</p> <ul style="list-style-type: none"> • SuSE Enterprise Server for 32-bit, x64 (64-bit; Opteron および Intel EMT64) <p>SLES 10 SP3, SLES 11, SLES 11 SP1</p> <ul style="list-style-type: none"> • ネイティブまたは内蔵のクラスタリングソリューション RHCS および SLE HA
HP StoreVirtual 4330 iSCSI (DMMP Recovery Kit を使用したマルチパス構成)	<p>HP StoreVirtual 4330 は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>HP StoreVirtual 4330 (ファームウェア HP LeftHand OS 10.5) + HP Ethernet 1Gb 4-port 331FLR (Networkdriver tg3-3.125g) + iSCSI (iscsi-initiator-utils-6.2.0.872-41.el6.x86_64)、DMMP (device-mapper-1.02.74-10.el6, device-mapper-multipath-0.4.9-56.el6)</p>

項目	説明
<p>HP StoreVirtual(LeftHand) シリーズ OS (SAN/iQ) バージョン 11.00 iSCSI</p>	<p>HP 社 StoreVirtual(LeftHand) ストレージにおいて、OS(SAN/iQ) バージョン 11.00 をサポートします。本バージョンが使われている全ての StoreVirtual シリーズがサポート対象であり、仮想ストレージ・アプライアンスである StoreVirtual VSA も含まれます。</p> <p>テスト構成は次の通りです。</p> <p>StoreVirtual VSA(11.0.00.1263.0) + RHEL 6.4 (x86_64) + DMMP(device-mapper-1.02.77-9.el6.x86_64, device-mapper-multipath-0.4.9-64.el6.x86_64)</p>
<p>HP StoreVirtual 4730 iSCSI (DMMP Recovery Kit を使用したマルチパス構成)</p>	<p>HP StoreVirtual 4730 は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>HP StoreVirtual 4730 (ファームウェアHP LeftHand OS 11.5) + HP FlexFabric 10Gb 2-port 536FLB Adapter (Networkdriver bnx2x-1.710.40) + iSCSI (iscsi-initiator-utils-6.2.0.873-10.el6.x86_64)、DMMP (device-mapper-1.02.79-8.el6, device-mapper-multipath-0.4.9-72.el6)</p>
<p>HP StoreVirtual(LeftHand) シリーズ LeftHand OS バージョン 11.5 iSCSI (DMMP Recovery Kitを使用したマルチパス構成)</p>	<p>HP 社 StoreVirtual(LeftHand) ストレージにおいて、LeftHand OS バージョン 11.5 をサポートします。本バージョンが使われている全ての StoreVirtual シリーズがサポート対象であり、仮想ストレージ・アプライアンスである StoreVirtual VSA も含まれます。</p> <p>テスト構成は次の通りです。</p> <p>StoreVirtual 4730(11.5.00.0673.0) + RHEL 6.5(x86_64) + DMMP(device-mapper-1.02.79-8.el6.x86_64, device-mapper-multipath-0.4.9-72.el6.x86_64)</p>

項目	説明
HP StoreVirtual 4330 iSCSI (Quorum/Witness Kit 使用)	<p>HP StoreVirtual 4330 は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>HP StoreVirtual 4330 (ファームウェア HP LeftHand OS 10.5) + iSCSI (iscsi-initiator-utils-6.2.0.872-41.el6.x86_64)、bonding(バージョン: 3.6.0)、tg3(バージョン: 3.125g) SCSI リザベーションを無効にするには、"/etc/default/LifeKeeper" に RESERVATIONS=noneを設定します。</p>
IBM San Volume Controller (SVC)	<p>シングルパス構成において、パートナーテストにより認定。SDD Recovery Kit および Device Mapper Multipath Recovery Kit を使用するマルチパス設定において、SIOS Technology Corp. により認定。</p>
IBM Storwize V7000 iSCSI	<p>パートナーテストによりiSCSI (iscsi-initiator-utils-6.2.0.872-34.el6.x86_64) と DMMP (device-mapper-1.02.66-6.el6、device-mapper-multipath-0.4.9-46.el6) を使用する IBM Storwize V7000 (Firmware Version 6.3.0.1)を認定しています。テストは、LifeKeeper for Linux v7.5 と RHEL 6.2 を使用して行われました。</p> <p>制限事項: IBM Storwize V7000 は、Quorum/Witness Server Kit および STONITH と組み合わせて使用する必要があります。/etc/default/LifeKeeper 内で以下の設定により、SCSI リザベーションを無効にしてください。</p> <p style="text-align: center;">RESERVATIONS=none</p>
IBM Storwize V7000 FC	<p>パートナーテストにより Red Hat Enterprise Linux Server Release 6.2 (Tikanga)、HBA: QLE2562 DMMP: 0.4.9-46 を組み合わせたマルチパス構成で認定済み。</p>
IBM Storwize V3700 FC	<p>パートナーテストにより Red Hat Enterprise Linux Server Release 6.5 (Santiago)、HBA: QLE2560、DMMP: 0.4.9-72 を組み合わせたマルチパス構成で認定済み。</p>

ストレージとアダプタの設定

項目	説明
IBM XIV Storage System	<p>パートナーテストにより Red Hat Enterprise Linux Server Release 5.6、HBA: NEC N8190-127 Single CH 4Gbps (Emulex LPe1150 相当品)、XIV Host Attachement Kit: バージョン 1.7.0 を組み合わせたマルチパス構成でのみ認定済み。</p> <p>注記: LifeKeeper を使用する IBM XIV Storage System で 32 個以上の LUN を作成する必要がある場合、詳細については IBM の営業担当者にお問い合わせください。</p>
Dell EqualLogic PS4000/4100/4110/6000/6010/6100/6110/6500/6510	<p>Dell EqualLogic は、SIOS Technology Corp. パートナーによってテスト済みです。テスト構成は次の通りです。Dell EqualLogic PS4000/4100/6000/6100/6110/6500/6510 + DMMP + DMMP Recovery Kit + RHEL 5.3 + iscsi-initiator-utils-6.2.0.868-0.18.el5。LUN 数が大きい場合 (20 以上) は、<code>/etc/default/LifeKeeper</code> の <code>REMOTETIMEOUT</code> 設定を <code>REMOTETIMEOUT=600</code> に変更してください。</p>
<p>富士通</p> <p>ETERNUS DX60 / DX80 / DX90 iSCSI</p> <p>ETERNUS DX60 S2 / DX80 S2 / DX90 S2 iSCSI</p> <p>ETERNUS DX410 / DX440 iSCSI</p> <p>ETERNUS DX410 S2 / DX440 S2 iSCSI</p> <p>ETERNUS DX8100 / DX8400 / DX8700 iSCSI</p> <p>ETERNUS DX8100 S2/DX8700 S2 iSCSI</p> <p>ETERNUS DX100 S3/DX200 S3 iSCSI</p> <p>ETERNUS DX500 S3/DX600 S3 iSCSI</p> <p>ETERNUS DX200F iSCSI</p> <p>ETERNUS DX60 S3 iSCSI</p>	<p>マルチパス環境にてDMMP ARKを利用する際、<code>/etc/multipath.conf</code> に以下の設定が必要となります。</p> <pre> prio alua path_grouping_policy group_by_prio failback immediate no_path_retry 10 Path_checker tur </pre>

項目	説明
富士通	
ETERNUS DX60 / DX80 / DX90 ファイバチャネル、シングルパスおよびマルチパス構成	
ETERNUS DX60 S2 / DX80 S2 / DX90 S2 ファイバチャネル、シングルパスおよびマルチパス構成	
ETERNUS DX410 / DX440 ファイバチャネル、シングルパスおよびマルチパス構成	
ETERNUS DX410 S2 / DX440 S2 ファイバチャネル、シングルパスおよびマルチパス構成	マルチパス環境にてDMMP ARKを利用する際、/etc/multipath.conf に以下の設定が必要となります。
ETERNUS DX8100 / DX8400 / DX8700 ファイバチャネル、シングルパスおよびマルチパス構成	<pre>prio alua path_grouping_policy group_by_prio</pre>
ETERNUS DX8100 S2/DX8700 S2 ファイバチャネル、シングルパスおよびマルチパス構成	<pre>failback immediate no_path_retry 10</pre>
ETERNUS DX100 S3/DX200 S3/DX500 S3/DX600S3 ファイバチャネル、シングルパスおよびマルチパス構成	<pre>Path_checker tur</pre> マルチパス環境にてETERNUSマルチパスドライバを利用する際、設定ファイルにパラメータを設定する必要はありません。
ETERNUS DX200F ファイバチャネル、シングルパスおよびマルチパス構成	
ETERNUS DX60 S3 ファイバチャネル、シングルパスおよびマルチパス構成	
ETERNUS DX8700 S3 / DX8900 S3 ファイバチャネル、シングルパスおよびマルチパス構成	
ETERNUS DX8700 S3 / DX8900 S3 iSCSI、シングルパスおよびマルチパス構成	

項目	説明
<p>NEC iStorage M10e iSCSI (SPS Recovery Kit を使用したマルチパス構成)</p>	<p>NEC iStorage M10e iSCSI は、以下の構成で SIOS Technology Corp. のパートナーによってテスト済みです。</p> <p>NEC iStorage M10e iSCSI + 1GbE NIC + iSCSI (iscsi-initiator-utils-6.2.0.873-10.el6.x86_64)、SPS (sps-utils-5.3.0-0.el6,sps-driver-E-5.3.0-2.6.32.431.el6)</p>

項目	説明
<p>NEC iStorage Storage Path Savior Multipath I/O</p>	<p>マルチパスデバイスを使用するアプリケーションとファイルシステムの保護: SPS デバイスを使用するアプリケーションやファイルシステムを SPS によって設定し保護するには、SPS Recovery Kit をインストールする必要があります。</p> <p>SPS Kit のインストール後は、1 つ以上のマルチパスデバイスノードを使用するアプリケーション階層を作成するだけで、SPS Kit が提供する新しいリソースタイプが自動的に組み込まれます。</p> <p>マルチパスデバイスノード: SPS Kit を使用するには、すべてのファイルシステムおよび Raw デバイスをネイティブの /dev/sd* デバイスノードではなく、マルチパスデバイスノード (/dev/vpath*) 上にマウントまたは設定する必要があります。</p> <p>SCSI-3 Persistent Reservations の使用: SPS Kit は、リザベーションタイプを「書き込み専用」とする SCSI-3 Persistent Reservations を使用します。この場合、クラスタのあるノードが予約したデバイスは、クラスタの他のノードから読み取り可能のままですが、他のノードからデバイスへの書き込みはできなくなります。このことは、それらの他のノード上で進行中の読み取り専用アクセスのためにファイルシステムをマウントできるという意味ではないことに注意してください。</p> <p>LifeKeeper では、sg_persist ユーティリティを使用してパーシステントリザベーションを発行し、監視します。必要であれば、LifeKeeper は sg_persist(8) ユーティリティをインストールします。</p> <p>テスト環境: SPS Kit は、Emulex HBA および Emulex lpfc ドライバを使用する NEC iStorage ディスクアレイにおいて、テストおよび認定済みです。SPS Kit は、SPS がサポートする他の NEC iStorage D、S、および M でも同様に問題なく動作すると考えられます。</p> <p>[テストされた Emulex HBA]</p> <p>iStorage D-10</p> <p>=====</p> <p>LP952</p> <p>LP9802</p>
<p>SIOS Protection Suite for Linux テクニカルドキュメンテーション Page 64</p>	<p>LP1050</p> <p>LP1150</p> <p>=====</p>

項目	説明
Pure Storage FA-400 シリーズ FC (DMMP Recovery Kit を使用したマルチパス構成)	パートナーテストにより、DMMP Recovery Kit を利用したFC接続のマルチパス構成で認定済みです。
QLogic ドライバ	QLogic アダプタを使用するサポート対象の他のファイバチャネルアレイについては、qla2200 または qla2300 ドライバのバージョン 6.03.00 以降を使用してください。
Emulex ドライバ	サポート対象の Emulex のファイバチャネル HBA については、lpfc ドライバ v8.0.16 以降を使用してください。
Adaptec 29xx ドライバ	Adaptec 29xx を使用するサポート対象の SCSI アレイについては、OS ディストリビューションに付属の aic7xxx ドライバのバージョン 6.2.0 以降を使用してください。

HP のマルチパス I/O 設定

項目	説明
Secure Path を使用するマルチパスクラスタのインストール	Secure Path を使用するマルチパスクラスタを新規にインストール場合は、次の手順を実行します。 <ol style="list-style-type: none"> 1. 選択した OS を各 サーバにインストールします。 2. クラスタハードウェア (FCA2214 アダプタ、ストレージ、スイッチ、およびケーブル) をインストールします。 3. HP Platform Kit をインストールします。 4. HP Secure Path ソフトウェアをインストールします。ここでシステムをリブートする必要があります。Secure Path からストレージへのパスを適切に設定したことを確認してください。詳細については、Secure Path のドキュメンテーションを参照してください。 5. LifeKeeper をインストールします。
Secure Path による永続的デバイスノード	Secure Path は、/dev/spdev/spXX (XX はデバイス名) の形式の「永続的な」デバイスノードをサポートします。これらのノードは、特定の SCSI デバイスノード /dev/sdXX へのシンボリックリンクです。LifeKeeper はこれらのノードを「通常の」SCSI デバイスノード /dev/sdXX であるかのように認識します。LifeKeeper は、デバイスが /dev/sda1 か /dev/sdq1 かを直接検出し、その後正しいデバイスノードを直接使用することにより、リブートおよびクラスタノードをまたがってデバイス名の永続性を独自に維持しています。 <p>注記: SCSI デバイスノードへのシンボリックリンクのサポートは、LifeKeeper v4.3.0 で追加されました。</p>

<p>アクティブ/パッシブコントローラおよびコントローラスイッチオーバー</p>	<p>MSA1000 では、一方のコントローラをアクティブに、他方のコントローラをスタンバイモードにすることによってマルチパスを実装しています。アクティブなコントローラまたはアクティブなコントローラへのパスのいずれかに問題が起きた場合、スタンバイコントローラがアクティブ化されて処理を引き継ぎます。コントローラをアクティブにする場合、コントローラの準備ができるまでにある程度の時間がかかります。アレイ上で設定されている LUN の数に応じて、30 ~ 90 秒の時間を必要とします。この間、ストレージへの I/O は、新しくアクティブになるコントローラに経路変更できるようになるまでブロックされます。</p>
<p>起動時にシングルパスでも通知が発生しない</p>	<p>システムがロードされたときに、サーバがシングルパスでしかストレージにアクセスできない場合でも、この問題に関する通知が発生しません。この問題は、システムがリブートしたときに上記のような物理的なパスの障害が起きると発生しますが、一時的なパス障害でも発生しています。システムをロードするときは、管理者はストレージへのすべてのパスが正しく構成されたことを必ず確認し、構成されていない場合は、ハードウェアの問題を修復するか、システムをリロードして一時的な問題を解決するかいずれかのアクションを取ることを推奨します。</p>

日立 HDLM のマルチパス I/O 設定

<p>マルチパスデバイスを使用するアプリケーションとファイルシステムの保護</p>	<p>HDLM デバイスを使用するアプリケーションやファイルシステムを LifeKeeper によって設定し保護するには、HDLM Recovery Kit をインストールする必要があります。</p> <p>HDLM Kit のインストール後は、マルチパスデバイスノードを 1 つ以上使用するアプリケーション階層を作成するだけで、HDLM Kit が提供する新しいリソースタイプが自動的に組み込まれます。</p>
<p>マルチパスデバイスノード</p>	<p>HDLM Kit を使用するには、すべてのファイルシステムおよび Raw デバイスをネイティブの <code>/dev/sd*</code> デバイスノードではなく、マルチパスデバイスノード (<code>/dev/sddlm*</code>) 上にマウントするか、設定する必要があります。</p>
<p>SCSI-3 Persistent Reservations の使用</p>	<p>HDLM Kit は、リザベーションタイプを「書き込み専用」とする SCSI-3 Persistent Reservations を使用します。この場合、クラスタのあるノードが予約したデバイスは、クラスタの他のノードから読み取り可能のままですが、他のノードからデバイスへの書き込みはできなくなります。このことは、それらの他のノード上で進行中の読み取り専用アクセスのためにファイルシステムをマウントできるという意味ではないことに注意してください。</p> <p>LifeKeeper では、<code>sg_persist</code> ユーティリティを使用してパーシステントリザベーションを発行し、監視します。必要であれば、LifeKeeper は <code>sg_persist(8)</code> ユーティリティをインストールします。</p>

ハードウェア要件	<p>HDLM Kit は、QLogic qla2432 HBA と 8.02.00-k5-rhel5.2-04 ドライバ、および SilkWorm3800 FC スイッチを使用する日立 SANRISE AMS1000 ディスクアレイにおいて、テストおよび認定済みです。HDLM Kit は、他の日立 ディスクアレイでも同様に問題なく動作すると考えられます。HDLM Kit は、SANRISE AMS シリーズ、SANRISE USP、および日立の VSP においても、テストおよび認定済みです。HBA および HBA ドライバは HDLM がサポートするものを使用してください。</p> <p>BR1200 は、Hitachi Data Systems により認定済みです。シングルパスとマルチパスの両方の設定で、RDAC ドライバが必要です。RDAC ドライバを使用する BR1200 設定のみをサポートします。HDLM (HDLM ARK) を使用する BR1200 設定はサポートしません。</p>
マルチパスソフトウェアの要件	<p>HDLM Kit は、以下の HDLM for Linux でテスト済みです</p> <p>05-80, 05-81, 05-90, 05-91, 05-92, 05-93, 05-94, 6.0.0, 6.0.1, 6.1.0, 6.1.1, 6.1.2, 6.2.0, 6.2.1, 6.3.0, 6.4.0, 6.4.1, 6.5.0, 6.5.1, 6.5.2, 6.6.0, 6.6.2, 7.2.0, 7.2.1, 7.3.0, 7.3.1, 7.4.0, 7.4.1, 7.5.0, 7.6.0, 7.6.1, 8.0.0, 8.0.1, 8.1.0, 8.1.1, 8.1.2, 8.1.3, 8.1.4, 8.2.0</p> <p>インストールされている HDLM パッケージに対する既知の依存関係はありません。</p> <p>注記: HDLM 6.0.0 以降から製品名が「Hitachi Dynamic Link Manager Software (HDLM)」に変更されました。6.0.0 (05-9X) より古いバージョンでは、「Hitachi HiCommand Dynamic Link Manager (HDLM)」という製品名です。</p> <p>注記: HDLM version 6.2.1 以降は、HDLM Recovery Kit v6.4.0-2 でサポートされていません。このバージョンの HDLM を使用する必要がある場合は、HDLM Recovery Kit v7.2.0-1 以降と LifeKeeper Core v7.3 以降を使用できます。</p> <p>注記: LVM と HDLM をともに使用する場合、HDLM がサポートするバージョンが必要です。また、フィルタ設定を <code>/etc/lvm/lvm.conf</code> に追加して、システムが <code>/dev/sddlm*</code> に対応する <code>/dev/sd*</code> を検出しないようにする必要があります。詳細については、HDLM のマニュアルの「LVM Configuration」を参照してください。</p>

<p>Linux ディストリビューションの要件</p>	<p>Linux ディストリビューションの要件</p> <p>HDLM は以下のディストリビューションでサポートされています。</p> <p>RHEL 4 (AS/ES) (x86 or x86_64) Update 1、2、3、4、Update 4 セキュリティフィックス (*2)、4.5、4.5 セキュリティフィックス(*4)、4.6、4.6 セキュリティフィックス(*8)、4.7、4.7 セキュリティフィックス(*9)、4.8、4.8 セキュリティフィックス(*12) (x86/x86_64)(*1)</p> <p>RHEL 5、5.1、5.1 セキュリティフィックス(*5)、5.2、5.2 セキュリティフィックス(*6)、5.3、5.3 セキュリティフィックス(*10)、5.4、5.4 セキュリティフィックス(*11)、5.5、5.5 セキュリティフィックス(*13)、5.6、5.6 セキュリティフィックス (*14)、5.7 (x86/x86_64)(*1)、5.8 (x86/x86_64)(*1)</p> <p>RHEL 6、6.1、6.2、6.3、6.4、6.5 (x86/x86_64)(*1)(*15)</p> <p>(*1) AMD Opteron(シングルコア、デュアルコア)あるいは Intel EM64T アーキテクチャ CPU + x86_64 カーネル</p> <p>(*2) 次のカーネルがサポートされています。 x86: 2.6.9-42.0.3.EL, 2.6.9-42.0.3.ELsmp, 2.6.9-42.0.3.ELhugemem x86_64: 2.6.9-42.0.3.EL, 2.6.9-42.0.3.ELsmp, 2.6.9-42.0.3.ELlargesmp</p> <p>(*3) 日立 では、RHEL4 U2 の環境をサポートしていません。</p> <p>(*4) 次のカーネルがサポートされています。 x86: 2.6.9-55.0.12.EL, 2.6.9-55.0.12.ELsmp, 2.6.9-55.0.12.ELhugememx 86_64: 2.6.9-55.0.12.EL, 2.6.9-55.0.12.ELsmp, 2.6.9-55.0.12.ELlargesmp</p> <p>(*5) 次のカーネルがサポートされています。 x86: 2.6.18-53.1.13.el5, 2.6.18-53.1.13.el5PAE, 2.6.18-53.1.21.el5, 2.6.18-53.1.21.el5PAE x86_64: 2.6.18-53.1.13.el5, 2.6.18-53.1.21.el5</p> <p>(*6) 次のカーネルがサポートされています。 x86: 2.6.18-92.1.6.el5, 2.6.18-92.1.6.el5PAE, 2.6.18-92.1.13.el5, 2.6.18-92.1.13.el5PAE, 2.6.18-92.1.22.el5, 2.6.18-92.1.22.el5PAE x86_64: 2.6.18-92.1.6.el5, 2.6.18-92.1.13.el5, 2.6.18-92.1.22.el5</p> <p>(*7) 次のカーネルがサポートされています。 x86: 2.6.9-34.0.2.EL, 2.6.9-34.0.2.ELsmp, 2.6.9-34.0.2.ELhugemem x86_64: 2.6.9-34.0.2.EL, 2.6.9-34.0.2.ELsmp, 2.6.9-34.0.2.ELlargesmp</p> <p>(*8) 次のカーネルがサポートされています。 x86: 2.6.9-67.0.7.EL, 2.6.9-67.0.7.ELsmp, 2.6.9-67.0.7.ELhugemem, 2.6.9-67.0.22.EL, 2.6.9-67.0.22.ELsmp, 2.6.9-67.0.22.ELhugemem x86_64: 2.6.9-67.0.7.EL, 2.6.9-67.0.7.ELsmp, 2.6.9-67.0.7.ELlargesmp, 2.6.9-67.0.22.EL, 2.6.9-67.0.22.ELsmp, 2.6.9-67.0.22.ELlargesmp</p> <p>(*9) 次のカーネルがサポートされています。 x86: 2.6.9-78.0.1.EL, 2.6.9-78.0.1.ELsmp, 2.6.9-78.0.1.ELhugemem, 2.6.9-78.0.5.EL, 2.6.9-78.0.5.ELsmp, 2.6.9-78.0.5.ELhugemem, 2.6.9-78.0.8.EL, 2.6.9-78.0.8.ELsmp, 2.6.9-78.0.8.ELhugemem, 2.6.9-78.0.17.EL, 2.6.9-78.0.17.ELsmp, 2.6.9-78.0.17.ELhugemem, 2.6.9-78.0.22.EL, 2.6.9-78.0.22.ELsmp, 2.6.9-78.0.22.ELhugemem x86_64: 2.6.9-78.0.1.EL, 2.6.9-78.0.1.ELsmp, 2.6.9-78.0.1.ELlargesmp, 2.6.9-78.0.5.EL, 2.6.9-78.0.5.ELsmp, 2.6.9-78.0.5.ELlargesmp, 2.6.9-78.0.8.EL, 2.6.9-78.0.8.ELsmp, 2.6.9-78.0.8.ELlargesmp, 2.6.9-78.0.17.EL, 2.6.9-78.0.17.ELsmp, 2.6.9-78.0.17.ELlargesmp, 2.6.9-78.0.22.EL, 2.6.9-78.0.22.ELsmp, 2.6.9-78.0.22.ELlargesmp</p>
	<p>2.6.9-78.0.17.ELlargesmp, 2.6.9-78.0.22.EL, 2.6.9-78.0.22.ELsmp, 2.6.9-78.0.22.ELlargesmp</p> <p>SIOS Protection Suite for Linux テクニカルドキュメンテーション Page 68</p> <p>(*10) 次のカーネルがサポートされています。 x86: 2.6.18-128.1.10.el5, 2.6.18-128.1.10.el5PAE, 2.6.18-128.1.14.el5, 2.6.18-</p>

インストール要件	HDLM Recovery Kit をインストールする前に HDLM ソフトウェアをインストールする必要があります。また、SCSI デバイスから HDLM デバイスに環境を転送したい場合は、HDLM 環境を設定した後、インストールセットアップスクリプトを実行する必要があります。そのようにしないと、sg3_utils がインストールされません。
HDLM パスの追加または修復	LifeKeeper は、HDLM リソースを起動する場合、パーシステントリザベーションを確立してその時点でアクティブなパスに登録します。最初のリザベーションの後に新しいパスが追加されるか、障害が起きたパスが修復されて HDLM がそのパスを自動的に再度アクティブにした場合、そのパスは、LifeKeeper が HDLM リソースに対する次の quickCheck を実行するまでリザベーションの一部として登録されません。その時点までに HDLM がそのパスに対する書き込みを許可した場合、リザベーションコンフリクトが発生し、システムのメッセージファイルに競合が記録されます。HDLM ドライバは、登録されたパスでそれらの I/O を再試行するため、アプリケーションにとっては検出可能な障害になりません。quickCheck によるパスの登録が完了すると、その後の書き込みは成功します。quickCheck がリザベーションコンフリクトを検出すると、ステータスが「Offline(E)」に変更されます。ステータスが「Offline(E)」の場合、ユーザはオンラインの HDLM コマンドを使用して手動でステータスを「Online」に変更する必要があります。

OS のバージョン / アーキテクチャ										
RHEL4										
U1- U4	U3 セ キュリ ティ フィク ス(*7)	U4 セ キュリ ティ フィク ス(*2)	4.5	4.5 セ キュリ ティ フィク ス(*4)	4.6	4.6 セ キュリ ティ フィク ス(*8)	4.7	4.7 セ キュリ ティ フィク ス(*9)	4.8	4.8 セキュ リティ フィク ス (*12)
x86/x86_64										

日立 HDLM のマルチパス I/O 設定

HDLM	05-80 05-81 05-90	X										
	05-91 05-92	X		X								
	05-93	X(*3)		X	X							
	05-94	X(*3)		X	X	X	X	X				
	6.0.0	X(*3)		X	X	X	X	X	X	X	X	X
	6.0.1	X(*3)		X	X	X	X	X	X	X	X	X
	6.1.0	X(*3)		X	X	X	X	X	X	X	X	X
	6.1.1	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.1.2	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.2.0	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.2.1	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.3.0	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.4.0	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.4.1	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.5.0	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.5.1	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.5.2	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.6.0	X(*3)	X	X	X	X	X	X	X	X	X	X
	6.6.2	X(*3)	X	X	X	X	X	X	X	X	X	X
	7.2.0	X(*3)	X	X	X	X	X	X	X	X	X	X
	7.2.1	X(*3)	X	X	X	X	X	X	X	X	X	X
7.3.0 以降	X(*3)	X	X	X	X	X	X	X	X	X	X	

LifeKeeper	v6.0	X	X	X									
------------	------	---	---	---	--	--	--	--	--	--	--	--	--

日立 HDLM のマルチパス I/O 設定

v6.0 (v6.0.1-2 以降)												
v6.1 (v6.1.0-5 以降)	X	X	X									
v6.2 (v6.2.0-5 以降)	X	X	X	X	X	X	X					
v6.2 (v6.2.2-1 以降)	X	X	X	X	X	X	X					
v6.3 (v6.3.2-1 以降)	X	X	X	X	X	X	X					
v6.4 (v6.4.0- 10 以降)	X	X	X	X	X	X	X	X	X			
v7.0 (v7.0.0-5 以降)	X	X	X	X	X	X	X	X	X	X	X	X
V 7.1 (v7.1.0-8 以降)	X	X	X	X	X	X	X	X	X	X	X	X
V7.2 (v7.2.0- 10 以降)	X	X	X	X	X	X	X	X	X	X	X	X
V 7.3 (v7.3.0- 21 以降)	X	X	X	X	X	X	X	X	X	X	X	X
V 7.4 (v7.4.0- 63 以降)	X	X	X	X	X	X	X	X	X	X	X	X
V 7.5 (v7.5.0- 3640 以 降)	RHEL4 は、LifeKeeper の v7.5 以降ではサポートされません。											

日立 HDLM のマルチパス I/O 設定

HDLM ARK	6.0.1-2	X	X	X	X	X	X	X				
	6.1.0-4	X	X	X	X	X	X	X				
	6.2.2-3	X	X	X	X	X	X	X				
	6.2.3-1	X	X	X	X	X	X	X	X	X	X	X
	6.4.0-2	X	X	X	X	X	X	X	X	X	X	X
	7.0.0-1	X	X	X	X	X	X	X	X	X	X	X
	7.2.0-1	X	X	X	X	X	X	X	X	X	X	X
X = サポートあり、空白 = サポートなし												

RHEL5																				
		未 更 新	5- 1	5.1 セ キ ュリ ティ	5- 2	5.2 セ キ ュリ ティ	5.3- X	5.3 セ キ ュリ ティ	5- 4	5.4 セ キ ュリ ティ	5- 5	5.5 セ キ ュリ ティ	5- 6	5.6 セ キ ュリ ティ	5.7 セ キ ュリ ティ	5.8 セ キ ュリ ティ	5.9 セ キ ュリ ティ	5.1- 0	5.1- 1	
				フィ ツク ス (*5)		フィ ツク ス (*6)		フィ ツク ス (*1- 0)		フィ ツク ス (*1- 1)		フィ ツク ス (*1- 3)		フィ ツク ス (*1- 4)		フィ ツク ス (*1- 6)		フィ ツク ス (*1- 7)		フィ ツク ス (*2- 3)
x86/x86_64																				

日立 HDLM のマルチパス I/O 設定

05-94	X	X															
6.0.0	X	X	X	X	X												
6.0.1	X	X	X	X	X												
6.1.0	X	X	X	X	X												
6.1.1	X	X	X	X	X												
6.1.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.2.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.2.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.3.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.4.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.4.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.5.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.5.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.5.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.6.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.6.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.2.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.2.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.3.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.3.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.4.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.4.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.5.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.6.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7.6.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.0.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.0.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.1.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.1.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.1.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.1.3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.1.4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.2.0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

LifeKeeper	v6.0 (v6.0.- 1-2 以 降)																										
------------	--------------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

v7.0																				
(v7.0-0-5以降)	X	X	X	X	X	X	X	X	X											

v7.1																				
(v7.1-0-8以降)	X	X	X	X	X	X	X	X	X	X	X									

v7.2																				
(v7.2-0-10以降)	X	X	X	X	X	X	X	X	X	X	X	X								

v7.3																				
(v7.3-0-21以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X						

v7.4																			
(v7.4-0-63以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				

v7.5 (v7.5- 0- 3640 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
-------------------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

v8.0																			
(v8.0-0-510以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

v8.1																				
(v8.1.- 1- 5620 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

日立 HDLM のマルチパス I/O 設定

v8.2 (v8.2-0-6213 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
v8.2.1 (v8.2-1-6353 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
v8.3.0 (v8.3-0-6389 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
v8.3.1 (v8.3-1-6397 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
v8.3.2 (v8.3-2-6405 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
v8.4.0 (v8.4-0-6427 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
v8.4.1 (v8.4-1-6449 以降)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

日立 HDLM のマルチパス I/O 設定

HDLM ARK	6.0.1-2																	
	6.1.0-4	X	X															
	6.2.2-3	X	X	X														
	6.2.3-1	X	X	X	X	X												
	6.4.0-2	X	X	X	X	X	X	X										
	7.0.0-1	X	X	X	X	X	X	X	X	X	X	X						
	7.2.0-1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.1.1-5620	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.2.0-6213	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.2.1-6353	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.3.0-6389	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.3.1-6397	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.3.2-6405	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	8.4.0-6427	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8.4.1-6449	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
X = サポートあり、空白 = サポートなし																		

		OS のバージョン / アーキテクチャ						
		RHEL6						
		6	6.1	6.2 セキュ リティ フィッ クス(*18)	6.3 セキュ リティ フィッ クス(*19)	6.4 セキュ リティ フィッ クス(*20)	6.5 セキュ リティ フィッ クス(*21)	6.6 セキュ リティ フィッ クス(*23)
		x86/x86_64						
HDLM	6.5.0							
	6.5.1							
	6.5.2	X						
	6.6.0	X						
	6.6.2	X						
	6.6.2-01	X	X					
	7.2.0	X	X	X				
	7.2.1	X	X	X				
	7.3.0	X	X	X				
	7.3.1	X	X	X				
	7.4.0	X	X	X	X	X	X	X
	7.4.1	X	X	X	X	X	X	X
	7.5.0	X	X	X	X	X	X	X
	7.6.0	X	X	X	X	X	X	X
	7.6.1	X	X	X	X	X	X	X
	8.0.0	X	X	X	X	X	X	X
	8.0.1	X	X	X	X	X	X	X
	8.1.0	X	X	X	X	X	X	X
	8.1.1	X	X	X	X	X	X	X
	8.1.2	X	X	X	X	X	X	X
8.1.3	X	X	X	X	X	X	X	
8.1.4	X	X	X	X	X	X	X	
8.2.0	X	X	X	X	X	X	X	

LifeKeeper	V 7.0								
------------	-------	--	--	--	--	--	--	--	--

	(v7.0.0-5 以降)							
--	------------------	--	--	--	--	--	--	--

V 7.1								
-------	--	--	--	--	--	--	--	--

	(v7.1.0-8 以降)							
--	------------------	--	--	--	--	--	--	--

V7.2								
------	--	--	--	--	--	--	--	--

	(v7.2.0-10 以降)							
--	-------------------	--	--	--	--	--	--	--

	V7.3	X						
--	------	---	--	--	--	--	--	--

	(v7.3.0-21 以降)							
--	-------------------	--	--	--	--	--	--	--

	V 7.4	X						
--	-------	---	--	--	--	--	--	--

	(v7.4.0-63 以降)							
--	-------------------	--	--	--	--	--	--	--

	V7.5	X	X	X	X			
--	------	---	---	---	---	--	--	--

(v7.5.0-3640 以降)								
------------------	--	--	--	--	--	--	--	--

日立 HDLM のマルチパス I/O 設定

v8.0								
(v8.0.0-510 以降)	X	X	X	X				
v8.1								
(v8.1.1-5620 以降)	X	X	X	X				
v8.1.2								
(v8.1.2-5795 以降)	X	X	X	X	X			
v8.2.0								
(v8.2.0-6213 以降)	X	X	X	X	X			
v8.2.1								
(v8.2.1-6353 以降)	X	X	X	X	X	X		
v8.3.0								
(v8.3.0-6389 以降)	X	X	X	X	X	X		
v8.3.1								
(v8.3.1-6937 以降)	X	X	X	X	X	X		
v8.3.2								
(v8.3.2-6405 以降)	X	X	X	X	X	X	X	
v8.4.0								
(v8.4.0-6427 以降)	X	X	X	X	X	X	X	
v8.4.1								
(v8.4.1-6449 以降)	X	X	X	X	X	X	X	

HDLM ARK	7.0.0-1							
	7.2.0-1	X	X	X	X			
	8.1.1-5620	X	X	X	X			
	8.1.2-5795	X	X	X	X	X		
	8.2.0-6213	X	X	X	X	X		
	8.2.1-6353	X	X	X	X	X	X	X
	8.3.0-6389	X	X	X	X	X	X	X
	8.3.1-6397	X	X	X	X	X	X	X
	8.3.2-6405	X	X	X	X	X	X	X
	8.4.0-6427	X	X	X	X	X	X	X
	8.4.1-6449	X	X	X	X	X	X	X
X = サポートあり、空白 = サポートなし								

Device Mapper のマルチパス I/O 設定

Device Mapper Multipath デバイスを使用するアプリケーションとファイルシステムの保護	<p>Device Mapper Multipath デバイスを使用するアプリケーションやファイルシステムを LifeKeeper によって設定し保護するには、Device Mapper Multipath (DMMP) Recovery Kit をインストールする必要があります。</p> <p>DMMP Kit のインストール後は、1 つ以上のマルチパスデバイスノードを使用するアプリケーション階層を作成するだけで、DMMP Kit が提供する新しいリソースタイプが自動的に組み込まれます。</p>
マルチパスデバイスノード	<p>DMMP Kit を使用するには、すべてのファイルシステムおよび Raw デバイスをネイティブの /dev/sd* デバイスノードではなく、マルチパスデバイスノード上にマウントまたは設定する必要があります。ディスク全体を利用できるサポート対象のマルチパスデバイスノードは、/dev/dm-#、/dev/mapper/<uuid>、/dev/mapper/<user_friendly_name> および /dev/mpath/<uuid> です。ディスクのパーティションに対応するには、/dev/mapper ディレクトリに作成される各パーティション用のデバイスノードを使用します。</p>

<p>SCSI-3 Persistent Reservations の使用</p>	<p>Device Mapper Multipath Recovery Kit は、リザーベーションタイプを「書き込み専用」とする SCSI-3 Persistent Reservations を使用します。この場合、クラスタのあるノードが予約したデバイスは、クラスタの他のノードから読み取り可能なままですが、他のノードからデバイスへの書き込みはできなくなります。このことは、それらの他のノード上で進行中の読み取り専用アクセスのためにファイルシステムをマウント <u>できるという意味ではない</u>ことに注意してください。</p> <p>LifeKeeper では、sg_persist ユーティリティを使用してパーシステントリザーベーションを発行、監視します。必要であれば、LifeKeeper は sg_persist(8) ユーティリティをインストールします。</p> <p>EMC Symmetrix (VMAX を含む) アレイをマルチパスソフトウェアおよび LifeKeeper と組み合わせて使用する場合は、SCSI-3 Persistent Reservations を LUN 単位で有効にする必要があります。これは、DMMP と PowerPath の両方に当てはまります。</p>
---	--

ハードウェア要件	<p>Device Mapper Multipath Kit は、EMC CLARiiON CX300、HP EVA 8000、HP MSA1500、HP P2000、IBM SAN Volume Controller (SVC)、IBM DS8100、IBM DS6800、IBM ESS、DataCore SANsymphony、HDS 9980V を使用して SIOS Technology Corp. によりテスト済みです。Device Mapper Multipath のサポートについては、ストレージベンダにお問い合わせください。</p> <p>CX300 および VNX シリーズでリザベーションのサポートを有効にするには、リザベーションに従うようにハードウェアハンドラに通知する必要があります。このアレイ用に <code>/etc/multipath.conf</code> 内に次のパラメータを設定してください。</p> <pre>hardware_handler 「3 emc 0 1」</pre> <p>HP MSA1500 の場合、デフォルトのパスチェッカ (tur) とのリザベーションコンフリクトを返します。これによりスタンバイノードは、すべてのパスを障害であると判定します。この状態を回避するには、このアレイ用に <code>/etc/multipath.conf</code> 内に次のパラメータを設定してください。</p> <pre>path_checker readsector0</pre> <p>HP 3PAR F400 は、デフォルトのパスチェッカとのリザベーション競合を返します。この競合を回避するには、このアレイ用に <code>/etc/default/LifeKeeper</code> 内に次のパラメータを設定 (追加) してください。</p> <pre>DMMP_REGISTRATION_TYPE=hba</pre> <p>HDS 9980V の場合、以下の設定が必要です。</p> <ul style="list-style-type: none"> • Host mode: 00 • System option: 254 (有効にする必要がある。すべてのサーバに影響を与えるグローバルな HDS 設定) • Device emulation: OPEN-V <p>HDS の DMMP 設定の詳細については、HDS ドキュメンテーション「Suse Linux Device Mapper Multipath for HDS Storage」または「Red Hat Linux Device Mapper Multipath for HDS Storage」の v1.15 以降を参照してください。このドキュメンテーションでは、互換性のある multipath.conf ファイルも提供しています。</p> <p>ファームウェアバージョン 6 以降を使用する EVA ストレージでは、DMMP Recovery Kit v6.1.2-3 以降が必要です。これ以前のバージョンの DMMP Recovery Kit は、バージョン 6 より前のファームウェアを使用する EVA ストレージでサポートされています。</p>
マルチパスソフトウェアの要件	<p>SUSE の場合、multipath-tools-0.4.5-0.14 以降が必要です。</p> <p>Red Hat の場合、device-mapper-multipath-0.4.5-12.0.RHEL4 以降が必要です。</p> <p>ベンダが提供する最新のマルチパスツールの組み合わせを使用することを推奨します。このマルチパス製品の機能と安定性は急速に向上しています。</p>
Linux ディストリビューションの要件	<p>IBM などの一部のストレージベンダは、現時点では SLES 11 を使用する DMMP を認定していません。</p> <p>SIOS Technology Corp. は、DMMP、SLES 11、EMC CLARiiON および Symmetrix アレイの組み合わせで報告された問題を現在調査中です。</p>

一時的なパス障害

Device Mapper Multipath デバイスで I/O テストを実行中に、サーバのリブートなどの SAN 上の操作によって一時的なパスの障害が報告されることは珍しくありません。ほとんどの場合、結果として単に 1 つのパスだけが障害となり他のパスは I/O を送信するため、パフォーマンスへのわずかな影響以外に検出される障害はありません。ただし一部のケースでは、複数のパスが障害として報告され、機能するパスがまったくない状態になることがあります。この状態では、ファイルシステムやデータベースなどのアプリケーションからは I/O エラーが発生しているように見えます。このような障害を排除する上で、Device Mapper Multipath およびベンダのサポートはこれまでに大きく改善されました。ただし、まだ問題が発生することはあります。このような状況を回避するため、以下の措置を検討してください。

1. ディスクアレイベンダの手順に従ってマルチパス構成が正しく設定されていることを確認します。
2. 「failback」機能の設定を確認します。この機能は、パスの障害および修復後にパスを再度アクティブにするまでの時間を指定します。「immediate」に設定した場合、パスがオンラインに戻るとすぐに使用を再開することを意味します。整数に設定した場合、パスがオンラインに戻ってから使用を再開するまでの秒数を意味します。10 ~ 15 に設定すると、一般的に SAN 上のスラッシングを回避するのに十分な処理時間が得られます。
3. 「no_path_retry」機能の設定を確認します。この機能は、すべてのパスに障害が発生したときに Device Mapper Multipath がやるべきことを指定します。10 ~ 15 に設定することを推奨します。この機能によって、すべてのパスに障害が起きた一時的なイベントを「乗り切る」方法が提供され、復旧に必要な妥当な時間を稼ぐことができます。LifeKeeper の DMMP Kit はストレージへの I/O を監視しており、4 分以内に応答がなかった場合、LifeKeeper はスタンバイサーバにリソースをスイッチオーバーします。注記：簡単には削除できない I/O が発生するため、LifeKeeper では「no_path_retry」設定を「queue」に設定することは推奨されません。それらを削除するためのメカニズムは、新しいバージョンの DM に含まれており、デバイスの設定を次のように変更できます。

```
/sbin/dmsetup message -u 'DMid' 0 fail_if_no_path
```

これによって no_path_retry の設定が一時的に「fail」に変更され、未処理の I/O がすべて失敗します。ただし、multipathd は、no_path_retry をいつでもデフォルトにリセットできます。失敗した I/O を消去するために設定が fail_if_no_path に変更されたときは、デバイスにアクセスする前に (手動または LifeKeeper によって) 設定をデフォルトにリセットする必要があります。

「no_path_retry」が「queue」に設定されていて障害が発生した場合、LifeKeeper はリソースをスタンバイサーバにスイッチオーバーします。ただし、LifeKeeper は失敗した I/O を削除しません。この I/O を消去するための推奨の方法はリブートですが、上記の dmsetup コマンドを使用して管理者が消去することもできます。I/O を消去しておかないと、他方のサーバでリソースがサービス休止状態になってロックを解放した場合にこの「古い」I/O が発行される事態になってデータの破損が発生します。

LifeKeeper I-O フェンシングの概要

I/O フェンシングは、障害ノードをデータから切り離すことにより、共有ストレージへの非協調的なアクセスを防止する機能です。複数のサーバが同じデータにアクセスできる環境では、データの破損を防ぐためにすべての書き

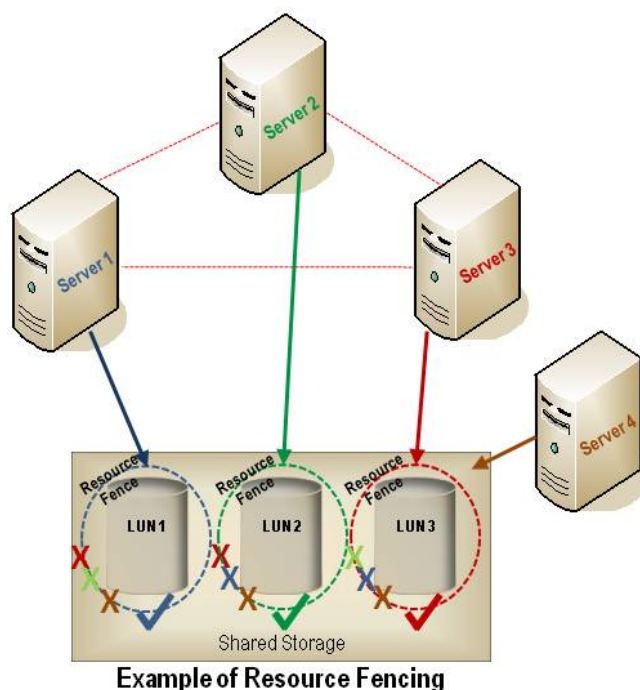
込みを制御された方法で行うことが不可欠です。障害検知メカニズムが破綻した場合、この破綻によってノード障害に似た状況になるため問題が発生します。例えば、2ノードクラスターで2つのノード間の接続に障害が発生した場合、各ノードは相手側に障害が発生したと「思い込む」ため、両方のノードがデータに対する制御を獲得しようと試みてデータの破損につながります。I/O フェンシングは、特定のノードからのデータアクセスをブロックすることによりこのデータ破損のリスクを排除します。

SCSI リザベーション

SCSI リザベーションを利用したストレージフェンシング

LifeKeeper for Linux は、リソースフェンシングとノードフェンシングの両方をサポートしますが、主要なフェンシングメカニズムは、SCSI リザベーションによるストレージフェンシングです。共有ストレージに対する最高レベルのデータ保護を提供するこのフェンシングを使用すると、非常に粒度の高いLUNレベルのロックによって最大限の柔軟性と最大限のセキュリティが可能になります。このアーキテクチャでベースとなる共有リソース(LUN)は、プライマリ quorum デバイスです。quorum は、共有ストレージに対する排他的なアクセスと定義できます。つまり、この共有ストレージは1度に1台のサーバからしかアクセスできません。quorum (排他的アクセス)を持つサーバは、「プライマリ」の役割を持ちます。quorum の確立 (排他的アクセスをどのサーバに与えるか) は、「quorum デバイス」によって行われます。

上述の通り、リザベーションが有効の場合、quorum デバイスはその共有リソースです。共有リソースは、共有リソースに対するリザベーションを持つサーバを判断して quorum を確立します。これにより、ある1つのサーバがそのLUNにアクセスできる限り、クラスターは実質的には単一のサーバで運用されることとなります。



SCSI リザベーションは、共有のユーザデータを保護し、LifeKeeper が指定するシステムのみデータを変更できるようにします。クラスター内外の他のシステムがそのデータを変更することは許可されません。さらに SCSI リザベーションによって、クラスター内の複数のサーバで障害が起きた場合に、LifeKeeper 保護下のアプリケーションは共有

のユーザデータに安全にアクセスできます。サーバの多数派 quorum は必要ありません。唯一の要件は、共有データの所有権の帰属が確立していることです。

quorum/witness 機能を追加すると、quorum のメンバーシップを確立することができます。このメンバーシップがない場合、スプリットブレインの状態で複数のサーバ(場合によっては全サーバ)がお互いを終了させることも考えられます。リザベーションが有効になっている構成に Watchdog を追加すると、部分的にサーバがハングしている状態からリカバリするメカニズムが提供されます。ハングしたサーバが LifeKeeper に検出されないような場合に、Watchdog はリカバリを開始します。また、サーバがハングしてさらにリザベーションが奪われた場合に、Watchdog はそのサーバをリポートしてリカバリを開始することができます。

I/O フェンシングのための代替方式

SCSI リザベーションを利用したリソースフェンシングに加えて、LifeKeeper for Linux はリザベーションの無効化もサポートします。リザベーションが有効か無効にかかわらず、以下の2つの点に注意すべきです。

- ストレージへのアクセスは LifeKeeper が制御する必要があります。
- ストレージへの意図しないアクセス(ファイルシステムのマウント、手動の fsck など)が発生しないように細心の注意を払う必要があります。

以上の2つのルールを順守してリザベーションを有効にすると、LifeKeeper はたいていのエラーを防止できます。リザベーションが(単独で)無効になった状態は、保護がない状態です。したがって、この保護を実現するには、他の選択肢を検討する必要があります。以降のセクションでは、SCSI リザベーションなしでも LifeKeeper で非常に信頼性の高い構成を実現できる各種のフェンシングオプションと代替方式を説明します。

リザベーションの無効化

リザベーションを使用すると、共有ストレージに対する最高レベルのデータ保護が可能になりますが、場合によっては、リザベーションを使用できず LifeKeeper 内で無効にしなければならないことがあります。リザベーションを無効にすると、複数のシステムが意図的または意図せずストレージにアクセスしようとする場合にストレージが調停役として動作することがなくなります。

そのため、システムハング、システムビジー、またはサーバが停止したように見えるあらゆる状況に対応できるように、クラスタメンバーシップによってストレージをフェンシングする別の方法を採用することを検討する必要があります。

リザベーションがなくても信頼性の高い構成を実現する鍵は、フェイルオーバーが発生したとき、「他の」サーバの電源がオフになったこと、または電源が再投入されたことを「知る」ことです。この要件を満たすために利用可能なフェンシングオプションは4つあり、これらは SCSI リザベーションなしでも LifeKeeper で非常に信頼性の高い構成を実現できます。オプションは以下の通りです。

- [STONITH](#) (Shoot the Other Node in the Head) (高信頼性のインターコネクト、すなわちサーバと STONITH デバイスとの間のシリアル接続を使用) - STONITH は、サーバがクラスタの一部とみなされなくなったときに、そのサーバを物理的に停止させたり、電源を切断したりする技術です。LifeKeeper フェイルオーバーのイベント時にサーバの電源を切断する機能をサポートしています。これにより共有データへの安全なアクセスを保証します。このオプションはリザベーションと同様の信頼性を提供できますが、利用できるのは物理的に同じ場所に配置された2つのノードに限定されます。
- [Quorum/Witness](#) - Quorum/Witness サーバは、特にクラスタサーバが異なる場所に配置されている場合に、クラスタ内でのメンバーシップを確認するために使用されます。このオプションはスプリットブレインに対

応できるもののシステムハングに対応できないため、単独での使用は推奨されません。

- [Watchdog](#) - Watchdog は、サーバーの状態を監視します。問題が検出されると、問題のあるサーバは再起動または電源を切断されます。このオプションではサーバーのハングからのリカバリはできますが、スプリットブレインには対応できません。したがって、このオプションもまた単独での使用は推奨されません。
- CONFIRM_SO - このオプションでは自動フェイルオーバーを無効にする必要があります。そのため、信頼性が非常に高い(管理者のスキルによって異なります)一方で、可用性はあまり高くありません。

これらの代替フェンシング方式はどれも単独では十分とは言えませんが、組み合わせて使用することで非常に信頼性の高い構成を実現できます。

非共有ストレージ

非共有ストレージ環境で LifeKeeper を使用する計画の場合、共有ストレージに存在するデータ破損のリスクは問題にならないため、リザベーションは不要です。ただし、データの部分的または完全な再同期およびマージが必要な場合があります。信頼性と可用性を最適化するには、非共有ストレージでも上記のオプションを検討する必要があります。

注記: 各オプションの信頼性と可用性の比較の詳細については、[I/O フェンシング比較表](#) を参照してください。

完全なデータ保護を実現するオプションはないことを理解することは重要です。ただし、以下のように組み合わせるとリザベーションとほぼ同等レベルの保護を実現できます。

リザベーションを使用しない I/O フェンシングの設定

ノードフェンシングをサポートするクラスタを構成するには以下の手順を実行します。

1. LifeKeeper を停止します。
2. LifeKeeper 内での SCSI リザベーションの使用を無効にします。無効にするには、クラスタのすべてのノードで LifeKeeper のデフォルトファイル /etc/default/LifeKeeper を編集します。Reservations 変数を追加または修正して「none」にします (RESERVATIONS="none")。このオプションはリザベーションを利用できない場合のみ使用することに注意してください。
3. I/O フェンシングを提供する STONITH デバイスを用意して設定します。この設定では、STONITH デバイスが reboot コマンドではなく、poweroff コマンドをシステムに対して実行するようにします。LifeKeeper の通信が何らかの理由で中断したとき、手動操作によって同時に両ノード上のデバイス階層を In Service の状態にしないように注意してください。
4. 必要に応じて、quorum/witness サーバを用意して設定します。quorum/witness サーバを設定、使用する詳細な手順と情報については、[Quorum/Witness Server Support Package](#) トピックを参照してください。

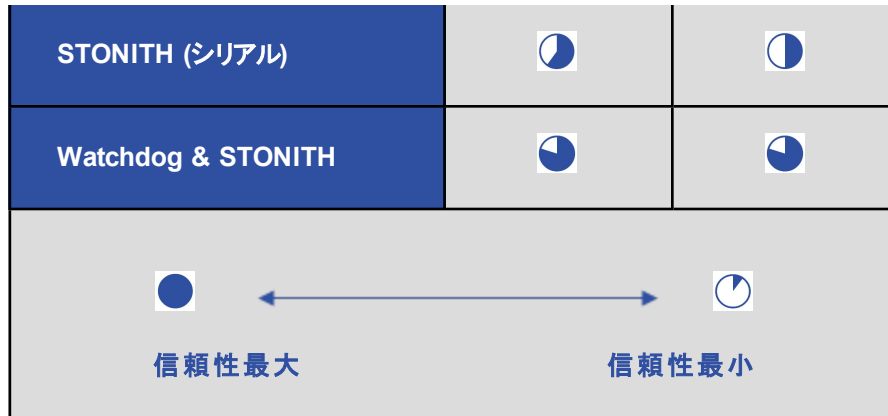
注記: サイトの障害時に最良の保護機能を提供するため、quorum/witness サーバはクラスタ内の別サーバから離れた場所にある必要があります。

5. 必要に応じて、Watchdog を設定します。詳細については、[Watchdog](#) トピックを参照してください。

I/O フェンシング表

I/O フェンシング表

	制御分離	ハングしたサーバ
Reservations オン		
単体		
Quorum/Witness		
Watchdog		
Watchdog & Quorum/Witness		
STONITH(シリアル)		
Reservations オフ		
なし		
STONITH (シリアル)		
CONFIRM_SO*		
Quorum/Witness		
Watchdog		
非共有ストレージ		
デフォルトの機能		
Quorum/Witness		
CONFIRM_SO*		
Watchdog		



* CONFIRM_SO は信頼性が高いが(管理者の知識による)、自動フェイルオーバーがオフになっているという事実により可用性が低い。

Quorum/Witness

Quorum/Witness Server Support Package for LifeKeeper

機能の概要

LifeKeeper Core の既存のフェイルオーバープロセスに Quorum/Witness Server Support Package for LifeKeeper (steeleye-lkQWK) を組み合わせることにより、ネットワーク全体にわたる障害の恐れがある環境において、より高い信頼度でシステムフェイルオーバーを実行できます。つまり、「[スプリットブレイン](#)」の発生リスクを大幅に軽減しながら、ローカルサイトのフェイルオーバーとWAN 越しのノードへのフェイルオーバーを実行することができます。このパッケージでは、多数決ベースの quorum check を使用して 3 ノード以上のクラスタを制御できます。追加の quorum ロジックは、witness サポートパッケージをインストールした場合のみ有効になります。

1 台以上の witness サーバを使用すると、通信障害後にリソースを起動する前に、障害ノードのステータスについて他のノードから「セカンドオピニオン」を取得できます。witness サーバは、クラスタを構成するサーバを判断する調停役として機能する追加的なサーバです。フェイルオーバー先となることができるノードは、witness サーバが障害となったノードのステータスに関して同じ意見である場合のみ、リソース起動が許可されます。これにより、ノード間で発生する単純な通信障害から発生するフェイルオーバーを回避し、全体のアクセスや、パフォーマンス、サービス中のノードに影響を与えないようにします。実際の運用では、最初に行われたとき、witness ノードを含め、クラスタ内の他のすべてのノードに問い合わせをします。

パッケージの要件

すでに説明した要件に加えて、このパッケージの要件として、ライセンス認証された標準の LifeKeeper Core が witness server として機能するサーバにインストールされている必要があります。注記: コミュニケーションパスが正しく設定されている限り、複数のクラスタが単一の quorum/witness server を共有できます (詳細については、下の[共有 witness トポロジーのための追加設定](#)を参照してください)。

quorum/witness モードのクラスタに参加するすべてのノード (witness 専用のノードを含む) には、Quorum/Witness Server Support Package for LifeKeeper をインストールする必要があります。tcp_remote quorum モードを使用する場合は、/etc/default/LifeKeeper 内の QUORUM_HOSTS に設定したホストには Quorum/Witness Server Support Package for LifeKeeper をインストールする必要はありません。

パッケージのインストールと設定

Quorum/Witness Server Support Package for LifeKeeper は、quorum/witness モードのクラスタ内の各サーバ (witness 専用のサーバを含む) にインストールする必要があります。witness ノードに必要な唯一の設定は、[適切なコミュニケーションパスを作成すること](#)です。

witness サーバを追加する一般的なプロセスには以下の手順が含まれます。

- witness ノード用のサーバをセットアップし、他のノードとのネットワーク通信ができることを確認します。
- witness ノード上に LifeKeeper Core をインストールし、適切にライセンス認証/アクティベーションを行います。
- クラスタ内のすべてのノードに quorum/witness サポートパッケージをインストールします。
- witness ノードを含め、すべてのノード間でコミュニケーションパスを作成します。

上記の手順が完了すると、クラスタは quorum/witness モードで動作するようになり、フェイルオーバーが許可される前に、witness ノードを含む他のノードにフェイルオーバーの確認が行われます。パッケージインストール後のデフォルト設定では、多数決ベースの quorum check および witness check が有効になっています。

注記: Quorum (クラスタ構成に必要な最小メンバー数) が多数決ベースのため、クラスタを構成するノードの台数を奇数にすることを推奨します。

詳細な設定オプションについては、下の「設定可能なコンポーネント」セクションを参照してください。

注記: witness パッケージをインストールしたノードであれば、witness 機能に参加できます。witness 専用ノードとは、互換性のある LifeKeeper Core と witness パッケージがインストールされていて、保護対象のリソースを持たないノードのことを単に指しています。

設定可能なコンポーネント

quorum/witness パッケージでは、quorum と witness という2つのモードを設定できます。デフォルトでは、quorum/witness パッケージをインストールすると、quorum と witness の両方のモードが有効になります。これは、witness 機能を必要とする大部分の環境に適した動作です。

これらのモードの動作は、/etc/default/LifeKeeper 設定ファイルでカスタマイズすることが可能です。quorum と witness のモードを個別に調整することもできます。パッケージがインストールされると設定ファイルにはデフォルト設定が書き込まれ、*majority* がデフォルトの quorum モードに、*remote_verify* がデフォルトの witness モードになります。以下はその例です。

```
QUORUM_MODE=majority
WITNESS_MODE=remote_verify
```

注記: クラスタの各ノードでまったく異なる quorum/witness 設定をすることはできますが、予想外の状況や診断の困難な状況になることを避けるため、すべてのノードで同じ設定にすることを推奨します。

使用可能な quorum モード

quorum チェックモードとして次の3種類のモードが用意されています。これらは、`/etc/default/LifeKeeper` の `QUORUM_MODE` 設定を使用して設定できます。`majority` (デフォルト)、`tcp_remote`、`none/off`。以下に各モードを説明します。

majority

デフォルトの `majority` 設定では、チェック時に可視/生存している LifeKeeper ノードの数に基づいて Quorum が決定されます。このチェックは、単純な多数決方式です。全ノード数の過半数を見ることができるノードは Quorum に属します。

tcp_remote

`tcp_remote` quorum モードは、以下の点を除いて `majority` モードと共通です。

- 問い合わせを受けるサーバは、クラスタとそのコミュニケーションパスから独立して設定する (これらのサーバに LifeKeeper をインストールする必要はありません)。
- サーバへの確認は、単に指定ポート上の TCP/IP サービスに接続できるかどうかによって行われる。

このモードでは、TCP のタイムアウト設定 (`QUORUM_TIMEOUT_SECS`) と問い合わせ先のホスト (`QUORUM_HOSTS`) を `/etc/default/LifeKeeper` に追加する必要があるため、追加設定が必要です。`tcp_remote` の設定例は以下の通りです。

```
QUORUM_MODE=tcp_remote
```

```
# comm_up/down および lcm_avail (その他の場合もある) の
# イベントハンドラで実行する quorum 検証のスタイル。
# とることができる値:
# none/off: 何も実行せずチェックを省略し、すべてが良好であると仮定する。
# majority: このノード、および到達可能なノードが
# クラスタ内の半分を超えるノードを持つことを検証する。
# tcp_remote: このノードが、TCP/IP 経由で半分を超える
# QUORUM_HOSTS に到達可能であることを検証する。
```

```
QUORUM_HOSTS=myhost:80,router1:443,router2:22
```

```
# QUORUM_MODE が tcp_remote の場合、これは、host:port 値の
# カンマ区切りリストにする必要がある (例: myhost:80,router1:443,router2:22)。
# QUORUM_MODE が他の値の場合は、カンマ区切りリストにする必要はない。QUORUM_TIMEOUT_SECS=20
# TCP/IP witness 接続を完了するまでの許容時間。
# この時間内に完了しなかった接続は、
# 失敗 / 使用不可として扱われる。
# これは、QUORUM_MODE が tcp_remote の場合にのみ適用される。
```

```
WITNESS_MODE=remote_verify
```

```
# これは、off/none、remote_verify のいずれかの値に設定できる。remote_verify
# モードでは、Core のイベントハンドラ (comm_down) が、
```

使用可能な witness モード

- # 他の認識可能なノードもシステムが停止していると
- # 判断するかかどうかを調べて、システムの停止をダブルチェックする。

QUORUM_LOSS_ACTION=fastboot

- # これは、osu、fastkill、fastboot のいずれかの値に設定できる。
- # fastboot は、quorum の喪失を検出した場合、
- # 即座にシステムを再起動する。
- # fastkill は、quorum の喪失を検出した場合、
- # 即座にシステムを停止するか、電源をオフにする。
- # osu は、In Service のリソースを Out of Service にする。
- # 注記：この動作では、ディスクの同期やファイルシステムのアンマウントは実行されない。

QUORUM_DEBUG=

- # Quorum モジュールからのデバッグメッセージを有効にするには、
- # true/on/1 に設定する。

HIDE_GUI_SYS_LIST=true

注記: このモードは本質的に柔軟性と複雑さを備えるため、使用するには、LifeKeeper および関連する特定のネットワーク/クラスタ設定の両方に対する十分な理解と注意が必要です。

none/off

このモードでは、すべての quorum check が無効になっています。この設定では、クラスタの実際の状態にかかわらず、あたかもそのノードが常に Quorum を持っているように quorum check が動作します。

使用可能な witness モード

2 種類の witness モードが用意されており、`/etc/default/LifeKeeper` の `WITNESS_MODE` 設定を使用して設定できます。`remote_verify` および `none/off`。以下に各モードを説明します。

remote_verify

このデフォルトモードでは、witness check によってノードのステータスを確認します。通常、この確認はノード障害が疑われるときに行われます。このモードでは、各ノードはクラスタ内の他のすべての可視ノードに対して障害ノードのステータスに関する意見を求めることにより、疎通を二重にチェックします。

none/off

このモードでは、witness check が無効になっています。通信障害の場合は、あたかも witness 機能がインストールされていないかのような論理で動作します。

注記: リソースを持たずに quorum/witness 専用ノードとして動作するサーバは witness check を実行する必要がないため、そのようなサーバでは、witness モードを `none/off` に設定する必要があります。

Quorum を喪失したときに利用可能なアクション

witness パッケージでは、quorum を喪失したときにシステムがどのように応答すべきかについて 3 種類のオプションを提供しています。“fastboot”、“fastkill” および “osu” です。これらのオプションは、

/etc/default/LifeKeeper 内の QUORUM_LOSS_ACTION 設定によって選択できます。3つのオプションはすべてそのシステムのリソースをサービス休止状態にしますが、それぞれ異なる動作をします。quorum パッケージがインストールされている場合のデフォルトオプションは fastboot です。以下に各オプションを説明します。

fastboot

fastboot オプションを選択している場合、(通信ができないことにより) Quorum の喪失が検出されるとシステムは直ちにリブートします。これは過激な方法ですが、確実にシステムを外部のリソースから素早く切り離すことができます。ストレージレベルのレプリケーションなど、多くの場合にリソースをこのように即座にリリースすることが望まれます。

このオプションには、以下の2つの重要な注意点があります。

1. システムは、シャットダウン手順を最初に行うことなく直ちにハードリブートを実行します。(ディスクの同期などの)タスクは一切実行されません。
2. システムは、ストレージとのネゴシエーションやリソースへのアクセスなどを含む通常の起動ルーチンを実行しながら復帰します。

fastkill

fastkill オプションは、fastboot オプションに非常に似ていますが、システムは Quorum を喪失したときにハードリブートするのではなく、即座に停止します。fastboot オプションと同様に(ディスクの同期などの)タスクは一切実行されません。システムは手動でリブートする必要があります。その後システムは、ストレージとのネゴシエーションやリソースへのアクセスなどを含む通常の起動ルーチンを実行しながら復帰します。

osu

osu オプションは、最も穏健なオプションです。Quorum を喪失したシステムはそのまま稼働しますが、システム上のリソースはサービス休止状態にされます。一部のクラスタ構成ではこの方法で十分ですが、他のクラスタ構成では保護能力不足だったり応答が遅すぎる場合があります。

共有 witness トポロジーのための追加設定

quorum/witness サーバを複数のクラスタで共有する場合、個々のクラスタの管理を簡素化するように設定することができます。標準的な操作では、LifeKeeper GUI を使用して最初のノードに接続しようとすると、LifeKeeper GUI はすべてのクラスタノードとの接続を試みます。つまり、クラスタ内の各システムから見えるすべてのシステムに接続します。共有 witness サーバはすべてのクラスタに接続されているため、GUI は witness ノードから見えるすべてのクラスタ内のすべてのシステムに接続することになります。

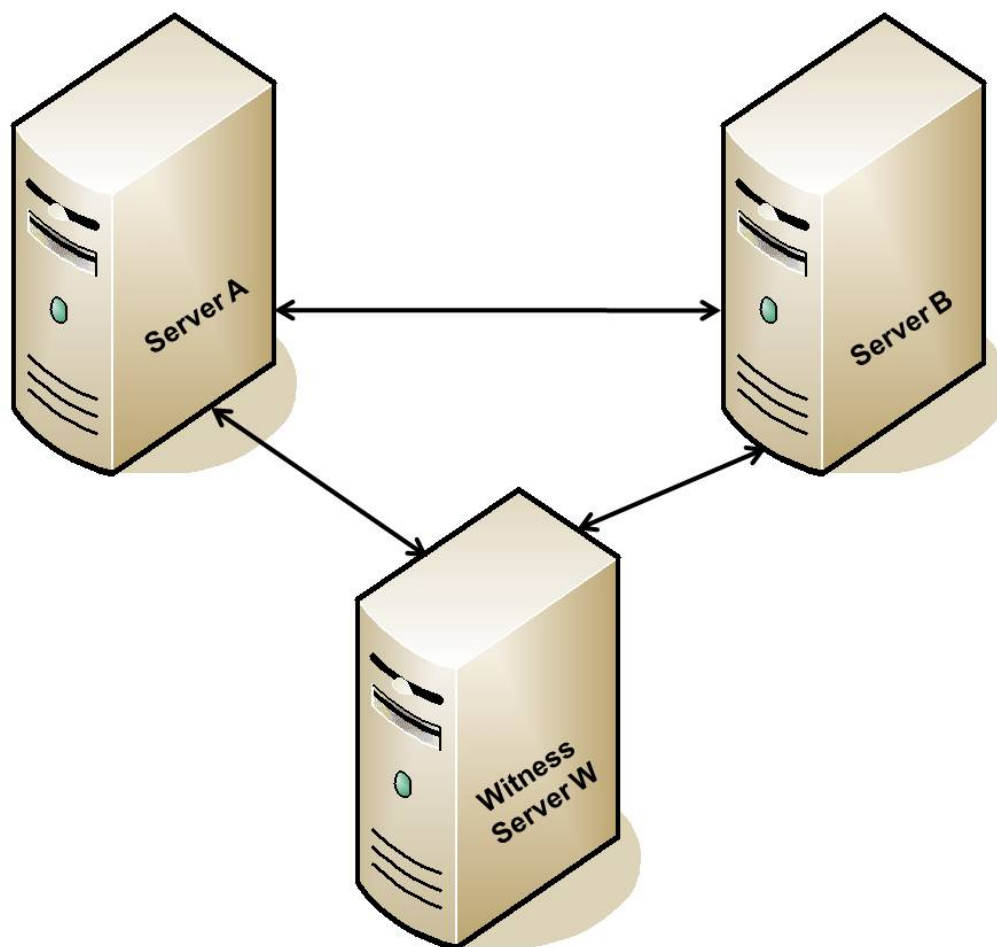
この状況を回避するには、**すべての共有 witness サーバで HIDE_GUI_SYS_LIST 設定パラメータを「true」に設定する必要があります**。この設定によって witness サーバから見えるサーバは実質的に不可視になり、GUI は最初に接続したサーバに関連付けられたクラスタ内のサーバにのみ接続するようになります。**注記:** この設定は witness サーバにのみ設定してください。

GUI は、最初に接続したサーバに関連付けられたクラスタ内のサーバにのみ接続するため、その最初のサーバが witness サーバで、かつ HIDE_GUI_SYS_LIST が「true」に設定されている場合、GUI はコミュニケーションパスが確立している他のサーバに自動的に接続しません。この現象は LifeKeeper GUI の典型的な動作ではないため、ネットワークまたは他の設定に問題があるとインストーラが間違っていると判断する可能性があります。この設定をした witness サーバ上で LifeKeeper GUI を使用する場合は、クラスタ内の他のいずれかのノードに手動で接続すると、クラスタの残りのノードが正しく GUI に表示されます。

注記: すべてのクラスタ内のすべてのシステムで witness check が実行されるのを防ぐには、共有する quorum/witness 専用ノードで `witness_mode` を常に `none/off` に設定してください。

2 ノードクラスタに witness ノードを追加する

以下は Quorum/Witness Server Support Package for LifeKeeper を利用する 2 ノードクラスタに 3 番目のノードとなる witness ノードを追加する場合の例です。



witness ノードを持つ単純な 2 ノードクラスタ

サーバAとサーバBは、LifeKeeper Coreを使用したセットアップがすでに完了し、サーバAで作成されたリソース階層がサーバBに拡張されています(サーバWは、拡張されたリソース階層を持っていません)。以下の手順を使用して3番目のノードをwitnessノードとして追加します。

期待される動作 (デフォルトモードを仮定)

1. witness 用のノードをセットアップし、他の2ノードとのネットワーク通信ができることを確認します。
2. witness ノード上に LifeKeeper Core をインストールし、適切にライセンス認証/アクティベーションを行います。
3. 3ノードすべてに Quorum/Witness Server Support Package をインストールします。
4. 3ノードの間すべてにコミュニケーションパスを作成します。
5. `/etc/default/LifeKeeper` に目的の quorum チェックモード (`majority`、`tcp_remote`、`none/off`) を設定します (この例では `majority` を選択)。これらのモードの説明については、[使用可能な quorum モード](#)を参照してください。
6. `/etc/default/LifeKeeper` に、目的の witness モード (`remote_verify`、`none/off`) を設定します。これらのモードの説明については、[使用可能な witness モード](#)を参照してください。

期待される動作 (デフォルトモードを仮定)

シナリオ 1

サーバ A とサーバ B との間の通信に障害が発生

サーバ A とサーバ B の間の通信に障害が発生した場合、以下のように動作します。

- サーバ A と B は、通信障害イベントの処理を開始します。ただし、全く同時とは限りません。
- 両方のサーバは簡単な quorum check を実行し、両方共自身が多数派に属すると判断します (A と B の両方から W が見えているため、既知の3ノードのうちの2ノード側にいると判断します)。
- 各サーバは、まだ通信可能な他方のノードに対し、自ノードと通信できなくなったサーバの状態について問い合わせます。このシナリオでは、サーバ A が B のステータスについて W に問い合わせ、サーバ B が A のステータスについて W に問い合わせることになります。
- サーバ A と B は共に witness サーバへの問い合わせによって他方のサーバがまだ生存していると判断し、フェイルオーバー処理は何も発生しません。リソースはサービス状態のままになります。

シナリオ 2

サーバ A と W との間の通信に障害が発生

witness パッケージがインストールされていると、すべてのノードが witness ノードとして動作することが可能であり、実際にそのように動作するため、このシナリオは前のシナリオと同じになります。この場合、サーバ A と witness サーバ W は共にサーバ B への問い合わせによって他方のサーバがまだ生存していると判断します。

シナリオ 3

サーバ A と他の全ノードとの間の通信に障害が発生 (A に障害が発生)

この場合、サーバ B は以下の動作をします。

- サーバAとの通信障害イベントの処理を開始します。
- witness サーバWとまだ通信が可能であり、したがって Quorum を持っていると判断します。
- サーバAが見えないことをサーバWに確認した後、通常のフェイルオーバー動作を開始します。
- これにより保護対象のリソースはサーバB上でサービス中になります。

B がソースとして動作している状態で、サーバAの通信が回復

前のシナリオの状態から、サーバAが通信を再開したとします。サーバBは、comm_up イベントを処理し、Quorum を持っている (3 ノードすべてが見える) ことと、サービス中のリソースを持っていることを判断します。サーバAは、comm_up イベントを処理し、自身も Quorum を持っていることと、リソースが別の場所でサービス中であることを判断します。この時点では、サーバAはリソースを起動しません。

B がソースとして動作している状態で、サーバAに電源が入れられて他のノードと通信可能

この場合、サーバBは前のシナリオと同じように応答しますが、サーバAはlcm_avail イベントを処理します。サーバAは、Quorum を持っていると判断し、この場合は、現在サーバBでサービス中のリソースを起動しないことにより正常に応答します。

B がソースとして動作している状態で、サーバAに電源が入れられて他のノードと通信不能

この場合、サーバAはlcm_avail イベントを処理し、サーバBとWはサーバAと通信できないので何もしません。サーバAは、3 ノードのうちの1 ノードとしか通信できないため Quorum を持っていないと判断します。Quorum を持たない場合、サーバAはリソースを起動しません。

シナリオ 4

サーバAと他の全ノードとの間の通信に障害が発生 (Aのネットワークに障害が発生しているが、Aは稼働中)

この場合、サーバBは以下の動作をします。

- サーバAとの通信障害イベントの処理を開始します。
- witness サーバWとまだ通信が可能であり、したがって Quorum を持っていると判断します。
- サーバAが見えないことをサーバWに確認した後、通常のフェイルオーバー動作を開始します。
- これにより保護対象のリソースはサーバB上でサービス中になります。

また、この場合、サーバAは以下の動作をします。

- サーバBとの通信障害イベントの処理を開始します。
- サーバBとも witness サーバWとも通信できないため、Quorum を持っていないと判断します。
- 直ちにリポートします (「fastboot」がデフォルトの動作のため、ハードリポートされます)。

STONITH

STONITH (Shoot the Other Node in the Head) は、クラスタ内のノードの電源をリモートから切断するフェンシング方式です。LifeKeeperのSTONITH機能は、外部電源スイッチコントロール、IPMI対応マザーボード、および

ハイパーバイザーが提供する電源機能を利用してクラスタ内の他のノードの電源を切断します。

STONITH で IPMI を使用する

IPMI (Intelligent Platform Management Interface) は、コンピュータシステムにアクセスする共通インターフェースのセットを提供します。IPMI を使用すると、システムの状態を監視してシステムを管理できます。IPMI を STONITH で使用すると、故障と思われるクラスタノードの電源スイッチをクラスタソフトウェアが制御することにより、シリアル接続またはネットワーク接続を介してノードの電源切断やリブートができるため、故障ノードが共有データにアクセスしたりデータを破損するのを確実に防ぎます。

パッケージの要件

- IPMI ツールのパッケージ (ipmitool-1.8.11-6.el6.x86_64.rpm など)

VMware vSphere 環境での STONITH

vCLI (vSphere Command-Line Interface) は、ESXi ホストと仮想マシンを含む仮想インフラストラクチャを管理するための VMware でサポートされているコマンドラインインターフェースです。vCLI コマンドがお使いの環境のニーズに合致する場合は、VMware の仮想マシン間での LifeKeeper STONITH の実装に適用することができます。

パッケージの要件

STONITH サーバ

- VMware vSphere SDK パッケージまたは VMware vSphere CLI (vSphere CLI は、vSphere SDK と同じインストールパッケージに含まれています)

監視対象仮想マシン

- VMware Tools

インストールと設定

LifeKeeper をインストールし、クラスタ内の各ノードでコミュニケーションパスを設定した後、STONITH をインストールおよび設定します。

1. 次のコマンドを実行して LifeKeeper STONITH スクリプトをインストールします。

```
/opt/LifeKeeper/samples/STONITH/stonith-install
```

2. (*IPMI を利用する場合のみ) BIOS または ipmitool コマンドを使用して、以下の BMC (Baseboard Management Controller) 変数を設定します。
 - 静的 IP アドレスの使用
 - IP アドレス
 - サブネットマスク

- ユーザ名
- パスワード
- ユーザに管理者権限を追加
- ユーザのネットワークアクセスを有効化

ipmitool コマンドの使用例を示します。

(詳細については、ipmitool のマニュアルページを参照してください。)

```
# ipmitool lan set 1 ipsrc static
# ipmitool lan set 1 ipaddr 192.168.0.1
# ipmitool lan set 1 netmask 255.0.0.0
# ipmitool user set name 1 root
# ipmitool user set password 1 secret
# ipmitool user priv 1 4
# ipmitool user enable 1
```

3. 設定ファイルを編集します。

設定ファイルを編集し、STONITH を有効にして電源を切断するコマンドラインを追加します。**注記:** フェンシンググループ (2 台のマシン間で、通信は喪失しているものお互いをまだ STONITH できる場合に、お互いに電源オフとリブートをし続ける状態) を回避するため、リブートではなく電源オフを推奨します。

/opt/LifeKeeper/config/stonith.conf

```
# LifeKeeper STONITH configuration
#
# Each system in the cluster is listed below. To enable STONITH for a
# given system,
# remove the '#' on that line and insert the STONITH command line to power off
# that system.
# Example1: ipmi command
# node-1 ipmitool -l lanplus -H 10.0.0.1 -U root -P secret power off
# Example2: vCLI-esxcli command
# node-2 esxcli --server=10.0.0.1 --username=root --password=secret vms vm kill --
type='hard' --world-id=1234567
# Example3: vCLI-vmware_cmd command
# node-3 vmware-cmd -H 10.0.0.1 -U root -P secret <vm_id> stop hard
minute-maid ipmitool -l lanplus -H 192.168.0.1 -U root -P secret power off
kool-aid ipmitool -l lanplus -H 192.168.0.2 -U root -P secret power off
vm1 esxcli --server=10.0.0.1 --username=root --password=secret vms vm kill --
type='hard' --world-id=1234567
vm2 vmware-cmd -H 10.0.0.1 -U root -P secret <vm_id> stop hard
```

<vm_id>

<vm_id>

vSphere CLI コマンドは、vSphere SDK for Perl の上で実行されます。vm_id > は、VM の識別子として使用されています。この変数によって、設定対象の VM 用の設定ファイルを指定します。

設定ファイルのパスを調べるには、以下の手順を実行してください。

1. 次のコマンドを実行します。

```
vmware-cmd -H <vmware host> -l
```

2. 上記のコマンドによって VMware ホストのリストが表示されます。

vmware-cmd -l の出力例を以下に示します (3 台の VM を表示)。

```
/vmfs/volumes/4e08c1b9-d741c09c-1d3e-0019b9cb28be/lampserver/lampserver.vmx  
/vmfs/volumes/4e1e1386-0b862fae-a859-0019b9cb28bc/oracle10/oracle.vmx  
/vmfs/volumes/4e08c1b9-d741c09c-1d3e-0019b9cb28be/lampserver02/lampserver02.vmx
```

出力されたリストで設定中の VM を見つけます。

3. パス名を <vm_id> 変数にペーストします。上記の例では以下の通りになります。

```
vmware-cmd -H 10.0.0.1 -U root -P secret /vmfs/volumes/4e08c1b9-d741c09c-1d3e-  
0019b9cb28be/lampserver/lampserver.vmx stop hard
```

注記: VMware コマンドの詳細については、引数なしで vmware-cmd を実行するとすべてのオプションに関するヘルプページが表示されます。

期待される動作

LifeKeeper がノードとの通信障害を検出すると、そのノードの電源が切断され、フェイルオーバーが発生します。問題を修復した後に、手動でそのノードの電源を入れる必要があります。

Watchdog

Watchdog は、サーバが正常に動作しない場合に、問題の発生を予防する修正処置 (リポート) を確実に実行できるようにサーバを監視する方法です。Watchdog は、特別な Watchdog ハードウェアを使用して実装する場合と、ソフトウェアのみのオプションを使用して実装場合があります。

(注記: この構成は、Red Hat Enterprise Linux Versions 5 および 6 でのみ検証されています。他のオペレーティングシステムでは検証されていないため、現時点ではサポートされません。)

コンポーネント

- Watchdog タイマのソフトウェアドライバまたは外部ハードウェアコンポーネント
- Watchdog デモン - 該当する Linux ディストリビューションを通じて rpm が入手可能

設定

- LifeKeeper Core デーモン - LifeKeeper のインストールに付随
- ヘルスチェックスクリプト - LifeKeeper の監視スクリプト



LifeKeeper と Watchdog の相互運用性

次のセクションを注意深く読んでください。デーモンは、エラーからリカバリするように設計されており、注意深く設定しないとデーモンはシステムをリセットします。インストールおよび設定を行う前に慎重に計画してください。このセクションの目的は、Watchdog についての説明や設定をすることではありません。ここでは、Watchdog 構成での LifeKeeper との相互運用性についての説明や設定のみを行います。

設定

以下の手順は、root ユーザ権限を持つ管理者が行う必要があります。管理者は、Watchdog のリスクおよび問題についてすでに熟知しているものとします。

ヘルスチェックスクリプト (LifeKeeper 監視スクリプト) は、LifeKeeper の設定と Watchdog の設定 (/opt/LifeKeeper/samples/watchdog/LifeKeeper-watchdog) を関連付けるコンポーネントです。このスクリプトは、LifeKeeper の完全な監視を提供するものであり、一切の修正は不要です。

1. 以前に Watchdog を設定していた場合は、次のコマンドを入力して Watchdog を停止します。そうでない場合は、手順 2 に進みます。

```
/etc/rc.d/init.d/watchdog stop
```

Watchdog の停止を示す以下の確認メッセージが表示されるはずです。

```
Stopping watchdog:[OK]
```

アンインストール

2. Watchdog ソフトウェアのインストールで供給される Watchdog 設定ファイル (`/etc/watchdog.conf`) を編集します。

- `test-binary` を次のように修正します。

```
test-binary = /opt/LifeKeeper/samples/watchdog/LifeKeeper-watchdog
```

- `test-timeout` を次のように修正します。

```
test-timeout = 5
```

- `interval` を次のように修正します。

```
interval = 7
```

`interval` は、LifeKeeper のコミュニケーションパスのタイムアウト (15 秒) よりも小さい値にしてください。約半分の値が妥当です。

3. LifeKeeper が起動していることを確認します。まだの場合は、[LifeKeeper の起動](#) トピックを参照してください。
4. 次のコマンドを入力して Watchdog を起動します。

```
/etc/rc.d/init.d/watchdog start
```

Watchdog の起動を示す以下の確認メッセージが表示されるはずですが、

```
Starting watchdog: [OK]
```

5. 今後の再起動の際に Watchdog を自動的に起動させるには、次のコマンドを入力します。

```
chkconfig --levels 35 watchdog on
```

注記: Watchdog を設定すると、予想外のリポートがときどき発生する可能性があります。これは、Watchdog の仕組みから来る一般的な性質です。正常に応答しないプロセスがあると、Watchdog 機能は LifeKeeper (またはオペレーティングシステム) がハングしていると判断し、(警告なしに) システムをリポートします。

アンインストール

LifeKeeper をアンインストールする場合は、慎重に行ってください。以下に列記の通り、上記の手順を逆の順で実行します。

警告: LifeKeeper を構成する RPM パッケージを削除する方法で LifeKeeper をアンインストールする場合は、**先に Watchdog を停止してください**。上記の手順 2 では、LifeKeeper の Watchdog スクリプトを呼び出すように Watchdog 設定ファイルを修正しています。したがって、先に Watchdog を停止しておかないと、存在しないスクリプトを呼び出すこととなります。リポートを実行するこのスクリプトが見つからない場合は、エラーが発生します。この状態は Watchdog を停止するまで継続します。

1. 次のコマンドを入力して Watchdog を停止します。

```
/etc/rc.d/init.d/watchdog stop
```

Watchdog の停止を示す以下の確認メッセージが表示されるはずですが、

```
Stopping watchdog: [OK]
```

2. Watchdog ソフトウェアのインストールで供給される Watchdog 設定ファイル(/etc/watchdog.conf) を編集します。

- test-binary および interval の両エントリをコメントアウトします (各行の先頭に # を追加します)。

```
#test-binary =
#interval =
```

(注記: interval が他の機能によって以前から使用されていた場合は、そのままにしておくこともできます)

3. LifeKeeper をアンインストールします。 [LifeKeeper の削除](#) トピックを参照してください。
4. これで Watchdog を起動し直すことができます。LifeKeeper のみが Watchdog を使用していた場合は、次のコマンドを入力すると Watchdog を永続的に無効にできます。

```
chkconfig --levels 35 watchdog off
```

リソースポリシー管理

概要

SIOS Protection Suite for Linux のリソースポリシー管理では、リソースのローカルリカバリとフェイルオーバーの動作管理機能が提供されます。リソースポリシーは、**lkpolicy** コマンドラインツール (CLI) を使用して管理できます。

SIOS Protection Suite

SIOS Protection Suite には、個々のアプリケーションおよび関連し合うアプリケーションのグループを監視する機能があり、定期的にローカルリカバリを実行したり、保護下のアプリケーションに障害が発生したときに通知することができます。関連し合うアプリケーションの例としては、主アプリケーションが下位のストレージまたはネットワークリソースに依存する階層などがあります。アプリケーションまたはリソースに障害が発生した場合のデフォルトの動作は以下の通りです。

1. **ローカルリカバリ**: 最初に、リソースまたはアプリケーションのローカルでリカバリを試みます。このときは、外部の介入なしにローカルサーバ上でリソースまたはアプリケーションをリストアしようとします。ローカルリカバリが成功した場合、SIOS Protection Suite は追加のアクションを実行しません。
2. **フェイルオーバー**: 次に、ローカルリカバリでリソースまたはアプリケーションのリストアに失敗した (またはリソースを監視するリカバリキットがローカルリカバリをサポートしていない) 場合、**フェイルオーバー**は開始されます。フェイルオーバー処理では、クラスタ内の別のサーバ上で該当アプリケーション (および依存するすべてのリソース) を起動しようと試みます。

リカバリ動作の詳細については、[SIOS Protection Suite 障害検出とリカバリのシナリオ](#)を参照してください。

ポリシーによるカスタム動作およびメンテナンスモード動作

SIOS Protection Suite Version 7.5 以降では、デフォルトのリカバリ動作を変更する追加ポリシーを設定する機能をサポートします。リソース単位 またはサーバ単位で、4 つのポリシーが設定可能です (リソース単位のポリシーに関する注意については下のセクションを参照してください)。 **サーバレベルでポリシーを変更する方法を推奨します。**

利用可能なポリシーは以下の通りです。

標準ポリシー

- **Failover** - このポリシー設定を使用すると、リソースフェイルオーバーを有効 / 無効にできます。(S: リザベーションが適切に処理されるには、フェイルオーバーは、個々の SCSI リソースで無効にすることはできません。)
- **LocalRecovery** - SIOS Protection Suite は、デフォルトでは、フェイルオーバーを実行する前に、個々のリソースまたは保護対象アプリケーション全体を再起動することにより、保護対象リソースのリカバリを試みます。このポリシー設定を使用すると、ローカルリカバ리를有効 / 無効にできます。
- **TemporalRecovery** - 通常、SIOS Protection Suite は、障害リソースのローカルリカバ리를実行します。ローカルリカバリに失敗すると、SIOS Protection Suite は、リソース階層を別ノードにフェイルオーバーします。ローカルリカバリに成功した場合は、フェイルオーバーは実行されません。

ローカルリカバリに成功した場合でも、サーバの何らかの異常によって短時間の間にローカルリカバリが再試行される場合があります。結果として何度も連続してローカルリカバリが試行されることになります。これが発生すると、問題のアプリケーションは可用性が悪化します。

この反復的なローカルリカバリ / 障害サイクルを回避するために、時間的リカバリポリシーを設定できます。時間的リカバリポリシーを使用すると、管理者は指定した時間内に試行するローカルリカバリの回数を (成功かどうかにかかわらず) 制限することができます。

例: リソースが試行するローカルリカバリの回数を 30 分間で 3 回に限定するポリシー定義をユーザが設定した場合、30 分以内に 3 回目のローカルリカバリが試行されると、SIOS Protection Suite はフェイルオーバーを実行します。

定義した時間的リカバリポリシーは有効または無効にできます。時間的リカバリポリシーが無効の場合、時間的リカバリ処理は継続して実行され、ポリシーが適用されるはずの時間に通知がログに表示されますが、実際のアクションは実行されません。

注記: 時間的リカバリポリシーを設定した状態で、フェイルオーバーとローカルリカバリの一方または両方を無効にすることは可能です。フェイルオーバーまたはローカルリカバ리를無効にした場合に、時間的リカバリポリシーは実行されることがないため、この状態は非論理的です。

メタポリシー

「メタ」ポリシーは、他の複数のポリシーに影響を与える可能性があるポリシーです。通常、これらのポリシーは、標準ポリシーであれば複数個の設定が必要になるような特定のシステム動作を実現するためのショートカットとして使用します。

- **NotificationOnly** - このモードでは、管理者は SIOS Protection Suite を「監視専用」状態にすることができます。1 つのリソース (または、サーバ単位の場合はすべてのリソース) のローカルリカバリおよびフェイルオーバーの両方が影響を受けます。障害が検知されると、ユーザインターフェースには **Failure** 状態が表示されます。ただし、リカバリもフェイルオーバーも実行されません。**注記:** 管理者は、障害の原因となった問題を手動で修正し、障害が起きたリソースを復帰させて通常の SIOS Protection Suite の運用を継続する必要があります。

リソースレベルのポリシーに関する重要な考慮事項

リソースレベルのポリシーとは、リソース階層全体またはサーバレベルのポリシーとは異なり、特定のリソースにのみ適用されるポリシーです。

例:

アプリケーション

- IP
- file system

上記のリソース階層では、アプリケーションは IP とファイルシステムの両方に依存しています。ポリシーは、特定のリソースのローカルリカバリまたはフェイルオーバーが無効にするように設定できます。これは、例えば、IP リソースのローカルリカバリが失敗し、IP リソースのフェイルオーバーが無効に設定されていた場合、IP リソースはフェイルオーバーを実行せず、他のリソースのフェイルオーバーも発生させないことを意味します。ただし、ファイルシステムリソースのローカルリカバリが失敗し、ファイルシステムリソースのポリシーのフェイルオーバーが無効化されていない場合、階層全体がフェイルオーバーを実行します。

注記: 重要事項として、リソースレベルのポリシーは設定対象の特定のリソースにのみ適用されることに注意してください。

上記は単純な例です。複雑な階層を構成することもできるため、リソースレベルのポリシーを設定するときは注意してください。

lkpolicy ツール

lkpolicy ツールは、SIOS Protection Suite for Linux が稼働するサーバのポリシーを管理 (参照、設定、削除) するためのコマンドラインツールです。lkpolicy は、ポリシーの設定および修正、ポリシーの削除、利用可能なポリシーと現在の設定値の表示をサポートします。さらに、設定したポリシーは、有効または無効に設定できるため、リカバリ動作に影響を与えながらリソース / サーバ設定を保持できます。

全体的な使用方法は次の通りです。

```
lkpolicy [--list-policies | --get-policies | --set-policy | --remove-policy] <name value pair data...>
```

<name value pair data...>は、運用方法および対象のポリシーによって異なります (特にポリシーを設定する場合)。例: 有効 / 無効タイプのポリシーのほとんどでは、必要なのは [-on] または [-off] のスイッチのみですが、時間的ポリシーの場合は、しきい値を設定するための値も必要です。

lkpolicy の使用方法の例

ローカルおよびリモートサーバとの認証

lkpolicy ツールは、サーバが公開する API を通じて SIOS Protection Suite サーバと通信します。この API は、lkpolicy ツールなどのクライアントに対して認証を要求します。lkpolicy ツールで SIOS Protection Suite サーバに最初にアクセスしようとしたときに、そのサーバに対する認証情報がまだ保存されていない場合、ユーザは認証情報を求められます。認証情報はユーザ名とパスワードの形式であり、さらに以下の条件があります。

ポリシーのリスト表示

1. クライアントには SIOS Protection Suite の管理者権限が必要です。したがって、そのユーザ名は、(PAM による)オペレーティングシステムの認証設定によって *lkadmin* グループに属する必要があります。必ずしも **root** で実行する**必要はありません**が、root ユーザはデフォルトで適切なグループに属しているため、root を使用することもできます。
2. 認証情報は認証情報ストアに保存されるため、ツールを使用してこのサーバにアクセスするたびに手動で認証情報を入力する必要はありません。

認証情報ストアと credstore ユーティリティによる管理の詳細については、[SIOS Protection Suite の認証情報の設定](#)を参照してください。

lkpolicy によるセッションの例は以下ようになります。

```
[root@thor49 ~]# lkpolicy -l -d v6test4
Please enter your credentials for the system 'v6test4'.
Username: root
Password:
Confirm password:
Failover
LocalRecovery
TemporalRecovery
NotificationOnly
[root@thor49 ~]# lkpolicy -l -d v6test4
Failover
LocalRecovery
TemporalRecovery
NotificationOnly
[root@thor49 ~]#
```

ポリシーのリスト表示

```
lkpolicy --list-policy-types
```

現在のポリシーの表示

```
lkpolicy --get-policies
```

```
lkpolicy --get-policies tag=\*
```

```
lkpolicy --get-policies --verbose tag=mysql\* # all resources starting with mysql
```

```
lkpolicy --get-policies tag=mytagonly
```

ポリシーの設定

```
lkpolicy --set-policy Failover --off
```

```
lkpolicy --set-policy Failover --on tag=myresource
```

```
lkpolicy --set-policy Failover --on tag=\*
```

```
lkpolicy --set-policy LocalRecovery --off tag=myresource
```

```
lkpolicy --set-policy NotificationOnly --on
```

```
lkpolicy --set-policy TemporalRecovery --on recoverylimit=5 period=15
```

```
lkpolicy --set-policy TemporalRecovery --on --force recoverylimit=5 period=10
```

ポリシーの削除

```
lkpolicy --remove-policy Failover tag=steve
```

注記: *NotificationOnly* はポリシーのエイリアスです。*NotificationOnly* を有効にすることは、対応する *LocalRecovery* および *Failover* ポリシーを無効にすることと等価です。

認証情報の設定

他のシステムと通信するための認証情報は、認証情報ストアを使用して管理されています。このストアは、必要に応じて `/opt/LifeKeeper/bin/credstore` ユーティリティで管理できます。このユーティリティを使用すると、サーバアクセスに必要な認証情報をサーバごとに設定、変更、削除することができます。

認証情報の追加または変更

認証情報の追加と変更は同じ方法で実行できます。代表的な例として、サーバー `server.mydomain.com` に対する資格情報を追加または変更する場合は次のようになります。

```
/opt/LifeKeeper/bin/credstore -k server.mydomain.com myuser
```

この例では、`server.mydomain.com` へのアクセスに使用するユーザ名として *myuser* を指定してしています。パスワードを入力 / 確認するプロンプト (*passwd* の様に) が表示されます。

注記: LifeKeeper サーバの認証情報を格納するために使用されるキー名は、`lkpolicy` などのコマンドで使用するホスト名と完全に一致する必要があります。コマンドで使用するホスト名が FQDN であれば、認証情報のキーも FQDN でなければなりません。コマンドで使用するホスト名がショートネームであれば、認証情報のキーもショートネームでなければなりません。

認証情報ストアにデフォルトキーを作成することもできます。サーバキーが存在しない場合にデフォルトの認証情報が認証に使用されます。デフォルトキーを追加、変更するには以下のコマンドを実行してください。

```
/opt/LifeKeeper/bin/credstore -k default myuser
```

ストア内の認証情報のリスト表示

現在格納されている認証情報をリスト表示するには、以下のコマンドを実行します。

```
/opt/LifeKeeper/bin/credstore -l
```

これにより、認証情報ストア内に格納されているキーが表示されます。この場合の「キー」は、認証情報を使用する対象のサーバを示しています(認証情報自体は秘密情報のため、このコマンドが表示するのは、実際の認証情報の内容ではなくキー名のみです)。

サーバの認証情報の削除

特定のサーバに対する認証情報を削除するには、以下のコマンドを実行します。

```
/opt/LifeKeeper/bin/credstore -d -k myserver.mydomain.com
```

この例では、サーバ myserver.mydomain.com に対する認証情報がストアから削除されます。

追加情報

credstore ユーティリティの詳細については、以下のコマンドを実行してください。

```
/opt/LifeKeeper/bin/credstore --man
```

コマンドのマニュアルページがすべて表示されます。

LifeKeeper API

LifeKeeper API を使用すると、LifeKeeper サーバ間の通信を行えるようになります。

重要な注記: 現在、この API は内部使用のみに予約されていますが、将来のリリースでは、ユーザやサードパーティが使用できるように開放される可能性があります。

ネットワーク設定

LifeKeeper の各サーバは、ポート 778 の SSL 接続を使用してこの API を提供します。このポートは、`/etc/default/LifeKeeper` 内の設定変数 `API_SSL_PORT` を使用して変更できます。

認証

LifeKeeper API は認証に PAM を使用します。API へのアクセス権限は、グループ `lkadmin`、`lkoper`、`lkguest` のメンバーであるユーザにのみ付与されます。ユーザに権限を与えるには、システムの PAM 設定に応じて、ローカルシステムファイル (`/etc/passwd` および `/etc/group`) を使用するか、ユーザを LDAP または Active Directory のグループに追加します。

注記: LifeKeeper API は、`lkpasswd` で管理されるユーザデータベースは使用しません。

Chapter 2: LifeKeeper 管理

概要

LifeKeeper は操作時に管理を必要としません。LifeKeeper は、保護されたリソースを監視し、障害が発生した場合に指定されたリカバリアクションを実行するように、自動的に機能します。以下のケースでは LifeKeeper GUI を使用します。

- **リソースおよび階層の定義**。LifeKeeper は次のインターフェースオプションを提供します。
 - LifeKeeper GUI。
 - LifeKeeper コマンドラインインターフェース。
- **リソース監視**。LifeKeeper GUI は、リソースステータス情報および LifeKeeper ログへのアクセスを提供します。
- **手動での処理**。メンテナンスやその他の管理アクションのために、サーバまたは特定のリソースを停止することが必要になる場合があります。LifeKeeper GUI には、特定のリソースを稼働させたり停止させることができるメニュー機能が用意されています。アプリケーションが LifeKeeper の保護下に置かれると、これらの LifeKeeper のインターフェースを介してのみアプリケーションを起動および停止させることができます。LifeKeeper の起動および停止は、コマンドラインを介してのみ行われます。

LifeKeeper の管理、設定、およびメンテナンス操作を実行する詳細な手順については、[GUI の作業](#) および [メンテナンス作業](#) を参照してください。

※ LifeKeeper が提供しているコマンド (実行可能なスクリプトやプログラム) を実行するには、スーパーユーザ権限が必要です。

su コマンドや sudo でスーパーユーザ権限を付与したユーザで、LifeKeeper のコマンドを実行することは可能ですが、SIOS Technology Corp では、root ユーザ以外で LifeKeeper のコマンドのテストはしていません。

エラーの検出および通知

アプリケーション内の問題を検出して通知する機能は、最適な総合的耐障害性ソリューションを構築する上で非常に重要です。すべての個々のアプリケーションは、障害発生メカニズムと形式によって異なるため、一般的なメカニズムを示すことはできません。ただし、一般的に、多くのアプリケーションの設定は、LifeKeeper に用意されている Core システムのエラー検出機能を利用することができます。[リソースエラー回復シナリオ](#) および [サーバ障害回復シナリオ](#) の各トピックでは、共通する 2 つの障害状況を使用して、LifeKeeper のコア機能を示しています。

LifeKeeper には、エラー、アラーム、およびリカバリ手順を起動するイベントを定義するための完全な環境も用意されています。このインターフェースは、通常、システムエラーログ用のパターンマッチ定義 (/var/log/messages)、またはカスタムビルドのアプリケーション固有の監視プロセスが必要になります。

N-Way リカバリ

N-Way リカバリを使用すると、異なる複数のリソースを、クラスタ内の異なるバックアップサーバにフェイルオーバーす

ることができます。

[保護されたリソースに戻る](#)

管理作業

サーバプロパティの編集

1. サーバプロパティを編集するには、[サーバプロパティを表示する](#)場合と同様に、[Server Properties] ダイアログを表示してください。
2. 該当のサーバに適切な権限でログインした場合は、次の項目が編集可能になります。
 - [シャットダウン方法](#)
 - [フェイルオーバー確認](#)
3. 変更が加えられると、[Apply] ボタンが有効になります。このボタンをクリックすると変更が適用されます。ウィンドウは閉じられません。
4. 完了したら、[OK] をクリックし、変更内容を保存してウィンドウを閉じるか、または [Cancel] をクリックして、変更内容を適用せずにウィンドウを閉じます。

コミュニケーションパスの作成

サーバ間の LifeKeeper コミュニケーションパスを設定する前に、ハードウェアおよびソフトウェアのセットアップを検証します。詳細については、SPS for Linux リリースノートを参照してください。

サーバのペア間にコミュニケーションパスを作成するには、両方のサーバに個別にパスを定義する必要があります。LifeKeeper では、サーバのペア間に TCP (TCP/IP) と TTY の両方のコミュニケーションパスを作成することができます。TTY パスは、所定のペア間に 1 つだけ作成できます。これに対し、TCP コミュニケーションパスは、パスの終点となるローカルおよびリモートのアドレスを指定することで、サーバのペア間に複数作成することができます。所定のリモートサーバへの TCP パスを使用する順序を LifeKeeper に設定するには、優先値を使用します。


重要: 単一のコミュニケーションパスを使用した場合、互いに通信するクラスタ内のサーバの機能に支障をきたす可能性があります。単一のコミュニケーションパスを使用しているときに、そのコミュニケーションパスで障害が発生した場合、複数のサーバ上で同時に LifeKeeper の階層が使用可能になることがあります。これは、「偽のフェイルオーバー」と呼ばれます。また、TCP コミュニケーションパス上のネットワークラフティックが大きくなると、偽のフェイルオーバーや LifeKeeper の初期化の問題など、予期せぬ動作が生じる可能性があります。


1. 開始するには、次の 4 つの方法があります。
 - サーバアイコンを右クリックし、[サーバコンテキストメニュー](#)が表示されたら [Create Comm Path] をクリックしてください。
 - [グローバルツールバー](#)で、[Create Comm Path] ボタンをクリックしてください。
 - [サーバコンテキストツールバー](#)で (表示されている場合)、[Create Comm Path] ボタンをクリックして

ください。

- [\[Edit\] メニュー](#)で、**[Server]**、**[Create Comm Path]**の順に選択してください。
2. **[Create Comm Path]**というタイトルのダイアログが表示されます。表示されるオプションのそれぞれについて、**[Help]**をクリックすると、選択した項目の説明が表示されます。
 3. リストボックスから**[Local Server]**を選択し、**[Next]**をクリックしてください。
 4. リストボックスで1つ以上の**[Remote Servers]**を選択してください。リストボックスにリモートサーバが表示されていない(つまり、クラスタにまだ接続されていない)場合は、**[Add]**を使用して入力してください。ローカルとリモートの両方のサーバに対するネットワークアドレスが解決可能であることを確認する必要があります(たとえば、DNSで解決するか、`/etc/hosts`ファイルに追加します)。**[Next]**をクリックしてください。
 5. **[Device Type]**に対して**[TCP]**または**[TTY]**を選択して、**[Next]**をクリックしてください。
 6. **[Device Type]**が**[TCP]**に設定されている場合は、1つ以上の**[Local IP Addresses]**を選択してください。**[Device Type]**が**[TTY]**に設定されている場合は、**[Local TTY Device]**を選択してください。**[Next]**をクリックしてください。
 7. **[Device Type]**が**[TCP]**に設定されている場合は、**[Remote IP Address]**を選択してください。**[Device Type]**が**[TTY]**に設定されている場合は、**[Remote TTY Device]**を選択してください。**[Next]**をクリックしてください。
 8. **[Device Type]**が**[TCP]**に設定されている場合は、このコミュニケーションパスに対して**[Priority]**を入力または選択してください。**[Device Type]**が**[TTY]**に設定されている場合は、このコミュニケーションパスに対して**[Baud Rate]**を入力または選択してください。**[Next]**をクリックしてください。
 9. **[Create]**をクリックしてください。ネットワーク接続が正常に作成されたことを示すメッセージが表示されます。**[Next]**をクリックしてください。
 10. 複数のローカルIPアドレスまたは複数のリモートサーバを選択したときに、**[Device Type]**が**[TCP]**に設定されている場合は、手順6に戻り、次のコミュニケーションパスの設定を続けます。複数のリモートサーバを選択したときに、**[Device Type]**が**[TTY]**に設定されている場合は、手順5に戻り、次のコミュニケーションパスの設定を続けます。
 11. 終了メッセージが表示されたら、**[Done]**をクリックしてください。

[\[Server Properties\] ダイアログ](#)を表示するか、またはコマンド `lcdstatus -q`を入力することで、コミュニケーションパスを確認することができます。`lcdstatus`の使用方法については、LCD(1M)マニュアルページを参照してください。**[ALIVE]**ステータスが表示されます。

さらに、GUIの右ペインのサーバアイコンをチェックしてください。これが、作成済みの1つ目のコミュニケーションパスである場合は、1つのコミュニケーションパスが**[ALIVE]**であるが、冗長コミュニケーションパスがないことを示す黄色のハートビートがサーバアイコンに表示されます。 

[ALIVE]のコミュニケーションパスが2つ以上ある場合は、緑色のハートビートがサーバアイコンに表示されます。 

重要:IPv6アドレスを使用してコミュニケーションパスを作成する場合は、自動設定/ステートレスアドレスではなく、静的に割り当てられたアドレスを使用してください。自動設定/ステートレスアドレスは、時間がたつと変更され、コミュニケーションパスが使用できなくなる可能性があります。

数分たってもコミュニケーションパスが使用可能にならない場合は、ペアのサーバのコンピュータ名が正しいことを確認してください。TTY コミュニケーションパスを使用している場合は、2つのサーバ間のケーブル接続が正しく、緩んでいないことを確認してください。必要に応じて `portio (1M)` コマンドを使用して、TTY 接続の動作を確認してください。

コミュニケーションパスの削除

1. 開始するには、次の4つの方法があります。
 - サーバアイコンを右クリックして、[サーバコンテキストメニュー](#)が表示されたら **[Delete Comm Path]** をクリックしてください。
 - [グローバルツールバー](#)で、**[Delete Comm Path]** ボタンをクリックしてください。
 - [サーバコンテキストツールバー](#)で (表示された場合)、**[Delete Comm Path]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Server]**、**[Delete Comm Path]** の順に選択してください。
2. **[Delete Comm Path]** というタイトルのダイアログが表示されます。表示されるオプションのそれぞれについて、**[Help]** をクリックすると、選択した項目の説明が表示されます。
3. リストから **[Local Server]** を選択し、**[Next]** をクリックしてください。このダイアログが表示されるのは、[グローバルツールバー](#)の **[Delete Comm Path]** ボタンを使用するか、[\[Edit\] メニュー](#)の **[Server]** を選択して、削除を選択した場合のみです。
4. 削除するコミュニケーションパスを選択して、**[Next]** をクリックしてください。
5. **[Delete Comm Path(s)]** をクリックしてください。出力パネルが有効の場合、ダイアログが閉じ、コミュニケーションパスを削除するコマンドの結果が**出力パネル**に表示されます。有効でない場合は、ダイアログが表示されたままこれらの結果が表示されます。すべての結果が表示されたら、**[Done]** をクリックして終了してください。ネットワーク接続が正常に削除されたことを示すメッセージが表示されます。
6. **[Done]** をクリックして、ダイアログを閉じ、GUI ステータス表示に戻ってください。

サーバのプロパティ - フェイルオーバー

プライマリサーバがローカルリカバリを試行して失敗した場合、または完全に機能停止した場合、ほとんどのサーバ管理者は、LifeKeeper で保護されたリソースをバックアップサーバに復元することが必要になります。これがデフォルトの LifeKeeper の動作になります。ただし管理者によっては、保護されたリソースをリカバリサイトで自動的に稼動するようにはしたくない場合もあります。例えば、ディザスタリカバリ状況においてサーバ間のネットワーク接続の信頼性が低くなるような WAN 環境に LifeKeeper がインストールされている場合です。

デフォルトでは、保護されたすべてのリソースに対して自動フェイルオーバーが有効になっています。保護されたリソースに対する自動フェイルオーバーを無効にしたり、バックアップサーバへの自動フェイルオーバーを行わないようにするには、**[Server Properties]** の **[General]** タブにある **[Failover]** セクションを使用して、以下の通り設定してください。

クラスタ内の各サーバで次のようにします。

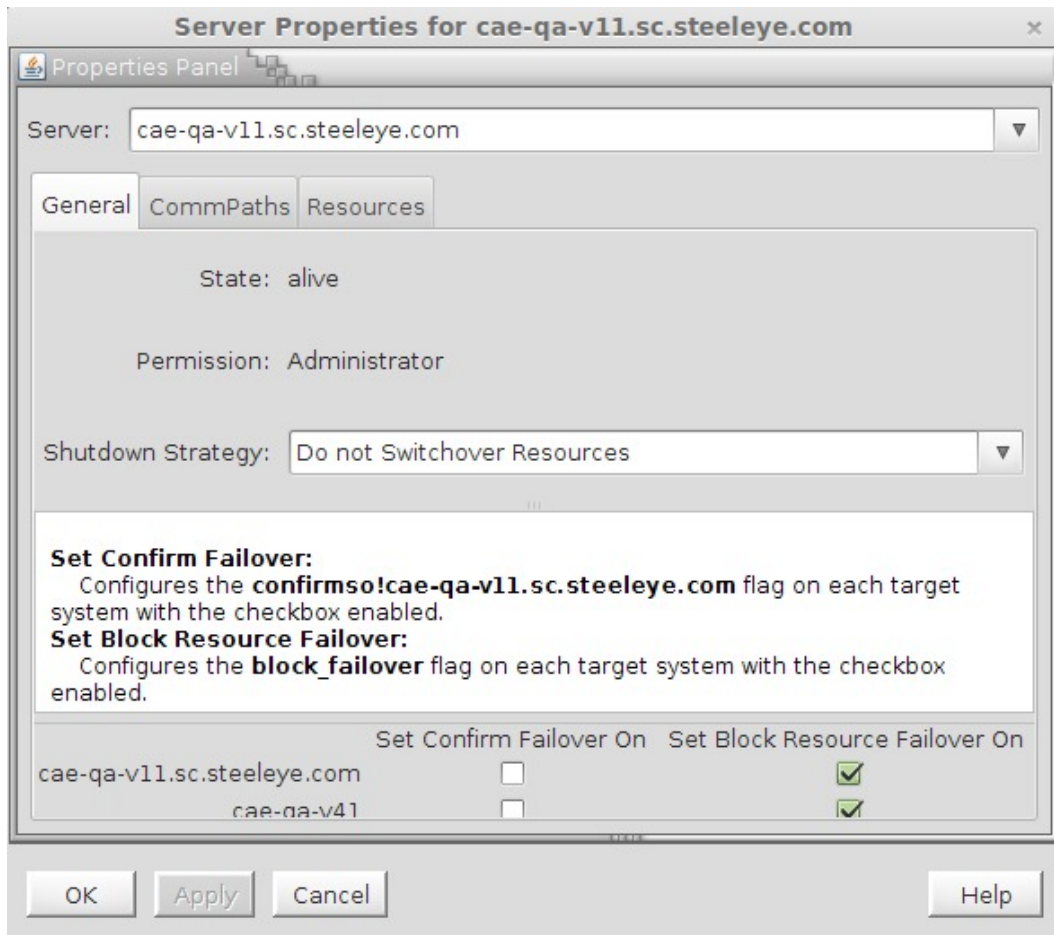
1. [サーバのプロパティを表示する](#)場合と同様に、**[Server Properties]** ダイアログを表示してください。
2. **[General]** タブを選択してください。[Server Properties] ダイアログの **[Failover]** セクションで、システムおよびリソースのフェイルオーバー機能を無効にするサーバをチェックしてください。デフォルトでは、LifeKeeper のすべてのフェイルオーバー機能が有効になっています。

[Set Confirm Failover On] 列で、ローカルサーバの完全な機能停止に対応するバックアップサーバとしての資格を喪失させるサーバを選択してください。

[Set Block Resource Failover On] 列で、このローカルサーバ上でリソース階層に障害が発生した場合に対応するバックアップサーバとしての資格を喪失させるサーバを選択してください。最初にシステムのフェイルオーバー機能を無効にしなければ、リソースのフェイルオーバーを無効にすることはできません。

選択内容を適用するには、**[Apply]** ボタンを押してください。

設定の詳細につきましては、[\[Confirm Failover\]](#) と [\[Block Resource Failover\]](#) の設定 のページを参照してください。



リソース階層の作成

- リソース階層の作成を開始するには、次の4つの方法があります。
 - サーバアイコン**を右クリックして、[サーバコンテキストメニュー](#)が表示されたら、**[Create Resource Hierarchy]**をクリックしてください。
 - [グローバルツールバー](#)で、**[Create Resource Hierarchy]** ボタンをクリックしてください。
 - [サーバコンテキストツールバー](#)で (表示された場合)、**[Create Resource Hierarchy]** ボタンをクリックしてください。
 - [Edit]** メニューで、**[Server]** を選択して、**[Create Resource Hierarchy]** をクリックしてください。
- [Create Resource Wizard] というタイトルのダイアログが表示され、クラスタ内にインストールされているすべての認識されたリカバリキットのリストが表示されます。アプリケーションを保護するためにリソース階層を構築する Recovery Kit を選択して、[Next] をクリックしてください。
- [Switchback Type] を選択して、[Next] をクリックしてください。
- [Server] を選択して、[Next] をクリックしてください。注記: サーバコンテキストメニューから開始した場合、クリックしたサーバアイコンから自動的にサーバが決定されるので、この手順はスキップされます。
- 続いて表示されるダイアログを使用して、作成しているリソース階層の種類に必要なデータを入力してください。

LifeKeeper アプリケーションリソース階層

LifeKeeper をリカバリキット無しでインストールした場合、デフォルトでは [Select Recovery Kit] リストに、ファイルシステムまたは Generic Application 用のオプションが含まれています。Generic Application のオプションは、関連付けられたリカバリキットがないアプリケーションに使用できます。

Raw I/O Recovery Kit または IP Recovery Kit (これらは両方とも、別個にパッケージ化され、LifeKeeper Core メディアに含まれている Core Recovery Kit です) をインストールした場合、[Select Recovery Kit] リストはこれらの Recovery Kit に対する追加のオプションを提供します。

これらの利用可能なオプションについては、以下のトピックを参照してください。

- [ファイルシステムリソース階層の作成](#)
- [Generic Application リソース階層の作成](#)
- [Raw デバイスリソース階層の作成](#)

IP Recovery Kit については、IP Recovery Kit テクニカルドキュメンテーションを参照してください。

Recovery Kit のオプション

インストールしたオプションの各リカバリキットは、[Select Recovery Kit] リストにエントリを追加します。たとえば、Oracle、Apache、および NFS の Recovery Kit がリストに表示されます。必要なリソース階層を作成する手順については、各リカバリキットに付属の管理ガイドを参照してください。

注記: ファイルシステムまたは論理ボリューム上に構築されるその他のアプリケーションリソース階層を作成する場合、最初に Logical Volume Manager (LVM) Recovery Kit をインストールする必要があります。

ファイルシステムリソース階層の作成

このオプションは、ファイルシステムのみを保護する場合 (たとえば、保護を必要とする共有ファイルがある場合) に使用します。

1. ファイルシステムリソース階層の作成を開始するには、次の4つの方法があります。
 - サーバアイコンを右クリックして、[サーバコンテキストメニュー](#)が表示されたら、**[Create Resource Hierarchy]** をクリックしてください。
 - [グローバルツールバー](#)で、**[Create Resource Hierarchy]** ボタンをクリックしてください。
 - [サーバコンテキストツールバー](#)で (表示された場合)、**[Create Resource Hierarchy]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Server]** を選択して、**[Create Resource Hierarchy]** をクリックしてください。
2. **[Create Resource Wizard]** というタイトルのダイアログが表示され、**[Recovery Kit]** リストが表示されます。**[File System Resource]** を選択して、**[Next]** をクリックしてください。
3. **[Switchback Type]** を選択して、**[Next]** をクリックしてください。
4. **[Server]** を選択して、**[Next]** をクリックしてください。注意: サーバコンテキストメニューから開始した場合、クリックしたサーバアイコンから自動的にサーバが決定されるので、この手順はスキップされます。
5. **[Create gen/filesys Resource]** ダイアログが表示されます。ファイルシステムリソース階層に対して **[Mount Point]** を選択し、**[Next]** をクリックしてください。選択したマウントポイントが、クラスタ内の別のサーバと共有されていることを確認するには、各ストレージキットをチェックして、マウントされたデバイスを共有として認識しているかどうかを確認します。ストレージキットがマウントされたデバイスを認識していない場合は、次のエラーダイアログが表示されます。

<file system> is not a shared file system

[OK] を選択すると、**[Create gen/filesys Resource]** ダイアログに戻ります。

注記:

- マウントポイントを選択リストに表示するには、そのマウントポイントがマウントされている必要があります。マウントポイントに対するエントリが `/etc/fstab` ファイルに存在する場合、階層の作成および拡張時にこのエントリが削除されます。NAS Recovery Kit を使用する前に (特にマウント設定が複雑な場合)、`/etc/fstab` のバックアップを作成することをお勧めします。`/etc/default/LifeKeeper` で調整可能な `REPLACEFSTAB=true|TRUE` を設定することにより、`/etc/fstab` のエントリが削除さ

れても、元に戻すように指定することができます。

- これらのリソースの多く(SIOS DataKeeper、LVM、Device Mapper Multipath など)は、ファイルシステムリソースを作成するために、クラスタ内の各サーバで LifeKeeper リカバリキットを必要とします。これらのキットが適切にインストールされていない場合、クラスタ内で共有されるファイルシステムが表示されません。
6. ファイルシステムリソース階層に対するデフォルトの **[Root Tag]** が作成されます。(これは、ステータス表示でこのリソースに使用されるラベルです)。このルートタグを選択するか、独自のルートタグを作成して、**[Next]** をクリックしてください。
 7. **[Create Instance]** をクリックしてください。インスタンス作成のステータスを示すメッセージが、ウィンドウに表示されます。
 8. **[Next]** をクリックしてください。ファイルシステム階層が正常に作成されたというメッセージが、ウィンドウに表示されます。
 9. この時点で、[ファイルシステムリソース階層の拡張](#)に移動するには **[Continue]** をクリックし、GUIに戻るには **[Cancel]** をクリックします。**[Cancel]** をクリックすると、階層が1つのサーバにしか存在しないという警告メッセージが表示され、この時点で保護されなくなります。

Generic Application リソース階層の作成

このオプションは、関連付けられたリカバリキットがないユーザ定義のアプリケーションを保護する場合に使用します。下記のユーザ指定スクリプトに対するテンプレートは、`$LKROOT/lkadm/subsys/gen/app/templates` に用意されています。これらのテンプレートは、保護するアプリケーション用にカスタマイズしてテストする前に、別のディレクトリにコピーしてください。

注記: ファイルシステム、ディスクパーティション、IP アドレスなどの他のリソースに依存するアプリケーションについては、これらの各リソースを個別に作成し、Create Dependency を使用して適切な依存関係を作成してください。

1. Generic Application リソース階層の作成を開始するには、次の4つの方法があります。
 - サーバアイコンを右クリックして、[サーバコンテキストメニュー](#)が表示されたら、**[Create Resource Hierarchy]** をクリックしてください。
 - [グローバルツールバー](#)で、[Create Resource Hierarchy] ボタンをクリックしてください。
 - [サーバコンテキストツールバー](#)で (表示された場合)、**[Create Resource Hierarchy]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Server]** を選択して、**[Create Resource Hierarchy]** をクリックしてください。
2. [Create Resource Wizard] というタイトルのダイアログが表示され、**[Recovery Kit]** リストが示されます。Generic Application を選択して、**[Next]** をクリックしてください。
3. **[Switchback Type]** を選択して、**[Next]** をクリックしてください。
4. **[Server]** を選択して、**[Next]** をクリックしてください。**注記:** サーバコンテキストメニューから開始した場合、クリックしたサーバアイコンから自動的にサーバが決定されるので、この手順はスキップされます。
5. 次のダイアログで、アプリケーションに対する **[Restore Script]** へのパスを入力し、**[Next]** をクリックしてくだ

さい。これは、アプリケーションを起動するコマンドです。テンプレートディレクトリに、テンプレート起動スクリプト `restore.template` が用意されています。この `restore` スクリプトが、既に起動されているアプリケーションに影響を与えてはなりません。

6. アプリケーションに対する **[Remove Script]** へのパスを入力し、**[Next]** をクリックしてください。これは、アプリケーションを停止するコマンドです。テンプレートディレクトリに、テンプレート停止スクリプト `remove.template` が用意されています。
7. アプリケーションに対する **[quickCheck Script]** へのパスを入力し、**[Next]** をクリックしてください。これは、アプリケーションを監視するコマンドです。テンプレートディレクトリに、テンプレート監視スクリプト `quickCheck.template` が用意されています。
8. アプリケーションに対する **[Local Recovery Script]** へのパスを入力し、**[Next]** をクリックしてください。これは、ローカルサーバ上の障害が発生したアプリケーションの復元を試みるコマンドです。テンプレートディレクトリに、テンプレート回復スクリプト `recover.template` が用意されています。
9. **[Application Information]** を入力し、**[Next]** をクリックしてください。これは、起動、停止、回復、監視の各スクリプトで必要になる可能性のあるアプリケーションに関するオプションの情報です。
10. **[Bring Resource In Service]** に対して [Yes] または [No] を選択し、**[Next]** をクリックしてください。[No] を選択すると、リソース状態が作成後に OSU に設定されます。[Yes] を選択すると、あらかじめ用意された `restore` スクリプトが実行されます。ファイルシステム、ディスクパーティション、IP アドレスなどの他のリソースに依存するアプリケーションについては、適切な依存リソースをまだ作成していない場合には、[No] を選択してください。
11. **[Root Tag]** を入力してください。これは、リソースインスタンスに対する一意の名前です。(これは、ステータス表示でこのリソースに対して表示されるラベルです。)
12. **[Create Instance]** をクリックして、作成プロセスを起動してください。インスタンス作成のステータスを示すメッセージが、ウィンドウに表示されます。
13. **[Next]** をクリックしてください。階層が正常に作成されたというメッセージが、ウィンドウに表示されます。
14. この時点で、[Generic Application リソース階層の拡張](#)に移動するには **[Continue]** をクリックし、GUIに戻るには **[Cancel]** をクリックします。**[Cancel]** をクリックすると、階層が1つのサーバにしか存在しないという警告が表示され、この時点で保護されなくなります。

Raw デバイスリソース階層の作成

このオプションは、Raw デバイスリソースを保護する場合に使用します。たとえば、既存のデータベース階層に追加する必要がある Raw デバイスに対して追加のテーブルスペースを作成する場合、このオプションを使用して Raw デバイスリソースを作成します。

注記: LifeKeeper では、ディスク論理ユニット (または LUN) レベルのディスクパーティションリソースを、一度に1つのクラスタの1つのシステムにロックします。

1. Raw デバイスリソース階層の作成を開始するには、次の4つの方法があります。
 - サーバアイコンを右クリックして、[サーバコンテキストメニュー](#)が表示されたら、**[Create Resource Hierarchy]** をクリックしてください。
 - [グローバルツールバー](#)で、**[Create Resource Hierarchy]** ボタンをクリックしてください。

- [サーバコンテキストツールバー](#)で (表示された場合)、**[Create Resource Hierarchy]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Server]** を選択して、**[Create Resource Hierarchy]** をクリックしてください。
2. **[Create Resource Wizard]** というタイトルのダイアログが表示され、**[Recovery Kit]** リストが表示されます。Raw デバイスを選択して、**[Next]** をクリックしてください。
 3. **[Switchback Type]** を選択して、**[Next]** をクリックしてください。
 4. **[Server]** を選択して、**[Next]** をクリックしてください。
注記: サーバコンテキストメニューから開始した場合、クリックしたサーバアイコンから自動的にサーバが決定されるので、この手順はスキップされます。
 5. このリソースが配置される共有ストレージデバイスで **[Raw Partition]** を選択して、**[Next]** をクリックしてください。
 6. **[Root Tag]** を入力してください。これは、リソースインスタンスに対する一意の名前です。(これは、ステータス表示でこのリソースに対して表示されるラベルです。)
 7. **[Create Instance]** をクリックして、作成プロセスを起動してください。作成中に何が発生したかを示すテキストが、**[Creating scsi/raw resource]** というタイトルのウィンドウに表示されます。
 8. **[Next]** をクリックしてください。階層が正常に作成されたというメッセージが、ウィンドウに表示されます。
 9. この時点で、[Raw リソース階層の拡張](#)に移動するには **[Continue]** をクリックし、GUIに戻るには **[Cancel]** をクリックします。**[Cancel]** をクリックすると、階層が1つのサーバにしか存在しないということを警告するメッセージが表示され、この時点で保護されなくなります。

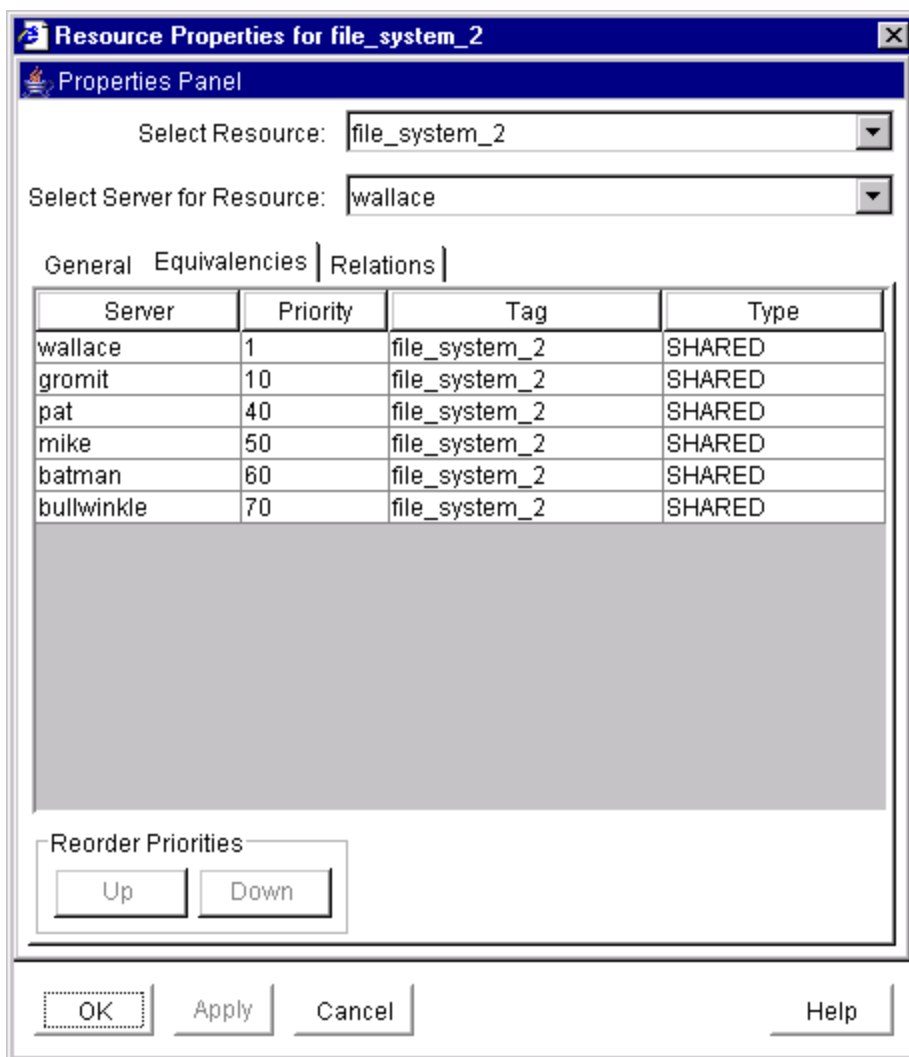
リソースのプロパティの編集

1. リソースのプロパティを編集するには、[リソースのプロパティを表示する](#)場合と同様に、**[Resource Properties]** ダイアログを表示してください。
2. 該当のサーバに適切な権限でログインした場合は、次の項目が編集可能になります。
 - スイッチバック
 - リソース設定 (特殊な設定を持つリソースの場合のみ)
 - [リソースの優先順位](#)
3. 変更が加えられると、**[Apply]** ボタンが有効になります。このボタンをクリックすると変更が適用されます。ウィンドウは閉じられません。
4. 完了したら、**[OK]** をクリックし、変更内容を保存してウィンドウを閉じるか、または **[Cancel]** をクリックして、変更内容を適用せずにウィンドウを閉じます。

リソースの優先順位の編集

リソース階層が定義されているサーバの優先順位は、編集または変更することができます。最初に、[リソースのプロパティを表示する](#)場合と同様に、**Resource Properties** ダイアログを表示してください。下に示すように

Resource Properties ダイアログの [Equivalencies] タブに、サーバ上の特定のリソースに対する優先順位が表示されます。



優先順位を変更するには、次の2つの方法があります。

- **[Up]/[Down]** ボタンを使用してイクイバレンシを移動することにより、優先順位を変更してください。または
- 優先順位の値を直接編集してください。

[Up] および [Down] ボタンの使用

1. Equivalencies 表で行をクリックして、イクイバレンシを選択してください。選択したイクイバレンシに応じて、**[Up]** または **[Down]** ボタンが有効になります。最も優先順位が高いサーバを選択した場合以外は、**[Up]** ボタンが有効になります。最も優先順位が低いサーバを選択した場合以外は、**[Down]** ボタン

が有効になります。

2. **[Up]** または **[Down]** をクリックして、優先順位リストのイクイバレンシを移動してください。

数値の優先順位の列は変更されませんが、イクイバレンシがリスト内で上下に移動します。

優先順位の値の編集

1. Equivalencies 表の Priority 列で優先順位 の値をクリックすることにより、優先順位 を選択してください。優先順位 の値の周りにボックスが表示され、値が強調表示されます。

2. 必要な優先順位を入力して、**Enter** を押してください。

- **注記:** 有効なサーバの優先順位は、1 ~ 999 です。

優先順位を編集した後、Equivalencies 表が再ソートされます。

変更の適用

Equivalencies 表で必要な優先順位を設定したら、**[Apply]** (または **[OK]**) をクリックして変更を適用します。**[Apply]** ボタンをクリックすると、加えられた変更内容が適用されます。**[OK]** ボタンをクリックすると、加えられた変更内容が適用され、ウィンドウが閉じられます。**[Cancel]** ボタンをクリックすると、**[Apply]** が直前にクリックされているので、加えられた変更内容を保存せずにウィンドウが閉じられます。

リソース階層の拡張

LifeKeeper の **[Extend Resource Hierarchy]** オプションは、あるサーバから既存の階層をコピーして、別の LifeKeeper サーバ上に同様の階層を作成します。階層が他のサーバに拡張されると、そのリソースに対してカスケーディングフェイルオーバーが使用可能になります。既存の階層が現在存在するサーバは、テンプレートサーバと呼ばれます。新たに拡張された階層が配置されるサーバは、ターゲットサーバと呼ばれます。

ターゲットサーバは、拡張された階層をサポートすることができ、他のリモートサーバ上の同等の階層と(アクティブな LifeKeeper コミュニケーションパスを介して)通信できなければなりません。つまり、既存の階層内のリソースに関連付けられているすべてのリカバリキットが、ターゲットサーバだけでなく、階層が現在存在する他のすべてのサーバに既にインストールされている必要があります。

1. GUI を介してリソース階層を拡張するには、次の5つの方法があります。

- 新しいリソース階層を**作成**してください。階層が作成されたことがダイアログに表示されたら、**[Continue]** ボタンをクリックして、Pre-Extend Wizard を介して新しい階層の拡張を開始してください。
- グローバルまたはサーバ固有のリソースアイコンを右クリックして、[リソースコンテキストメニュー](#)を表示し、次に **[Extend Resource Hierarchy]** をクリックして、Pre-Extend Wizard を介して選択したリソースを拡張してください。
- [グローバルツールバー](#)で、**[Extend Resource Hierarchy]** ボタンをクリックしてください。[Pre-Extend Wizard] ダイアログが表示されたら、**[Template Server]** および **[Tag to Extend]** を選択し、それぞれ選択した後に **[Next]** をクリックしてください。
- [リソースコンテキストツールバー](#)で (表示された場合)、**[Extend Resource Hierarchy]** ボタンをク

リックして、Pre-Extend Wizard を表示してください。

- [\[Edit\] メニュー](#)で、**[Resource]** を選択して、**[Extend Resource Hierarchy]** をクリックしてください。[Pre-Extend Wizard] ダイアログが表示されたら、**[Template Server]** および **[Tag to Extend]** を選択し、それぞれ選択した後に **[Next]** をクリックしてください。
2. デフォルトの **[Target Server]** を選択するか、または選択リストの中から1つ入力して、**[Next]** をクリックしてください。
 3. **[Switchback Type]** を選択して、**[Next]** をクリックしてください。
 4. デフォルト値を選択するか、または独自の **[Template Priority]** を入力して、**[Next]** をクリックしてください。
 5. 独自の **[Target Priority]** を選択するか入力して、**[Next]** をクリックしてください。
 6. ダイアログに、次に実行される拡張前のチェックが表示されます。これらのテストが成功した場合、LifeKeeper は、拡張している特定の種類のリソースに必要な手順の実行を開始します。

[Extend Resource Hierarchy] オプションの **[Accept Defaults]** ボタンは、LifeKeeper の **[Extend Resource Hierarchy]** のデフォルト値を熟知して、値の入力や確認をしなくて素早くLifeKeeper リソース階層を拡張したいユーザ向けです。GUI ダイアログを使用して対話的に段階を追ってLifeKeeper リソース階層を拡張する場合は、**[Next]** ボタンを選択します。

注記: マルチルート階層のすべてのルートは、まとめて拡張する必要があります。つまり、単一ルート階層として拡張することはできません。

注記: コマンドラインによる手順については、SAP ドキュメントのコマンドラインからの SAP リソースの拡張を参照してください。

ファイルシステムリソース階層の拡張

この操作は、[リソース階層の拡張](#) に関するセクションで説明されているように、[ファイルシステムリソース階層の作成](#) を完了した後に自動的に開始したり、既存のファイルシステムリソースから開始することができます。それが済んだら、次に以下の手順を完了します。これらの手順は、ファイルシステムリソースに固有のもので、

1. *[Extend gen/filesys Resource Hierarchy]* ダイアログボックスが表示されます。ファイルシステム階層に対して **[Mount Point]** を選択し、**[Next]** をクリックしてください。
2. LifeKeeper が提供する **[Root Tag]** を選択するか、またはターゲットサーバ上のリソース階層に対する独自のタグを入力して、**[Next]** をクリックしてください。
3. ダイアログに拡張操作のステータスが表示され、階層が正常に拡張されたことを示すメッセージが表示されて終了します。同じリソース階層を別のサーバに拡張する場合は、**[Next Server]** をクリックしてください。その場合は、拡張の操作が繰り返されます。または、**[Finish]** をクリックして、この操作を完了してください。
4. 拡張された階層が検証されると、確認情報がダイアログに表示されます。これが終了すると、**[Done]** ボタンが有効になります。**[Done]** をクリックして終了してください。

Generic Application リソース階層の拡張

この操作は、[リソース階層の拡張](#) に関するセクションで説明されているように、[Generic Application リソース階層](#)

[の作成](#)を終了した後自動的に開始したり、既存の Generic Application リソースから開始することができます。それが済んだら、次に以下の手順を完了します。これらの手順は、Generic Application リソースに固有のものであります。

1. LifeKeeper が提供する **[Root Tag]** を選択するか、またはターゲット サーバ上のリソース階層に対する独自のタグを入力して、**[Next]** をクリックしてください。
2. 次に **[Application Information]** (オプション) を入力し、**[Next]** をクリックしてください。
3. ダイアログに拡張操作のステータスが表示され、階層が正常に拡張されたことを示すメッセージが表示されて終了します。同じリソース階層を別のサーバに拡張する場合は、**[Next Server]** をクリックしてください。その場合は、拡張の操作が繰り返されます。または、**[Finish]** をクリックして、この操作を完了してください。
4. 拡張された階層が確認されると、確認情報がダイアログに表示されます。これが終了すると、**[Done]** ボタンが有効になります。**[Done]** をクリックして終了してください。

Raw デバイスリソース階層の拡張

この操作は、[リソース階層の拡張](#)に関するセクションで説明されているように、[Raw デバイスリソース階層の作成](#)を終了した後自動的に開始したり、既存の Raw デバイスリソースから開始することができます。それが済んだら、次に以下の手順を完了します。これらの手順は、Raw デバイスリソースに固有のものであります。

1. LifeKeeper が提供する **[Root Tag]** を選択するか、またはターゲット サーバ上のリソース階層に対する独自のタグを入力して、**[Next]** をクリックしてください。
2. ダイアログに拡張操作のステータスが表示され、階層が正常に拡張されたことを示すメッセージが表示されて終了します。同じリソース階層を別のサーバに拡張する場合は、**[Next Server]** をクリックしてください。その場合は、拡張の操作が繰り返されます。または、**[Finish]** をクリックして、この操作を完了してください。
3. 拡張された階層が検証されると、確認情報がダイアログに表示されます。これが終了すると、**[Done]** ボタンが有効になります。**[Done]** をクリックして終了してください。

階層の拡張解除

LifeKeeper の **[Unextend Resource Hierarchy]** オプションは、単一サーバから階層全体 (すべてのリソースを含む) を削除します。これは、すべてのサーバから1つの階層を削除する **[Delete Resource Hierarchy]** 選択項目とは異なります。

[Unextend Resource Hierarchy] を使用する場合、既存の階層を削除するサーバは、ターゲットサーバと呼ばれます。

[Unextend Resource Hierarchy] 選択項目は、ターゲットへのアクティブな LifeKeeper コミュニケーションパスを持つ LifeKeeper サーバから使用することができます。

1. 開始するには、次の5つの可能な方法があります。
 - 拡張解除したいリソース階層/サーバの組み合わせに対するアイコンを右クリックしてください。[リソースコンテキストメニュー](#)が表示されたら、**[Unextend Resource Hierarchy]** をクリックしてください。

- 拡張解除したいグローバルリソース階層のアイコンを右クリックしてください。[リソースコンテキストメニュー](#)が表示されたら、**[Unextend Resource Hierarchy]** をクリックしてください。ダイアログが表示されたら、リソース階層の拡張を解除するサーバを **[Target Server]** リストで選択し、**[Next]** をクリックしてください。
 - [グローバルツールバー](#)で、**[Unextend Resource Hierarchy]** ボタンをクリックしてください。ダイアログが表示されたら、リソース階層の拡張を解除するサーバを **[Target Server]** リストで選択し、**[Next]** をクリックしてください。次のダイアログで、**[Hierarchy to Unextend]** リストから拡張解除したいリソース階層を選択し、再度 **[Next]** をクリックしてください。
 - [リソースコンテキストツールバー](#)で (表示された場合)、**[Unextend Resource Hierarchy]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Resource]** をポイントして、**[Unextend Resource Hierarchy]** をクリックしてください。ダイアログが表示されたら、リソース階層の拡張を解除するサーバを **[Target Server]** リストで選択し、**[Next]** をクリックしてください。次のダイアログで、**[Hierarchy to Unextend]** リストから拡張解除したいリソース階層を選択し、再度 **[Next]** をクリックしてください。
2. 拡張解除するように指定したサーバおよびリソース階層を確認するメッセージが、ダイアログに表示されません。**[Unextend]** をクリックしてアクションを実行してください。
 3. [出力パネル](#)が有効な場合、ダイアログが閉じて、リソース階層の拡張を解除するコマンドの結果が出力パネルに表示されます。有効でない場合は、ダイアログが表示されたままこれらの結果が表示されます。すべての結果が表示されたら、**[Done]** をクリックして終了します。

リソース依存関係の作成

ほとんどの Recovery Kits では、元のリソース階層作成タスク中にそれらの依存関係が作成されますが、特定の条件下では、新規または追加のリソース依存関係を作成したり、既存のリソース依存関係を削除することが必要になる場合があります。一例として、既存の IP 依存関係を別の IP アドレスに変更する場合があります。リソース階層全体を削除して、新しいリソース階層を作成する代わりに、既存の IP 依存関係を削除して、異なる IP アドレスを持つ新しい依存関係を作成することができます。

1. 開始するには、次の4つの可能な方法があります。
 - 親子依存関係を追加したい、サーバの下の子サーバ固有のリソース、または親グローバルリソースに対するアイコンを右クリックしてください。[リソースコンテキストメニュー](#)が表示されたら、**[Create Dependency]** をクリックしてください。

注記: 右ペインでサーバ固有のリソースを右クリックした場合、**[Server]** の値はそのサーバになります。左ペインでグローバルリソースを右クリックした場合、**[Server]** の値は、リソースが最も高い優先度を持つサーバになります。
 - [グローバルツールバー](#)で、**[Create Dependency]** ボタンをクリックしてください。ダイアログが表示されたら、リソース依存関係の作成を開始するサーバを **[Server]** リストで選択し、**[Next]** をクリックしてください。次のダイアログで、**[Parent Resource Tag]** リストから親リソースを選択し、再度 **[Next]** をクリックしてください。
 - [リソースコンテキストツールバー](#)で (表示された場合)、**[Create Dependency]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Resource]** をポイントして、**[Create Dependency]** をクリックしてください。ダイア

ログが表示されたら、リソース依存関係の作成を開始するサーバを **[Server]** リストで選択し、**[Next]** をクリックしてください。次のダイアログで、**[Parent Resource Tag]** リストから親リソースを選択し、再度 **[Next]** をクリックしてください。

- サーバ上の既存の有効なリソースのドロップダウンボックスから、**[Child Resource Tag]** を選択してください。以下の例外を持つサーバ上で利用可能なすべてのリソースが、ダイアログに表示されます。
 - 親リソース、その先祖、およびその子。
 - 親リソースと同じサーバに拡張されていないリソース。
 - 親リソースと同じ相対優先度を持たないリソース。
 - 親リソースが稼働中の場合に、親と同じサーバ上で稼働していないリソース。

[Next] をクリックして、次のダイアログに進んでください。

- このダイアログで、依存関係の作成に対して適切な親および子のリソースタグが選択されていることを確認できます。**[Create Dependency]** をクリックして、親を拡張したクラスタ内のすべてのサーバで依存関係を作成してください。
- 出力パネル** が有効な場合、ダイアログが閉じて、依存関係を作成するコマンドの結果が出力パネルに表示されます。有効でない場合は、ダイアログが表示されたままこれらの結果が表示されます。すべての結果が表示されたら、**[Done]** をクリックして終了します。

リソース依存関係の削除

- 開始するには、次の4つの可能な方法があります。
 - 親子依存関係を削除したい、サーバの下の子サーバ固有のリソース、または親グローバルリソースに対するアイコンを右クリックしてください。[リソースコンテキストメニュー](#)が表示されたら、**[Delete Dependency]** をクリックしてください。
 - [グローバルツールバー](#)で、**[Delete Dependency]** ボタンをクリックしてください。ダイアログが表示されたら、リソース依存関係の削除を開始するサーバを **[Server]** リストで選択し、**[Next]** をクリックしてください。次のダイアログで、**[Parent Resource Tag]** リストから親リソースを選択し、再度 **[Next]** をクリックしてください。
 - [リソースコンテキストツールバー](#)で (表示された場合)、**[Delete Dependency]** ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、**[Resource]** をポイントして、**[Delete Dependency]** をクリックしてください。ダイアログが表示されたら、リソース依存関係の削除を開始するサーバを **[Server]** リストで選択し、**[Next]** をクリックしてください。次のダイアログで、**[Parent Resource Tag]** リストから親リソースを選択し、再度 **[Next]** をクリックしてください。
- ドロップダウンボックスから **[Child Resource Tag]** を選択してください。これは、削除したい依存関係における子のタグ名である必要があります。**[Next]** をクリックして、次のダイアログボックスに進んでください。
- このダイアログで、依存関係の削除に対して適切な親および子のリソースタグが選択されていることを確認できます。**[Delete Dependency]** をクリックして、クラスタ内のすべてのサーバ上の依存関係を削除してください。

4. [出力パネル](#)が有効な場合、ダイアログが閉じて、依存関係を削除するコマンドの結果が出力パネルに表示されます。有効でない場合は、ダイアログが表示されたままこれらの結果が表示されます。すべての結果が表示されたら、[Done]をクリックして終了します。

すべてのサーバからの階層の削除

1. 開始するには、次の5つの可能な方法があります。
 - 削除を開始するサーバの下、削除したい階層にあるリソースのアイコンを右クリックしてください。[リソースコンテキストメニュー](#)が表示されたら、[Delete Resource Hierarchy]をクリックしてください。
 - 削除したい階層にあるグローバルリソースのアイコンを右クリックしてください。[リソースコンテキストメニュー](#)が表示されたら、[Delete Resource Hierarchy]をクリックしてください。ダイアログが表示されたら、リソース階層の削除を開始するサーバを [Server] リストで選択し、[Next]をクリックしてください。
 - [グローバルツールバー](#)で、[Delete Resource Hierarchy] ボタンをクリックしてください。ダイアログが表示されたら、リソース階層の削除を開始するサーバを [Target Server] リストで選択し、[Next]をクリックしてください。次のダイアログで、[Hierarchy to Delete] リストから削除したい階層内のリソースを選択し、再度 [Next] をクリックしてください。
 - [プロパティパネルのリソースコンテキストツールバー](#)で (表示された場合)、[Delete Resource Hierarchy] ボタンをクリックしてください。
 - [\[Edit\] メニュー](#)で、[Resource] をポイントして、[Delete Resource Hierarchy] をクリックしてください。ダイアログが表示されたら、リソース階層の削除を開始するサーバを [Target Server] リストで選択し、[Next] をクリックしてください。次のダイアログで、[Hierarchy to Delete] リストから削除したい階層内のリソースを選択し、再度 [Next] をクリックしてください。
2. 削除するために指定した階層を確認するメッセージが、ダイアログに表示されます。[Delete] をクリックしてアクションを実行してください。
3. [出力パネル](#)が有効な場合、ダイアログが閉じて、階層を削除するコマンドの結果が出力パネルに表示されます。有効でない場合は、ダイアログが表示されたままこれらの結果が表示されます。すべての結果が表示されたら、[Done] をクリックして終了します。

LifeKeeper User Guide

[User Guide](#) は、検索可能な総合リソースで、LifeKeeper の GUI で実行できる多くの作業の詳細情報があります。このドキュメンテーションにアクセスするには、[User Guide](#) をクリックしてください。

GUI から実行できる作業は 3 つの分野に分類できます。

[共通の作業](#) - これらはどのユーザでも実行できる基本的な作業で、クラスタへの接続、サーバやリソースのプロパティの表示、ログファイルの表示、GUI の設定の変更などがあります。

[オペレータの作業](#) - これらは Operator (オペレータ) の権限を必要とする高度な作業で、リソースを in service および out of service にする操作などがあります。

[管理者の作業](#) - これらは、Administrator (管理者) の権限を必要とする作業です。サーバのプロパティの編集、リソースの作成、コミュニケーションパスの作成や削除などのサーバレベルの作業、およびリソースの編集、拡張、削除などのリソースレベルの作業があります。

以下の表に、それぞれのユーザ権限で使用できるデフォルトの作業を示します。特定のリソースタイプについてその他の作業が使用できることがあります。これらの作業については、関連するリソースキットのドキュメントで説明しています。

作業	権限		
	ゲスト	オペレータ	管理者
サーバとリソースの表示	X	X	X
サーバへの接続と切断	X	X	X
サーバのプロパティとログの表示	X	X	X
サーバのプロパティの変更			X
リソース階層の作成			X
コミュニケーションパスの作成と削除			X
リソースのプロパティの表示	X	X	X
リソースのプロパティの変更			X
リソースのサービス中とサービス休止の切り替え		X	X
リソース階層の拡張と拡張解除			X
リソースの依存関係の作成と削除			X
リソース階層の削除			X

LifeKeeper for Linux の使用

以下のトピックでは、LifeKeeper のグラフィカルユーザインターフェース (GUI)、および LifeKeeper の GUI から実行できる多数の作業について詳しく説明しています。

GUI

GUI のコンポーネントは、LifeKeeper Core のインストールの一部として、すでにインストールされています。

LifeKeeper の GUI は、Java テクノロジーを使用して、LifeKeeper およびその設定データ用にグラフィカルユーザインターフェースを提供します。LifeKeeper の GUI はクライアント / サーバアプリケーションなので、ユーザはクライアントシステムでグラフィカルユーザインターフェースを実行して、LifeKeeper が動作中のサーバシステムの監視や管理を行います。クライアントとサーバのコンポーネントは、同一システム上にある場合も、ない場合もあります。

GUI の概要 - 全般

クラスタマシンの必要なグループメンバシップをユーザが持っている限り、GUI を使用することで、任意のマシンから、任意のクラスタ内のサーバとリソースの管理、操作、または監視ができます (詳細については、[GUI のユーザの設定](#)を参照)。GUI のサーバとクライアントのコンポーネントについて説明します。

GUI サーバ

GUI サーバは、ハイパーテキスト転送プロトコル (HTTP) とリモートメソッド呼び出し (RMI) を使用して GUI クライアントと通信します。デフォルトでは、GUI サーバは LifeKeeper の起動時に初期化されますが、任意に設定することもできます。[GUI サーバの開始 / 停止](#)を参照してください。

GUI クライアント

GUI クライアントは、任意の LifeKeeper サーバ上の[アプリケーション](#)として、または Java が有効な任意のシステム上の[Web クライアント](#)として実行できます。

クライアントには、以下のコンポーネントがあります。

- 左上の[ステータスの表](#)には、接続しているサーバとそのリソースの上位のステータスが表示されます。
- 右上の[プロパティパネル](#)には、ステータスの表で直前に選択したオブジェクトの詳細情報が表示されます。
- 下部の[出力パネル](#)には、コマンドの出力が表示されます。
- ウィンドウの最下部にある[メッセージバー](#)には、処理のステータスメッセージが表示されます。
- コンテキストツールバー (プロパティパネル内) と[グローバルツールバー](#)を使用すると、頻繁に使用する作業に即座にアクセスできます。
- コンテキストメニュー (ポップアップ) と[グローバルメニュー](#)から、すべての作業にアクセスできます。

GUI クライアントの終了

[\[File\] メニュー](#)から **[Exit]** を選択すると、すべてのサーバから切断され、クライアントが終了します。

LifeKeeper GUI ソフトウェアパッケージ

LifeKeeper GUI は、LifeKeeper Core パッケージクラスタにバンドルされている **steeleye-lkGUI** ソフトウェアパッ

ページに含まれています。steeleye-lkGUI パッケージは、以下の動作を実行します。

- Java アーカイブフォーマットの LifeKeeper GUI クライアントをインストールする。
- LifeKeeper GUI サーバをインストールする。
- LifeKeeper 管理 Web サーバをインストールする。

注記: LifeKeeper 管理 Web サーバは、パブリック Web サーバとは異なるポート 81 を使用するように設定されます。

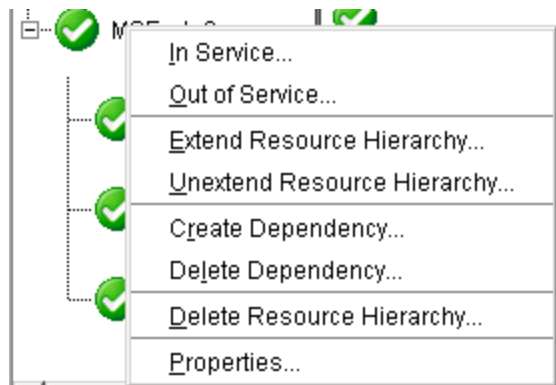
- ディレクトリ `/opt/LifeKeeper/htdocs/` に Java ポリシーファイルをインストールします。このファイルには、LifeKeeper GUI の実行に必要な最小限の権限があります。LifeKeeper GUI アプリケーションは、この場所にある `java.policy` ファイルを使用して、アクセスを制御します。
- GUI 管理用に LifeKeeper を準備する。

続行する前に、LifeKeeper サーバに LifeKeeper GUI パッケージがインストール済みであることを確認する必要があります。コマンド `rpm -qi steeleye-lkGUI` を入力して、このパッケージがインストール済みであるかどうかを確認できます。GUI パッケージがインストール済みである場合、出力にパッケージ名 `steeleye-lkGUI` が表示されます。

メニュー

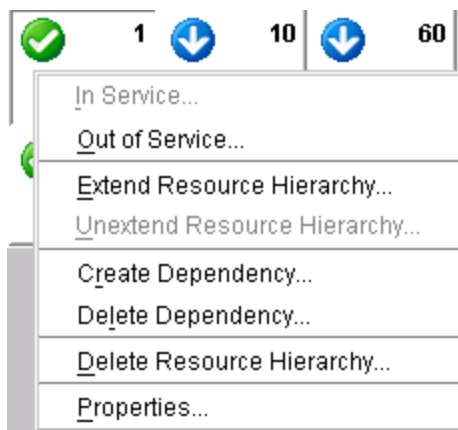
SIOS LifeKeeper for Linux のメニュー

リソースのコンテキストメニュー



リソースのコンテキストメニューは、[ステータスの表](#) 内にあるグローバル (クラスタ全体の) リソース (上図)、またはサーバ固有のリソースインスタンス (下図) を右クリックしたときに表示されます。デフォルトのリソースコンテキストメニューについて、ここで説明しますが、このメニューは特定のリソースタイプについてカスタマイズされていることがあります。この場合、メニューは該当するリソースキットのドキュメンテーションで説明されています。

選択したリソースについて、動作が呼び出されます。特定のサーバ上にあるリソースインスタンスを選択した場合、そのサーバについて動作が呼び出されます。一方、グローバル(クラスタ全体の)リソースを選択した場合は、サーバを選択する必要があります。



[In Service](#) -リソース階層を in service にします。

[Out of Service](#) -リソース階層を out of service にします。

[Extend Resource Hierarchy](#) -フェイルオーバをサポートするために、リソース階層を別のサーバに拡張します。

[Unextend Resource Hierarchy](#) -1 台のサーバから拡張リソース階層を削除します。

[Create Dependency](#) -2 つのリソース間に親 / 子の関係を作成します。

[Delete Dependency](#) -2 つのリソース間にある親 / 子の関係を削除します。

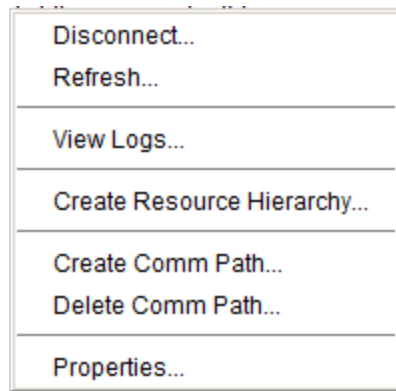
[Delete Resource Hierarchy](#) -リソース階層を LifeKeeper クラスタ内のすべてのサーバから削除します。

[Properties](#) -[Resource Properties] ダイアログを表示します。

サーバのコンテキストメニュー

サーバのコンテキストメニューは、[ステータスの表](#) 内にあるサーバアイコンを右クリックしたときに表示されます。このメニューは [Edit] メニューの [Server] サブメニューと同じですが、動作は常に、最初に選択したサーバ上で呼び出される点が異なります。

[File] メニュー



[Disconnect](#) - クラスタから切断します。

[Refresh](#) - GUI を最新情報に更新します。

[View Logs](#) - 接続しているサーバについて、LifeKeeper のログメッセージを表示します。

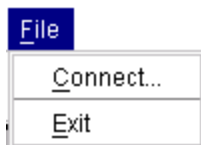
[Create Resource Hierarchy](#) - リソース階層を作成します。

[Create Comm Path](#) - サーバ間にコミュニケーションパスを作成します。

[Delete Comm Path](#) - サーバからコミュニケーションパスを削除します。

[Properties](#) - [Server Properties] ダイアログを表示します。

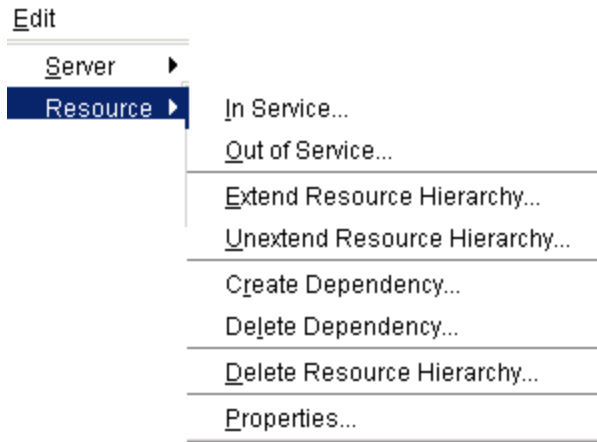
[File] メニュー



Connect - LifeKeeper クラスタに接続します。LifeKeeper クラスタ内の各サーバに接続するには、そのサーバでロギン認証が必要です。

Exit - すべてのサーバから切断し、GUI のウィンドウを閉じます。

[Edit] メニュー - [Resource]



[In Service](#) - リソース階層を in service にします。

[Out of Service](#) - リソース階層を out of service にします。

[Extend Resource Hierarchy](#) - フェイルオーバーをサポートするために、リソース階層を別のサーバに拡張します。

[Unextend Resource Hierarchy](#) - 1 台のサーバから拡張リソース階層を除去します。

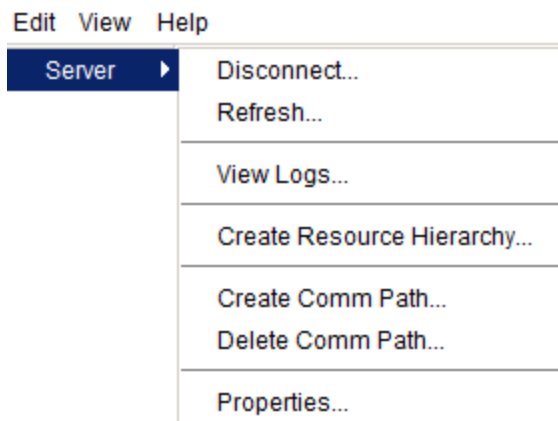
[Create Dependency](#) - 2 つのリソース間に親 / 子の関係を作成します。

[Delete Dependency](#) - 2 つのリソース間にある親 / 子の関係を削除します。

[Delete Resource Hierarchy](#) - リソース階層を LifeKeeper クラスタ内のすべてのサーバから削除します。

[Properties](#) - [\[Resource Properties\] ダイアログ](#)を表示します。

[Edit] メニュー - [Server]



[Disconnect](#) - クラスタから切断します。

Refresh - GUI を最新情報に更新します。

[View Logs](#) - 接続しているサーバについて、LifeKeeper のログメッセージを表示します。

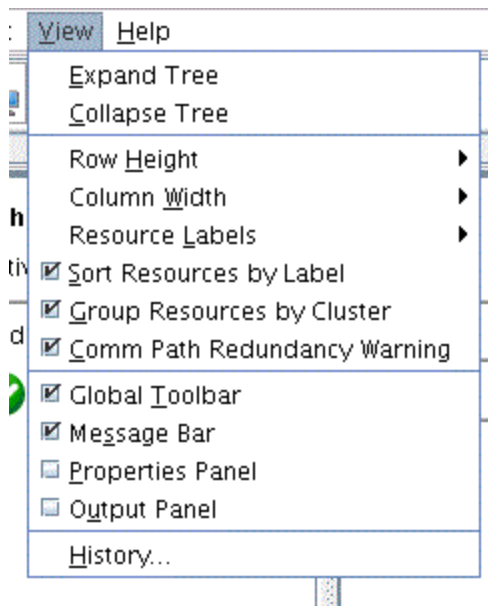
[Create Resource Hierarchy](#) - リソース階層を作成します。

[Create Comm Path](#) - サーバ間にコミュニケーションパスを作成します。

[Delete Comm Path](#) - サーバからコミュニケーションパスを削除します。

[Properties](#) - [Server Properties] ダイアログを表示します。

[View] メニュー



[Expand Tree](#) - リソース階層ツリー全体を展開します。

[Collapse Tree](#) - リソース階層ツリー全体を折り畳みます。

Row Height - リソース階層ツリーおよびリソーステーブルの行の高さを修正します。表示されたリソースの数により大、中、小を選択してください。

Column Width - リソース階層ツリーおよびリソーステーブルの列幅を修正します。表示されたリソースにより自動、大、中、小を選択してください。

Resource Labels - このオプショングループを使用すると、リソース階層ツリー内のリソースを、タグ名別とID別のいずれかで表示するかを指定できます。

Sort Resources by Label - リソースラベルのみでリソースを分類します。

Group Resources by Cluster -クラスタサーバおよびリソースラベルによって分類します。同じクラスタ内に属するリソースが同じグループになります。

Comm Path Redundancy Warning -サーバステータスグラフィックでコミュニケーションパスの状態についての説明を明記します。

- 選択した場合は、一組のサーバ間のコミュニケーションパスが冗長化されていない場合、サーバ警告が表示されます。
- 選択しない場合は、一組のサーバ間のコミュニケーションパスが冗長化されていない場合は無視しますが、コミュニケーションパスが切れた場合には、サーバ警告が表示されます。

Global Toolbar -チェックボックスがオンの場合、このコンポーネントを表示します。

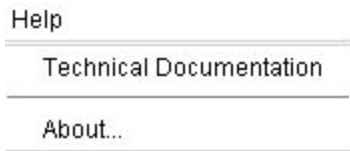
Message Bar -チェックボックスがオンの場合、このコンポーネントを表示します。

Properties Panel -チェックボックスがオンの場合、このコンポーネントを表示します。

Output Panel -チェックボックスがオンの場合、このコンポーネントを表示します。

History -メッセージバーに表示された最新メッセージを、LifeKeeper の GUI の [Message History] ダイアログボックスに表示します (最大 1000 行)。

[Help] メニュー



Technical Documentation - SIOS Technology Corp. のテクニカルドキュメンテーションの開始ページを表示します。

About... - LifeKeeper GUI のバージョン情報を表示します。

ツールバー

SIOS LifeKeeper for Linux のツールバー

GUI のツールバー

このツールバーは、[プロパティパネル](#)に表示されるデフォルトの[サーバ](#)と[リソース](#)のコンテキストツールバーを組み合わせたものですが、このツールバーから動作を実行するときには、サーバとリソースを選択する必要があります。



	Connect -LifeKeeper クラスタに接続します。
	Disconnect -LifeKeeper クラスタから切断します。
	Refresh -GUI を最新情報に更新します。
	View Logs -接続しているサーバについて、LifeKeeper のログメッセージを表示します。
	Create Resource Hierarchy -リソース階層を作成します。
	Delete Resource Hierarchy -リソース階層を LifeKeeper クラスタ内のすべてのサーバから削除します。
	Create Comm Path -サーバ間にコミュニケーションパスを作成します。
	Delete Comm Path -サーバからコミュニケーションパスを削除します。
	In Service -リソース階層を in service にします。
	Out of Service -リソース階層をサービス停止にします。




	Extend Resource Hierarchy -フェイルオーバをサポートするために、リソース階層を別のサーバに拡張します。
	Unextend Resource Hierarchy -1台のサーバから拡張リソース階層を削除します。
	Create Dependency -2つのリソース間に親 / 子の関係を作成します。
	Delete Dependency -2つのリソース間にある親 / 子の関係を削除します。
	Migrate Hierarchy to Multi-Site Cluster -既存の階層を Multi-Site Cluster 環境に移行します。

リソースのコンテキストツールバー

[ステータスの表](#) からサーバ固有のリソースインスタンスを選択すると、[プロパティパネル](#) にリソースのコンテキストツールバーが表示されます。

選択したサーバとリソースについて、動作が呼び出されます。灰色表示のリソースについて、動作を選択することはできません。



	In Service -リソース階層を in service にします。
	Out of Service -リソース階層を out of service にします。
	Extend Resource Hierarchy -フェイルオーバをサポートするために、リソース階層を別のサーバに拡張します。




	Unextend Resource Hierarchy -1 台のサーバから拡張リソース階層を削除します。
	Add Dependency -2 つのリソース間に親 / 子の関係を作成します。
	Remove Dependency -2 つのリソース間にある親 / 子の関係を削除します。
	Delete Resource Hierarchy -リソース階層をすべてのサーバから削除します。

サーバのコンテキストツールバー

[ステータスの表](#) からサーバを選択すると、[プロパティパネル](#) にサーバのコンテキストツールバーが表示されます。選択したサーバについて、動作が呼び出されます。



	Disconnect -LifeKeeper クラスタから切断します。
	Refresh -GUI を最新情報に更新します。
	View Logs -接続しているサーバについて、LifeKeeper のログメッセージを表示します。
	Create Resource Hierarchy -リソース階層を作成します。

	Delete Resource Hierarchy -リソース階層を LifeKeeper クラスタ内のすべてのサーバから削除します。
	Create Comm Path -サーバ間にコミュニケーションパスを作成します。
	Delete Comm Path -サーバからコミュニケーションパスを削除します。

GUI の実行の準備

LifeKeeper の GUI - 概要

LifeKeeper の GUI は、Java テクノロジーを使用して、LifeKeeper およびその設定データとのグラフィカルなステータスのインターフェースを提供します。LifeKeeper の GUI はクライアント / サーバアプリケーションなので、ユーザはクライアントシステムでグラフィカルユーザインターフェースを実行して、LifeKeeper が動作中のサーバシステムの監視や管理を行います。クライアントとサーバは、同一システム上にある場合も、ない場合もあります。クラスタマシンの必要なグループメンバシップをユーザが持っている限り、LifeKeeper の GUI を使用することで、任意のマシンから、任意のクラスタ内のサーバとリソースの管理、操作、または監視ができます (詳細については、[GUI のユーザの設定](#)を参照)。LifeKeeper GUI のサーバとクライアントのコンポーネントについて説明します。

GUI サーバ

システムの起動時に、LifeKeeper クラスタ内の各サーバで、LifeKeeper GUI サーバが初期化されます。LifeKeeper GUI サーバは、Java ネイティブインターフェース (JNI) 経由で LifeKeeper Core ソフトウェアと、リモートメソッド呼び出し (RMI) を使用して LifeKeeper GUI と通信します。

GUI クライアント

LifeKeeper GUI クライアントは、Linux システム上のアプリケーションとして、または Windows や Unix システム上の Web ブラウザから呼び出し可能なアプレットとして動作するように設計されています。

LifeKeeper GUI クライアントには、以下のグラフィカルコンポーネントがあります。

- 左上の[ステータスの表](#)には、接続しているサーバとそのリソースの上位のステータスが表示されます。
- 右上の[プロパティパネル](#)には、ステータスの表で直前に選択したオブジェクトの詳細情報が表示されず。
- 下部の[出力パネル](#)には、コマンドの出力が表示されます。

- ウィンドウの最下部にあるメッセージバーには、処理のステータスメッセージが表示されます。
- [サーバのコンテキスト](#) ツールバーと [リソースのコンテキスト](#) ツールバー (プロパティパネル内)、および [グローバルツールバー](#) からは、頻繁に使用する作業に即座にアクセスできます。
- [サーバのコンテキスト](#) メニューと [リソースのコンテキスト](#) メニュー (ポップアップ) および [グローバルメニュー](#) ([\[File\]](#)、[\[Edit Server\]](#)、[\[Edit Resource\]](#)、[\[View\]](#)、および [\[Help\]](#)) からは、すべての作業にアクセスできます。

グラフィックのリソース、サーバ、または表のセルを右クリックすると、コンテキストメニューが表示されます。また、多くの作業はこれらのコンテキストメニューから開始できます。この場合、リソースとサーバは自動的に指定されます。

GUI クライアントの開始

LifeKeeper GUI アプレットの開始

Web から LifeKeeper GUI アプレットを実行するには、好みの Web ブラウザを開き、URL `http://<server name>:81` (<server name> は LifeKeeper サーバの名前) に移動します。これにより、そのマシン上にある LifeKeeper GUI サーバから LifeKeeper GUI アプレットがロードされます。

ロードの完了後、[\[Cluster Connect\] ダイアログ](#)が表示されます。このダイアログで、任意の GUI サーバに接続できます。

注記: アプレットの実行時に、必須の Java プラグインがシステムにない場合、プラグインをダウンロードする Web サイトが自動的に表示されます。また、Java を有効にするように、[ブラウザのセキュリティパラメータ](#)を設定する必要があります。

パラメータが設定済みでもクライアントがロードされない場合は、[GUI のトラブルシューティング](#)を参照してください。

アプリケーションクライアントの開始

ある LifeKeeper サーバで管理者権限を持つユーザは、そのサーバからアプリケーションクライアントを実行できます。LifeKeeper GUI アプリケーションを開始するには、グラフィカルウィンドウから `/opt/LifeKeeper/bin/lkGUIapp` を実行してください。

この操作を実行してもクライアントがロードされない場合は、[GUI のトラブルシューティング](#)を参照してください。

GUI クライアントの終了

[\[File\] メニュー](#)から [\[Exit\]](#) を選択すると、すべてのサーバから切断され、クライアントが終了します。

LifeKeeper の GUI の設定

GUI 管理用の LifeKeeper サーバの設定

各 LifeKeeper サーバについて、以下の手順を実行してください。各手順には、詳細手順の参照先またはリンクがあります。

GUI の実行

1. 各サーバに、Java 実行時環境 (JRE) または Java ソフトウェア開発キット (JDK) をインストールする必要があります。必要な Java のバージョンおよび必要なダウンロードにアクセスするための URL については、SPS for Linux リリースノートを参照してください。注記: JRE は、SPS のインストールイメージファイルから、設定スクリプトを実行し、JRE のインストールのみを選択することでインストールできます (詳細については、SPS for Linux インストールガイドを参照)。
2. 各サーバで、LifeKeeper GUI サーバを開始してください ([GUI サーバの開始/停止](#)を参照)。注記: GUI サーバが後続の初期インストールを開始した後、LifeKeeper の開始と停止は、GUI サーバを含む LifeKeeper のすべてのデーモンプロセスの開始と停止を行います。
3. root 以外のユーザに GUI の使用を許可するように計画している場合は、[GUI ユーザの設定](#)が必要です。

GUI の実行

LifeKeeper の GUI は、以下の場所で実行できます。

- クラスタ内の LifeKeeper サーバ
- クラスタ外のリモートシステム

LifeKeeper クラスタ内のサーバで GUI の設定と実行を行う方法については、[LifeKeeper サーバでの GUI の実行](#)を参照してください。

LifeKeeper クラスタ外のリモートシステムで GUI の設定と実行を行う方法については、[リモートシステムでの GUI の実行](#)を参照してください。

GUI の設定

項目	説明
GUI のクライアントとサーバの通信	LifeKeeper GUI のクライアントとサーバは、通信に Java のリモートメソッド呼び出し (RMI) を使用します。RMI が正しく動作するためには、クライアントとサーバは解決可能なホスト名または IP アドレスを使用する必要があります。DNS が実装されていない場合 (または、他の名前解決メカニズムを使用して名前が解決できない場合) は、クライアントとサーバのそれぞれについて、 <code>/etc/hosts</code> ファイルを編集し、他のすべての LifeKeeper サーバの名前とアドレスを含めてください。

GUI サーバの Java プラットフォーム	<p>LifeKeeper GUI サーバには、Java 実行時環境 (JRE) - Java 仮想マシン、Java プラットフォームのコアクラス、およびサポートするファイル - をインストールする必要があります。JRE for Linux は、SPS for Linux インストールイメージファイルで提供されています (SPS for Linux インストールガイドを参照)。または、 http://www.oracle.com/technetwork/java/javase/downloads/index.html から直接ダウンロードすることもできます (注記: このサイトから直接ダウンロードする場合は、バージョン 1.8 (x64) をダウンロードしてください。また、32bit の Java 実行時環境 (JRE) はサポートしていません)。</p> <p>注記: デフォルトでは、LifeKeeper GUI サーバは、JRE が各サーバのディレクトリ <code>/usr/java/jre1.8.0_51</code> にインストールされていると予測します。JRE が見つからない場合、GUI サーバはディレクトリ <code>/usr/java/jdk1.8.0_51</code> から Java ソフトウェア開発キット (JDK) を探します。JRE または JDK を別のディレクトリの場所で使用する場合は、LifeKeeper のデフォルトファイル <code>/etc/default/LifeKeeper</code> の PATH を編集し、Java インタープリタ <code>java.exe</code> を持つディレクトリを含めてください。このファイルの編集時に LifeKeeper が実行中である場合は、変更内容を認識させるために LifeKeeper GUI サーバを停止し、再起動する必要があります。再起動しない場合、LifeKeeper GUI は Java コマンドを見つけることができません。</p>
Java リモートオブジェクトレジストリのサーバポート	<p>LifeKeeper GUI サーバは、各 LifeKeeper サーバ上の Java リモートオブジェクトレジストリ用にポート 82 を使用します。これにより、サーバは、典型的なファイアウォールの後にあるクライアントからの RMI 呼び出しをサポートできます。</p>
LifeKeeper の管理 Web サーバ	<p>LifeKeeper GUI サーバには、クライアントのブラウザの通信用に管理 Web サーバが必要です。現在、LifeKeeper GUI サーバは、管理 Web サーバとして <code>lighttpd</code> Web サーバのプライベートコピーを使用しています。この Web サーバは、<code>steeleye-lighttpd</code> パッケージによりインストールと設定が実行され、他の Web サーバとの競合を避けるためにポート 81 を使用します。</p>
GUI クライアントのネットワークアクセス	<p>LifeKeeper GUI クライアントには、LifeKeeper クラスタ内のすべてのホストへのネットワークアクセスが必要です。LifeKeeper GUI クライアントをブラウザ内で実行する場合、アプレットへのネットワークアクセスを可能にするためにセキュリティレベルを低下させる必要があります。セキュリティを低い値に設定した状態で、他のサイトを閲覧しないように注意してください (つまり、イントラネットまたは信頼できるサイトについてのみセキュリティ設定を変更する)。</p>

GUI の制限

項目	説明
GUI の相互運用性の制限	<p>LifeKeeper for Linux クライアントは、Linux サーバ上の LifeKeeper の管理にのみ使用できません。LifeKeeper for Linux の GUI と LifeKeeper for Windows は、同時には使用できません。</p>

GUI サーバの開始 / 停止

LifeKeeper GUI サーバを開始するには

LifeKeeper GUI サーバが動作していない場合は、`root` として以下のコマンドを入力してください。

```
/opt/LifeKeeper/bin/lkGUIserver start
```

トラブルシューティング

このコマンドは、管理しているサーバで LifeKeeper GUI サーバのデーモンプロセスが現在動作していない場合、それらのデーモンプロセスをすべて開始します。以下のようなメッセージが表示されます。

```
# Installing GUI Log
# LK GUI Server Startup at:
# Mon May 8 14:14:46 EDT 2006
# LifeKeeper GUI Server Startup completed at:
# Mon May 8 14:14:46 EDT 2006
```

LifeKeeper GUI サーバが開始した後、以降の LifeKeeper の開始操作はすべて、LifeKeeper GUI サーバのプロセスを自動的に開始します。

トラブルシューティング

LifeKeeper GUI は、各サーバのポート 81 を管理 Web サーバ用に、ポート 82 を Java リモートオブジェクトレジストリに使用します。他のアプリケーションがそれらのポートを使用している場合、LifeKeeper GUI は正しく機能しません。これらの値は、ファイル `/etc/default/LifeKeeper` の以下のエントリを編集することにより変更できます。

```
GUI_WEB_PORT=81 GUI_RMI_PORT=82
```

注記: これらのポートの値は、起動時に GUI サーバで初期化されます。ポートの値を変更した場合、`steeleye-lighttpd` プロセスを停止し、再起動する必要があります。これらの値は、接続するすべてのクラスタ全体で同一である必要があります。

LifeKeeper GUI サーバを停止するには

LifeKeeper GUI サーバが動作している場合は、`root` として以下のコマンドを入力してください。

```
/opt/LifeKeeper/bin/lkGUIserver stop
```

このコマンドは、管理しているサーバで LifeKeeper GUI サーバのデーモンプロセスが現在動作している場合、それらのデーモンプロセスをすべて停止します。以下のメッセージが表示されます。

```
# LifeKeeper GUI Server Shutdown at:
# Fri May 19 15:37:27 EDT 2006
# LifeKeeper GUI Server Shutdown Completed at:
# Fri May 19 15:37:28 EDT 2006
```

LifeKeeper GUI サーバのプロセス

LifeKeeper GUI サーバが動作していることを確認するには、以下のコマンドを入力してください。

```
ps -ef | grep runGuiSer
```

以下のような出力が表示されます。

```
root 2805 1 0 08:24 ?00:00:00 sh/opt/LifeKeeper/bin/runGuiSer
```

現在動作している他の GUI サーバのデーモンプロセスのリストを表示するには、以下のコマンドを入力してください。

```
ps -ef | grep S_LK
```

以下のような出力が表示されます。

```
root 30228 30145 0 11:20 ? 00:00:00 java -Xint -Xss3M
-DS_LK=true -Djava.rmi.server.hostname=thor48 ...
```

Java のセキュリティポリシー

LifeKeeper の GUI は、ポリシーベースのアクセス制御を使用します。GUI クライアントのロード時に、現在有効なセキュリティポリシーに基づいて権限が GUI クライアントに割り当てられます。ポリシーはさまざまな署名者 / 場所からのコードに提供される権限を指定し、外部から設定可能なポリシーファイルから初期化されます。

デフォルトでは、システム全体のポリシーファイルとオプションのユーザポリシーファイルが 1 つずつあります。システム全体でコードに権限を付与するシステムポリシーファイルが先にロードされ、次にユーザポリシーファイルが追加されます。LifeKeeper GUI がアプリケーションとして起動される場合は、これらのポリシーファイルに加えて、LifeKeeper GUI のポリシーファイルもロードされることがあります。

ポリシーファイルの場所

デフォルトでは、システムポリシーファイルは以下の場所にあります。

<JAVA.HOME>/lib/security/java.policy (Linux)

<JAVA.HOME>\lib\security\java.policy (Windows)

注記: JAVA.HOME は、システムのプロパティ「JAVA.HOME」の値を指し、JRE または JDK がインストールされたディレクトリの場所を指定します。

ユーザポリシーファイルは「.」の文字で始まり、デフォルトでは以下の場所にあります。

<USER.HOME>\.java.policy

注記: USER.HOME は、システムのプロパティ「user.home」の値を指し、ユーザのホームディレクトリを指定します。例えば、Windows NT ワークステーション上にあるユーザ Paul のホームディレクトリは、「paul.000」です。

Windows システムの場合、user.home のプロパティ値のデフォルト値は以下のとおりです。

C:\WINNT\Profiles**<USER>** (マルチユーザ **Windows NT** システム)

C:\WINDOWS\Profiles**<USER>** (マルチユーザ **Windows 95/98** システム)

C:\WINDOWS (シングルユーザ **Windows 95/98** システム)

デフォルトでは、LifeKeeper GUI のポリシーファイルは以下の場所にあります。

/opt/LifeKeeper/htdocs/java.policy (**Linux**)

ポリシーファイルの作成と管理

デフォルトでは、LifeKeeper GUI がアプリケーションとして起動される場合に LifeKeeper GUI のポリシーファイルが使用されます。LifeKeeper GUI をアプレットとして実行する場合、ホームディレクトリにユーザポリシーファイルを作

成する必要があります (存在しない場合)。ユーザポリシーファイルは、LifeKeeper GUI を実行するために必要な最低限の権限を指定する必要があります。このトピックの「ポリシーファイルの例」セクションで後述します。

ポリシーファイルの作成と管理は、単純なテキストエディタ、または Java 実行時環境 (JRE) や Java 開発キット (JDK) に含まれるグラフィカルな **Policy Tool** ユーティリティから行うことができます。Policy Tool を使用すると、入力が簡略化され、ポリシーファイルに必要な構文の知識が不要になります。Policy Tool の使用方法の詳細については、<http://docs.oracle.com/javase/8/docs/technotes/tools/>にある Policy Tool のドキュメンテーションを参照してください。

LifeKeeper GUI を実行するために必要な最低限の権限を持つ**ユーザポリシーファイルを作成する最も簡単な方法**は、`/opt/LifeKeeper/htdoc/java.policy`にある LifeKeeper GUI のポリシーファイルをホームディレクトリにコピーし、ファイル名を `.java.policy` に変更することです (ファイル名の前にあるドットは必須)。Windows システムでは、ファイル `http://<server name>:81/java.policy` (<server name> は LifeKeeper サーバのホスト名) を開いてホームディレクトリに `.java.policy` の名前を付けて保存することで、LifeKeeper GUI のポリシーファイルをコピーできます。ユーザポリシーファイルの正しい場所を特定する必要がある場合は、Java のコントロールパネルを使用して Java コンソールを有効にし、LifeKeeper GUI をアプレットとして起動します。ユーザポリシーファイルのホームディレクトリのパスが、Java コンソールに表示されます。

ポリシーファイルでの権限の付与

権限は、システムリソースへのアクセスを表します。アプレットにリソースへのアクセスを許可するには、対応する権限を、アクセスを試行するコードに明示的に付与する必要があります。権限は通常、名前を持ち (「ターゲット名」として参照される)、場合によっては、1 つ以上の動作を含むカンマ区切りリストを持ちます。例えば、以下のコードは、`/tmp` ディレクトリのファイル `abc` に対する読み取りアクセスを表す `FilePermission` オブジェクトを作成します。

```
perm = new java.io.FilePermission("/tmp/abc", "read");
```

この例では、ターゲット名は「`/tmp/abc`」、動作文字列は「`read`」です。

ポリシーファイルは、指定したコードソースからのコードに許可する権限を指定します。この例で、`/home/sysadmin` ディレクトリのコードにファイル `/tmp/abc` への読み取りアクセスを付与するポリシーファイルのエントリは以下のとおりです。

```
grant codeBase "file:/home/sysadmin/" { permissionjava.io.FilePermission
"/tmp/abc", "read"; };
```

ポリシーファイルの例

このポリシーファイルの例には、LifeKeeper GUI の実行に必要な最小限の権限があります。このポリシーファイルは、LifeKeeper GUI パッケージにより `/opt/LifeKeeper/htdoc/java.policy` にインストールされます。

```
/*
 * LifeKeeper GUI に必要な権限。コードベースで
 * これを制限することもできます。ただし、そのようにする場合は、リカバリキットが
 * 任意のコードベース付きの任意の jar コンポーネントを持つことができることに
 * 注意してください。したがって、これらも含めるには
 * grant 文を変更する必要があります。
 */
grant {
```

```

/*
 * LifeKeeper クラスタ内のすべてのマシンに対して
 * これを行うことができなければなりません。それに合わせてネットワーク仕様を
 * 制限することもできます。
 */
permission java.net.SocketPermission"*", "accept,connect,resolve";
/*
 * リモートプロパティファイルおよび
 * jar ファイルを取得するには、URLClassLoaders を使用してください。
 */
permission java.lang.RuntimePermission"createClassLoader";
/*
 * GUI をアプリケーションとして実行する場合のみ以下が必要です
 * (デフォルトの RMI セキュリティマネージャは、
 * ブラウザがアプレット用にインストールするセキュリティマネージャよりも
 * 制限あり)。
 */
permission java.util.PropertyPermission "*" ,"read";
permission java.awt.AWTPermission "*" ;
permission java.io.FilePermission "<<ALL FILES>>" ,"read,execute";
};

```

Java プラグイン

使用しているブラウザに関係なく([サポートするブラウザ](#)を参照)、ブラウザが初めて LifeKeeper GUI のロードを試行するときには、Java プラグインソフトウェアを自動ダウンロードするか、Java プラグインソフトウェアのダウンロードとインストールを行う Web ページを表示します。その後は、Java プラグインソフトウェアのテクノロジーをサポートする Web ページに遭遇するたびに、ブラウザは Java プラグインソフトウェアを自動的に起動します。

Java プラグインのダウンロード

Java プラグインソフトウェアは、Solaris、Linux、および Windows の Java 実行時環境 (JRE) の一部として含まれています。JRE のダウンロードには、お使いのネットワークとシステム設定のサイズにより、合計で 3 ~ 10 分かかります。ダウンロードの Web ページには、JRE と Java プラグインソフトウェアについての詳細なドキュメンテーションとインストール手順があります。

注記 1: プラグインのインストール後、およびプラグインのプロパティを変更するたびに、ブラウザを閉じて再起動する必要があります。

注記 2: LifeKeeper は、Java プラグインの version 8 update 51をサポートしています。

リモートシステムでの GUI の実行

LifeKeeper GUI を Java アプレットとして実行することにより、LifeKeeper クラスタ外の Linux、Unix、または Windows のシステムから LifeKeeper の管理ができます。この環境での GUI の設定と実行について、説明します。

リモートシステムでの GUI の設定

リモートの Linux、Unix、または Windows のシステムで LifeKeeper GUI を実行するには、使用するブラウザが JDK 1.7 (x64) アプレットをフルにサポートする必要があります。LifeKeeper GUI でサポートされているプラットフォームおよびブラウザについては、[SPS for Linux リリースノート](#)を参照してください。

- LifeKeeper GUI をアプレットとして実行する場合、ホームディレクトリにユーザポリシーファイルを作成する必要があります (存在しない場合)。ユーザポリシーファイルには、LifeKeeper GUI の実行に必要な最小限の権限を指定する必要があります。
 - LifeKeeper GUI を実行するために必要な最小限の権限を持つユーザポリシーファイルを作成する最も簡単な方法は、`/opt/LifeKeeper/htdoc/java.policy`にある LifeKeeper GUI のポリシーファイルをホームディレクトリにコピーし、ファイル名を `.java.policy`に変更します (ファイル名の前にあるドットは必須)。Windows システムでは、ファイル `http://<server name>:81/java.policy` (<server name> は LifeKeeper サーバのホスト名)を開いてホームディレクトリに `.java.policy` の名前を付けて保存することで、LifeKeeper GUI のポリシーファイルのコピーできます。ユーザポリシーファイルの正しい場所を特定する必要がある場合は、**Java のコントロールパネル**を使用して **Java コンソール**を有効にし、LifeKeeper GUI をアプレットとして起動します。ユーザポリシーファイルのホームディレクトリのパスが、Java コンソールに表示されます。
 - ユーザポリシーファイルがすでにある場合は、LifeKeeper サーバの `/opt/LifeKeeper/htdoc/java.policy`に指定されている必須エントリを、単純なテキストエディタを使用して既存のファイルに追加できます。詳細については、[Java のセキュリティポリシー](#)を参照してください。
- ブラウザのセキュリティパラメータを**低**に設定する必要があります。この設定では通常、Java と Java アプレットが有効になります。さまざまなブラウザとバージョンが存在するので、ブラウザのセキュリティパラメータの設定手順は[GUI アプレットを使用するためのブラウザのセキュリティパラメータの設定](#)で説明しています。

注記: セキュリティを低く設定した状態で外部サイトを閲覧するときには、注意が必要です。
- GUI を初めて実行するときに **Netscape** または **Internet Explorer** を使用し、かつ必要な Java プラグインがシステムにない場合、プラグインをダウンロードする Web サイトが自動的に表示されることがあります。必要な Java プラグインのバージョンとダウンロードにアクセスするための URL については、[SPS for Linux リリースノート](#)を参照してください。

リモートシステムでの GUI の実行

上記の作業を完了すると、リモートシステムで LifeKeeper GUI を Java アプレットとして実行できます。

- LifeKeeper GUI の Web ページの URL `http://<server name>:81` (<server name> は LifeKeeper サーバの名前)を開いてください。この Web ページには、LifeKeeper のスプラッシュ画面とアプレットがあります。Web ページが開くと、以下の動作が実行されます。
 - スプラッシュ画面が表示される。
 - アプレットがロードされる。
 - Java 仮想マシンが開始される。

- 一部のサーバファイルがダウンロードされる。
- アプレットが初期化される。

ネットワークとシステムの設定によっては、これらの動作に最大 20 秒かかることがあります。通常、アプレットのロード時と初期化時に、ブラウザには最小のステータスがいくつか表示されます。

すべてのものが正しくロードされた場合、アプレット領域に **[Start]** ボタンが表示されます。スプラッシュ画面に **[Start]** ボタンが表示されない場合、またはアプレットのロードと初期化が失敗した疑いがある場合は、「アプレットのトラブルシューティング」または[ネットワークに関するトラブルシューティング](#)を参照してください。

2. 要求されたら、**[Start]** をクリックしてください。LifeKeeper の GUI が表示され、[\[Cluster Connect\] ダイアログ](#)が自動的に表示されます。サーバが開始され、クラスターへの接続が確立した後、GUI のウィンドウに、接続しているサーバにより保護されているリソースとステータスがグラフィックで表示されます。GUI のメニューとツールバーのボタンから、LifeKeeper の管理機能を使用できます。

注記: 一部のブラウザでは、アプレットで作成されたウィンドウとダイアログに「**Warning: Applet Window**」が表示されます。これは通常の動作であり、無視できます。

アプレットのトラブルシューティング

アプレットのロードと初期化に失敗した疑いがある場合は、以下の操作を試してください。

1. アプレットが失敗したことを確認してください。通常、アプレットの状態を示すブラウザウィンドウ内に、メッセージが出力されます。**Netscape** と **Internet Explorer** では、テキストのステータスに加えて、アプレットの代わりにアイコンが表示されることがあります。アイコンをクリックすると、失敗の内容が表示される場合があります。
2. Java プラグインをインストールしていることを確認してください。問題が Java プラグインに関連する場合は、[Java プラグイン](#) のトピックを参照してください。
3. ブラウザの設定要件、特にセキュリティ設定を満たしていることを確認してください。詳細については、[GUI アプレットを使用するためのブラウザのセキュリティパラメータの設定](#)を参照してください。設定について明らかな間違いが見つからない場合は、次の手順に進んでください。
4. **Java コンソール**を開いてください。
 - **Firefox**、**Netscape**、および旧バージョンの **Internet Explorer** の場合は、マシンのコントロールパネルから **Java プラグイン** アプレットを実行し、コンソールを表示するオプションを選択してから、ブラウザを再起動してください。
 - 最新バージョンの **Internet Explorer** の場合は、**[Tools] > [Java Console]** を選択してください。[Java Console] のメニュー項目が表示されない場合は、**[Tools] > [Manage Add-Ons]** を選択し、コンソールを有効にしてください。その後、コンソールを表示するには、ブラウザの再起動が必要になることがあります。
 - **Mozilla** の場合は、**[Tools] > [Web Development] > [Java Console]** を選択してください。

5. URL `http://<server name>:81` を再度開いて、GUI アプレットを開始してください。Java プラグインのコンソールパネルを変更した場合は、ブラウザを再起動してください。
6. コンソールに表示されたメッセージを確認してください。メッセージは、問題の解決に役立ちます。問題がネットワークに関連する場合は、[ネットワークに関するトラブルシューティング](#)を参照してください。

LifeKeeper サーバでの GUI の実行

LifeKeeper GUI を実行する最も簡単な方法は、LifeKeeper サーバでアプリケーションとして実行することです。これは、実際には、同一システム上で GUI のクライアントとサーバを実行することです。

1. GUI 管理用の LifeKeeper サーバを設定した後、root として以下のコマンドを入力することにより、サーバ上で GUI をアプリケーションとして実行できます。

```
/opt/LifeKeeper/bin/lkGUIapp
```

2. lkGUIapp スクリプトが適切な環境変数を設定して、アプリケーションを開始します。アプリケーションのロード時に、LifeKeeper のアプリケーション指定ダイアログまたはスプラッシュ画面が表示されます。
3. アプリケーションのロード後、LifeKeeper の GUI が表示され、[Cluster Connect] ダイアログが自動的に表示されます。接続先のサーバ名、およびログインとパスワードを入力してください。
4. クラスタへの接続が確立した後、GUI のウィンドウに、接続しているサーバにより保護されているリソースとステータスがグラフィックで表示されます。GUI のメニューとツールバーのボタンから、管理機能を使用できます。

GUI アプレットを使用するためのブラウザのセキュリティパラメータ

警告:セキュリティを低い値に設定した状態での他のサイトの閲覧には注意してください。

Firefox

1. [Edit] メニューの [Preferences] を選択してください。
2. [Preferences] ダイアログボックスの [Content] を選択してください。
3. [Enable Java] と [Enable Java Script] のオプションを選択してください。
4. [Close] をクリックしてください。

Internet Explorer

セキュリティが最高の状態で Internet Explorer を使用するには、以下の手順で LifeKeeper サーバを信頼済みサイトのゾーンに追加してください。

1. [Tools] メニューの [Internet Options] をクリックしてください。
2. [Security] タブをクリックしてください。
3. [Trusted Sites] ゾーンを選択し、[Custom Level] をクリックしてください。

4. **[Reset custom settings]** の **[Medium/Low]** を選択し、**[Reset]** をクリックしてください。
5. **[Sites]** をクリックしてください。
6. 接続する LifeKeeper サーバのサーバ名とポート番号を入力してください (例: http://server1:81)。

以下の手順で行う別の方法 (セキュリティが低くなる可能性がある) もあります。

1. **[Tools]** メニューの **[Internet Options]** をクリックしてください。
2. **[Internet]** または **[Local Intranet]** を選択してください (リモートシステムと LifeKeeper クラスタが同じイントラネット上に存在するかどうかによって異なる)。
3. **[Security Level]** バーを **[Medium]** ([Internet] を選択した場合)、または **[Medium-low]** ([Local Intranet] を選択した場合) に調整してください。これらは、各ゾーンのデフォルト設定です。
4. **[OK]** をクリックしてください。

ステータスの表

ステータスの表には、接続しているサーバとそのリソースのステータスがグラフィック表示されます。以下の項目が表示されます。

- 最も上の行に、各サーバの状態。
- 左端の列に、各リソースのグローバル (サーバ全体での) 状態と親 / 子の関係。
- 残りのセルに、各サーバの各リソースの状態。

サーバとリソースの状態は、グラフィックス、テキスト、および色を使用して表示されます。サーバのテーブルの空白セルは、特定のリソースがそのサーバで定義されていないことを示します。

ステータスの表でサーバまたはリソースを選択した場合、その項目の詳細な状態の情報とコンテキスト依存ツールバーが [プロパティパネル](#) に表示されます。また、任意の項目のセルを右クリックすることで、該当する [サーバのコンテキストメニュー](#) または [リソースのコンテキストメニュー](#) をポップアップ表示できます。

ステータスの表は 2 つのセクションに分かれています。左右のセクションの境界を移動して、それらのセクションの相対サイズを変更できます。また、ステータスの表を折り畳んで、階層ツリーの上位項目のみを表示できます。ツリーの [リソース項目の折り畳み / 展開](#) を実行すると、表内にリストされる階層に対しても折り畳み / 展開が適用されます。

プロパティパネル

プロパティパネルには、ステータスの表から選択されたサーバまたはリソースのプロパティが表示されます。プロパティパネルは、[\[Server Properties\] ダイアログ](#) または [\[Resource Properties\] ダイアログ](#) と同じ機能を持ち、さらに一般的に使用するコマンドに即座にアクセスできるコンテキスト依存ツールバーがあります。このパネルの上部には、サーバを選択した場合は `server_name`、リソースを選択した場合は `server_name: resource_name` がキャプションとして表示されます。

プロパティパネルに表示されるコンテキスト依存ツールバーは、[サーバのコンテキストツールバー](#) と [リソースのコンテキストツールバー](#) です。サーバまたはリソースのツールバーもカスタマイズできます。ツールバーのカスタマイズの詳細については、該当する [Application Recovery Kit のドキュメンテーション](#) を参照してください。

プロパティパネル下部にあるボタンは、以下の機能を持ちます。

- **[Apply]** ボタンは、パネルの編集可能なプロパティに対する変更内容を適用します。このボタンが有効になるのは、編集可能なプロパティを変更した場合のみです。
- **[Reset]** ボタンは、サーバにすべてのプロパティの現在の値を照会し、これまで変更した内容を消去します。このボタンは常に有効です。
- **[Help]** ボタンは、プロパティパネルのコンテキスト依存ヘルプを表示します。このボタンは常に有効です。

プロパティパネルのサイズを増減するには、パネルの左端にある境界を左右にスライドしてください。このパネルを開閉するには、[\[View\] メニュー](#)の **[Properties Panel]** チェックボックスを使用してください。

出力パネル

出力パネルは、LifeKeeper GUI クライアントが送出したコマンドの出力を収集します。コマンドの実行開始時に、タイムスタンプ付きのラベルが出力パネルに追加され、そのラベルの下に、そのコマンドの出力がすべて追加されます。複数のコマンドを同時に実行する場合（通常は異なるサーバ上）、各コマンドの出力が対応するセッションに送られ、各コマンドの結果が見やすくなります。

出力パネルのサイズを増減するには、パネル上部にある境界を上下にスライドしてください。このパネルを開閉するには、[\[View\] メニュー](#)の **[Output Panel]** チェックボックスを使用してください。出力パネルを閉じているときには、各コマンドを開始するダイアログが表示されたままになり、このダイアログを閉じるまで出力がこのダイアログに表示されます。そして、このダイアログを閉じた後はコマンドの出力を確認できなくなります。出力パネルを再び開いた後は、LifeKeeper の GUI はデフォルトの動作に戻ります。

メッセージバー

メッセージバーは、[Status] ウィンドウの下に表示されます。メッセージが1行のテキストで表示されます。「Connecting to Server X」や「Failure to connect to Server X」などのメッセージが表示されます。

メッセージバーを非表示にするには、[\[View\] メニュー](#)の **[Message Bar]** チェックボックスをオフにします。

メッセージバーを表示するには、[View] メニューの **[Message Bar]** チェックボックスをオンにします。

メッセージバーに表示されたメッセージの履歴を表示する方法については、[メッセージ履歴の表示](#)を参照してください。

GUI の終了

[\[File\] メニュー](#)から **[Exit]** を選択すると、すべてのサーバから切断され、GUI のウィンドウが閉じます。

共通の作業

以下に、すべてのユーザが実行できる基本作業を示します。

LifeKeeper の起動

SPS ソフトウェアはすべて、`/opt/LifeKeeper` ディレクトリにインストールされます。

すべての[確認作業](#)が完了すると、両方のサーバで LifeKeeper を起動する準備が整います。このセクションでは、LifeKeeper サーバデーモンプロセスの起動について説明します。LifeKeeper GUI アプリケーションは、別個のコマンドを使用して起動され、[LifeKeeper GUI の設定](#)に説明されています。LifeKeeper には、LifeKeeper デーモンプロセスの起動と停止を行う[コマンドラインインターフェース](#)が用意されています。これらのデーモンプロセスは、LifeKeeper GUI を起動する前に実行する必要があります。

LifeKeeper サーバプロセスの起動

LifeKeeper がシステムで現在実行されていない場合は、すべてのサーバに対するユーザルートとして次のコマンドを入力してください。

```
/etc/init.d/lifekeeper start
```

数秒の遅延の後、情報メッセージが表示されます。

注記: LifeKeeper を起動するときに **LifeKeeper Distribution Enabling Package** を参照するエラーメッセージが表示された場合は、[LifeKeeper インストールイメージファイル](#)をインストール/再インストールする必要があります。

`/etc/init.d/lifekeeper start` コマンドの詳細については、コマンドラインに `man LCD` と入力して、LCD(1M) ヘルプページを参照してください。

LifeKeeper の自動再起動の有効化

上述のコマンドで LifeKeeper が起動しますが、システムを再起動するたびに毎回同じコマンドを実行する必要があります。サーバの起動時に LifeKeeper が自動的に起動するようにするには、次のコマンドを入力してください。

```
chkconfig lifekeeper on
```

詳細については、`chkconfig` マニュアルページを参照してください。

LifeKeeper の停止

LifeKeeper を停止する必要がある場合は、ルートとして次のコマンドを入力して停止してください。

```
/etc/init.d/lifekeeper stop-nofailover
```

このコマンドを使用すると、ローカルシステム上の LifeKeeper をシャットダウンします (実行されている場合)。まず、すべての保護されたリソースをローカルシステム上のサービスから取り除いてから、LifeKeeper デーモンをシャットダウンします。保護されたリソースは、クラスタ内の別のシステムにフェイルオーバーされません。LifeKeeper は、システムが再起動すると自動的に再起動されます。

```
/etc/init.d/lifekeeper stop-daemons
```

このコマンドを使用すると、リソースをサービスから取り除くセクションを実行しません。リソースはローカルシステム上で引き続き実行されますが、LifeKeeper の保護はなくなります。リソースを正常にシャットダウンしない場合、SCSI ロックなどの項目などを取り除くことができなくなるため、このコマンドの使用には注意が必要です。このコマンドを実行した後、そのシステムで障害が発生するかシステムがシャットダウンされた場合、システムは適切なリソースにフェイルオーバーを開始できません。LifeKeeper は、システムが再起動すると自動的に再起動されます。

```
/etc/init.d/lifekeeper stop
```

このコマンドを使用すると、サービスからリソースを取り除きますが、!nofailover! フラグ[LCDIf1ag(1M) を参照]を通信可能などのシステムにも設定しません。これは、shutdown_switchover フラグが設定されたらフェイルオーバーが実行されることを意味します。shutdown_switchover が設定されていない場合、このコマンドは、/etc/init.d/lifekeeper stop-nofailover と同様に動作します。LifeKeeper は、システムが再起動すると自動的に再起動されます。

```
/etc/init.d/lifekeeper stop-failover
```

このコマンドを使用すると、リソースをサービスから取り除き、フェイルオーバーが実行されます。このコマンドは、shutdown_switchover フラグを設定して /etc/init.d/lifekeeper stop コマンドを実行した場合と同様の動作をします。LifeKeeper は、システムが再起動すると自動的に再起動されます。

LifeKeeper の自動再起動の無効化

システムの再起動時に LifeKeeper が自動で再起動しないようにするには、次のコマンドを入力してください。

```
chkconfig lifekeeper off
```

詳細については、chkconfig マニュアルページを参照してください。

LifeKeeper プロセスの表示

現在実行されているすべての LifeKeeper Core デーモンプロセスのリストを表示するには、次のコマンドを実行してください。

```
ps -ef | grep LifeKeeper | grep -w bin | grep -v lklogmsg
```

出力の例を以下に示します。

```
root 11663 11662 0 14:03 pts/0 00:00:00 /bin/bash /etc/redhat-lsb/lb_start_daemon
/opt/LifeKeeper/sbin/runsvdir -P /opt/LifeKeeper/etc/service log: runit just
starte
d.....
.....

root 11666 11663 0 14:03 pts/0 00:00:00 /bin/bash -c ulimit -S -c 0 >/dev/null 2> &1 ;
/opt/LifeKeeper/sbin/runsvdir -P /opt/LifeKeeper/etc/service log: runit just
```

```

starte
d.....
.....

root 11880 11873 0 14:03 ? 00:00:00 /opt/LifeKeeper/bin/lk_logmgr -l/opt/LifeKeeper/out -
d/etc/default/LifeKeeper

root 12240 11877 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/lcm

root 12247 11879 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/ttymonlcm

root 12250 11876 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/lcd

root 12307 11874 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/lkcheck

root 12311 11875 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/lkscsid

root 12325 11871 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/lkvmhad

root 12335 12330 0 14:04 ? 00:00:00 /opt/LifeKeeper/bin/perl /opt/LifeKeeper/htdoc/cgi-
bin/DoRequest.fcgi

```

LifeKeeper の実行状態は次のコマンドで確認できます。

```
/opt/LifeKeeper/bin/lktest
```

LifeKeeper が実行中の場合は次のよう出力されます。

```

F S UID PID PPID C CLS PRI NI SZ STIME TIME CMD

4 S root 12240 11877 0 TS 39 -20 6209 14:04 00:00:00 lcm

4 S root 12247 11879 0 TS 39 -20 30643 14:04 00:00:00 ttymonlcm

4 S root 12250 11876 0 TS 29 -10 9575 14:04 00:00:00 lcd

```

LifeKeeper が実行中でない場合は何も出力されず、コマンドは 1 で終了します。

注記: グラフィカルユーザインターフェース (GUI) に必要なプロセスとともに、LifeKeeper Core デーモンを開始、停止、および監視する別の LifeKeeper プロセスも実行されています。プロセスのリストについては、[LifeKeeper 制御プロセスの表示](#) および [LifeKeeper GUI サーバプロセスの表示](#) を参照してください。また、ほとんどの LifeKeeper プロセスには子プロセス lklogmsg があり、予期しない出力をキャプチャして記録します。

LifeKeeper GUI サーバプロセスの表示

LifeKeeper GUI サーバが動作していることを確認するには、以下のコマンドを入力してください。

```
ps -ef | grep runGuiSer
```

以下のような出力が表示されます。

```
root 2805 1 0 8:24 ?00:00:00 sh /opt/LifeKeeper/bin/runGuiServer
```

現在動作している他の GUI サーバのデーモンプロセスのリストを表示するには、以下のコマンドを入力してください。

```
ps -efw | grep S_LK
```

以下のような出力が表示されます。

```
root 819 764 0 Oct16 ?00:00:00 java -Xint -Xss3M -DS_LK=true -
Djava.rmi.server.hostname=wake -Dcom.steeleye.LifeKeeper.rmiPort=82 -
Dcom.steeleye.LifeKeeper.LKROOT=/opt/LifeKeeper -DGUI_RMI_REGISTRY=internal -DGUI_
WEB_PORT=81 com.steeleye.LifeKeeper.beans.S_LK
```

LifeKeeper GUI サーバ管理 Web サーバが動作していることを確認するには、以下のコマンドを入力してください。

```
ps -ef|grep steeleye-light | egrep -v "lklogmsg|runsv"
```

以下のような出力が表示されます。

```
root 12330 11872 0 14:04 ? 00:00:00 /opt/LifeKeeper/sbin/steeleye-
lighttpd -D -f/opt/LifeKeeper/etc/lighttpd/lighttpd.conf
```

LifeKeeper の制御プロセスの表示

LifeKeeper の制御プロセスが動作していることを確認するには、以下のコマンドを入力してください。

```
ps -ef | grep runsv
```

以下のような出力が表示されます。

```
root 11663 11662 0 14:03 pts/0 00:00:00 /bin/bash /etc/redhat-lsb/lsb_start_daemon
/opt/LifeKeeper/sbin/runsvdir -P /opt/LifeKeeper/etc/service log:runit just
starte
d.....
.....

root 11666 11663 0 14:03 pts/0 00:00:00 /bin/bash -c ulimit -S -c 0 >/dev/null 2>&1 ;
/opt/LifeKeeper/sbin/runsvdir -P /opt/LifeKeeper/etc/service log: runit just
starte
d.....
.....

root 11667 11666 0 14:03 pts/0 00:00:00 /opt/LifeKeeper/sbin/runsvdir -P
/opt/LifeKeeper/etc/service log: runit just
starte
d.....
.....

root 11871 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lkvmhad
root 11872 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv steeleye-lighttpd
root 11873 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lk_logmgr
root 11874 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lkcheck
root 11875 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lkscsid
```

```
root 11876 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lcd
```

```
root 11877 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lcm
```

```
root 11878 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv lkguiserver
```

```
root 11879 11667 0 14:03 ? 00:00:00 /opt/LifeKeeper/sbin/runsv ttymonlcm
```

これらのプロセスは LifeKeeper Core のデーモンプロセスの開始、停止、および監視を行います。LifeKeeper を開始するにはこれらのプロセスが動作している必要があります。これらのプロセスは、デフォルトでシステムの起動時に開始するように設定されています。この動作は変更しないでください。

サーバのクラスタへの接続

1. 開始するには、以下の2つの方法があります。
 - [グローバルツールバー](#)の **[Connect]** ボタンをクリックする。
 - [\[File\] メニュー](#)の **[Connect]** をクリックする。
2. [\[Cluster Connect\] ダイアログ](#)の **[Server Name]** フィールドに、接続するクラスタ内のサーバ名を入力してください。

注記: IPv6 アドレスを使用する場合は、このアドレスを大かっこ [] で囲む必要があります。これにより、マシンの IPv6 アドレス経由で接続を確立できます。別の方法として、名前をアドレスに割り当てることができ、その名前を使用して接続できます。



3. **[Login]** と **[Password]** のフィールドに、指定のサーバ上で LifeKeeper が認証に使用するユーザのログイン名とパスワードを入力してください。
4. **[OK]** をクリックしてください。

GUI が正常に指定サーバに接続した場合、GUI は、新しいサーバが検出されなくなるまで、クラスタ内にあるすべての既知のサーバへの接続（およびステータス表示への追加）を継続します。

注記: 最初のログイン名とパスワードが、クラスタ内のサーバ上にあるクライアントで認証に失敗した場合、そのサーバでの別のログイン名とパスワードを入力するように要求されます。[\[Password\] ダイアログ](#)で **[Cancel]** を選択した場合、サーバへの接続は中止され、GUI はクラスタ内の残りのコンポーネントへの接続を継続します。

クラスタからの切断

この作業は、選択したサーバ経由で、GUI クライアントをクラスタ内のすべてのサーバから切断します。

1. 開始するには、以下の3つの方法があります。
 - [グローバルツールバー](#)の **[Disconnect]** ボタンをクリックする。
 - [\[Edit\] メニュー](#)の **[Server]** を選択し、**[Disconnect]** をクリックする。
 - [サーバのコンテキストツールバー](#)が表示される場合は、そこにある **[Disconnect]** ボタンをクリックする。
2. [\[Cluster Disconnect\] ダイアログ](#)の **[Select Server in Cluster]** リストから、切断するクラスタ内のサーバ名を選択してください。
3. **[OK]** をクリックしてください。クラスタ内の全サーバのリストを持つ **[Confirmation]** ダイアログが表示されます。
4. **[Confirmation]** ダイアログの **[OK]** をクリックして、クラスタ内の全サーバからの切断を確定してください。

クラスタからの切断後、そのクラスタ内にあるすべてのサーバが、GUI のステータス表示から消去されます。

接続サーバの表示

サーバの状態は、下図に示すように、表内のサーバのグラフィック表示で表されます。サーバアイコンが視覚的に示すサーバの状態の詳細については、[サーバの状態の表示](#)を参照してください。







サーバのステータスの表示

サーバの状態は、下図に示すように、表内のサーバのグラフィック表示で表されます。



サーバの状態	状態のシンボル	意味
--------	---------	----

ALIVE		<p>クライアントはサーバに有効な接続を行うことができます。</p> <p>このサーバから ALIVE のリモートサーバへのコミュニケーションパスが ALIVE です。</p> <p>DEAD とマークされたコミュニケーションパス、および DEAD のサーバをターゲットとするコミュニケーションパスは無視されます。これは、DEAD のサーバには DEAD のグラフィックで表されるからです。</p>
ALIVE		<p>クライアントはサーバに有効な接続を行うことができます。</p> <p>このサーバから指定リモートサーバへの1つ以上のコミュニケーションパスが DEAD です。</p> <p>このサーバから指定リモートサーバの間には、冗長コミュニケーションパスが存在しません。</p>
DEAD		<p>クラスタ内の他のサーバから DEAD として報告されました。</p>
UNKNOWN		<p>ネットワーク接続が失われました。最後に分かっている LifeKeeper の状態が ALIVE です。</p>

サーバのプロパティの表示

- 開始するには、以下の2つの方法があります。
 - プロパティを表示するサーバのアイコンを右クリックします。[サーバのコンテキストメニュー](#)が表示されたら、**[Properties]** をクリックします。サーバのプロパティは、サーバをクリックすると[プロパティパネル](#) (有効になっている場合) にも表示されます。
 - [\[Edit\] メニュー](#) の **[Server]** をポイントし、**[Properties]** をクリックします。ダイアログが表示されたら、表示するサーバを **[Server]** リストから選択します。
- 別のサーバのプロパティを表示する場合は、サーバを **[Server]** リストから選択してください。
- 確認が完了したら、**[OK]** をクリックしてウィンドウを閉じてください。

サーバのログファイルの表示

- 開始するには、以下の4つの方法があります。
 - サーバのアイコンを右クリックして[サーバのコンテキストメニュー](#)を表示し、次に **[View Log]** をクリックして [LifeKeeper Log Viewer] ダイアログを表示する。
 - [グローバルツールバー](#) の **[View Log]** ボタンをクリックし、[LifeKeeper Log Viewer] ダイアログの [Server] リストから、表示するサーバを選択する。
 - [サーバのコンテキストツールバー](#) が表示される場合は、その **[View Log]** ボタンをクリックする。

- **[Edit] メニュー**の **[Server]** をポイントし、**[View Log]** をクリックする。次に、**[LifeKeeper Log Viewer]** ダイアログの **[Server]** リストから、表示するサーバを選択する。
2. **グローバルツールバー**または **[Edit] メニュー**から検索を開始して、別のサーバのログを表示する場合は、**[LifeKeeper Log Viewer]** ダイアログの **[Server]** リストから、そのサーバを選択します。**サーバのコンテキストメニュー**または**サーバのコンテキストツールバー**から **[View Log]** を選択した場合は、この機能は使用できません。
 3. 確認が完了したら、**[OK]** をクリックして **[Log Viewer]** ダイアログを閉じてください。

リソースのタグとID の表示

リソースのタグとID を即座に表示するには、**[Status]** ウィンドウのリソースアイコンにカーソルを合わせて、マウスの左ボタンを1回押します(シングルクリック)。優先順位が最も低いサーバのリソースのタグとID がメッセージバーに表示されます。特定サーバ上にあるリソースのタグとID を表示する場合は、表内のリソースインスタンスセルを左クリックしてください。

メッセージバーに表示されるメッセージは、以下ようになります。

```
Resource Tag = ipdnet0-153.98.87.73, Resource ID = IP-153.98.87.73
```

特定の状況では、GUI がリソースID を特定できないことがあります。この場合は、リソースタグのみがメッセージバーに表示されます。

リソースのステータスの表示



















リソースのステータつまり状態は、**グローバルリソースのステータス**(すべてのサーバについて)と**サーバリソースのステータス**(1台のサーバ上)の2つの形式で表示されます。グローバルリソースのステータスは、**[Status]** ウィンドウの左ペインにある**リソース階層ツリー**に表示されます。サーバリソースのステータスは、リソースの列とサーバ行の交点にある表のセルにあります。






サーバリソースのステータス

下図に、アクティブ、スタンバイ、および不明のリソースステータスを持つサーバを示します。

- 「wallace」上のリソースはすべて、アクティブです。
- 「gromit」、「pat」、「mike」、および「batman」上のリソースはすべてスタンバイです。
- 「bullwinkle」上のリソースはすべて、不明です。





グローバルリソースのステータス

 wallace	 gromit	 pat	 mike	 batman	 bullwinkle
 1 Active	 10 StandBy	 20 StandBy	 30 StandBy	 40 StandBy	 50 Unknown
 1 Active	 10 StandBy	 20 StandBy	 30 StandBy	 40 StandBy	 50 Unknown

サーバリソースの状態	状態のシンボル	意味
アクティブ		このサーバ上でリソースは動作可能であり、保護されています。(ISP)
可用性の低下		このサーバ上でリソースは動作可能ですが、バックアップリソースによる保護はされていません。(ISU)
スタンバイ		サーバは、リソースの動作を引き継ぐことができます。(OSU)
障害		このサーバ上のリソースに問題が検出されました。例えば、リソースを In Service にする試行が失敗しました。(OSF)
不明		リソースが初期化されていないか (ILLSTATE)、このサーバで LifeKeeper が動作中ではありません。
	空のパネル	サーバのリソースが定義されていません。

グローバルリソースのステータス

 device-nfs18857	 1 Active	 10 StandBy	 20 StandBy	 30 StandBy	 40 StandBy	 50 Unknown
---	---	---	---	--	---	---

状態のシンボル	説明	意味 / 原因
	正常	リソースがアクティブ (ISP) で、バックアップがアクティブです。
	警告	リソースがアクティブ (ISP) です。1つ以上のバックアップが、不明または障害 (OSF) としてマークされています。
	障害。リソースが、いずれのサーバでもアクティブではありません (OSF)。	リソースが、通常の原因により Out of Service になりました。 リソースが、通常ではない方法により動作が停止しました。 リカバリは完了していないか、失敗しました。
	不明。利用可能な情報からは、状態を特定できませんでした。	複数のサーバがアクティブであることを告げています。 サーバへの接続が遮断されました。 サーバのリソースインスタンスがすべて、不明の状態です。

リソースのプロパティの表示

- 開始するには、以下の3つの方法があります。
 - プロパティを表示するリソース / サーバの組み合わせのアイコンを右クリックします。[リソースのコンテキストメニュー](#)が表示されたら、**[Properties]** をクリックします。リソースのプロパティは、[プロパティパネル](#) (有効になっている場合) にも表示されます。
 - プロパティを表示するグローバルリソースのアイコンを右クリックします。[リソースのコンテキストメニュー](#)が表示されたら、**[Properties]** をクリックします。ダイアログが表示されたら、表示するリソースが存在するサーバを **[Server]** リストから選択します。
 - [\[Edit\] メニュー](#) の **[Resource]** をポイントし、**[Properties]** をクリックします。ダイアログが表示されたら、プロパティを表示するリソースを **[Resource]** リストから選択し、表示するリソースが存在するサーバを **[Server]** リストから選択します。
- 別のリソースのプロパティを表示する場合は、リソースを **[Resource]** リストから選択してください。
- 別のサーバ上にあるリソースのプロパティを表示する場合は、サーバを **[Server]** リストから選択してください。
- 確認を終了したら、**[OK]** をクリックしてウィンドウを閉じてください。

Resource Labels

Resource Labels

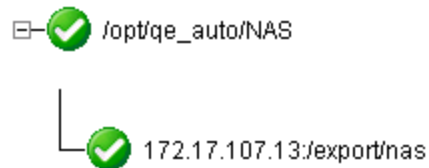
このオプショングループを使用すると、リソース階層ツリー内のリソースを、タグ名別とID別のいずれかで表示するかを指定できます。

注記: リソース階層ツリーに表示されるリソースタグ/IDは、優先順位が最も低い番号を持つサーバに属します。特定サーバ上にあるリソースのタグ/IDを表示する場合は、表内のリソースインスタンスセルを左クリックしてください。メッセージバーにそのタグ/IDが表示されます。

By tag name:



By ID:



メッセージ履歴の表示

1. [\[View\] メニュー](#)の **[History]** をクリックしてください。LifeKeeper GUI の [Message History] ダイアログが表示されます。
2. 履歴のメッセージをすべて消去する場合は、**[Clear]** をクリックしてください。
3. ダイアログを閉じるには、**[OK]** をクリックしてください。

[Message History] ダイアログには、メッセージバーからの最新のメッセージが表示されます。履歴リストには、最大 1000 行を表示できます。最大行数を超えた場合、新しいメッセージにより最も古いメッセージが「押し出され」ます。

これらのメッセージは、クライアントとサーバとの間の動作のみを表し、時系列で表示されます。最新のメッセージがリストの上部に表示されます。

メッセージ履歴の解釈

<-- は、メッセージがサーバから受信したことを示し、通常は以下の形式をとります。

```
<--"server name":"action"
<--"server name":"app res":"action"
<--"server name":"res instance":"action"
```



--> はメッセージがクライアントから送信されたことを示し、通常は以下の形式をとります。

```
-->"server name":"action"
-->"server name":"app res":"action"
-->"server name":"res instance":"action"
```



[Clear] ボタンをクリックすると、履歴が消去されますが、ダイアログは閉じません。

[OK] ボタンをクリックすると、履歴を消去せずにダイアログが閉じます。

リソース階層ツリーの展開と折り畳み

	<p>このツリーのセグメントでは、リソース <code>file_system_2</code> が展開されており、リソース <code>nfs-/opt/qe_auto/NFS/export1</code> が折り畳まれています。</p> <p>展開されているリソースアイコンの左には、 が表示されず、</p> <p>折り畳まれているリソースアイコンの左には、 が表示されます。</p>
--	--

リソース階層ツリーを展開するには、



-  をクリックするか、
-  の右側にあるリソースアイコンをダブルクリックしてください。

リソース階層ツリーをすべて展開するには、

- [View] メニューの [Expand Tree] をクリックするか、
- [Status] ウィンドウの左ペインにある列ヘッダの [Resource Hierarchy Tree] ボタンをダブルクリックしてください。

注記: リソース階層ツリーに表示されるリソースタグ / ID は、優先順位が最も低い番号を持つサーバに属します。特定サーバ上にあるリソースのタグ / ID を表示する場合は、表内のリソースインスタンスセルを左クリックしてください。メッセージバーにそのタグ / ID が表示されます。

リソース階層ツリーを折り畳むには、

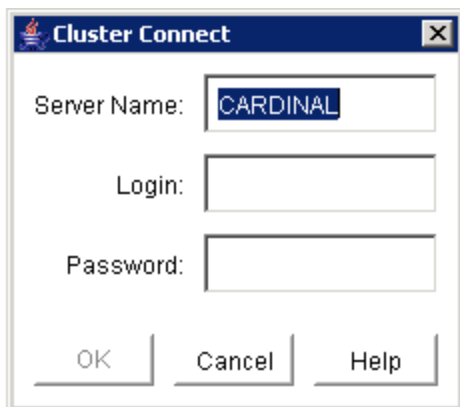
-  をクリックするか、
-  の右側にあるリソースアイコンをダブルクリックしてください。

リソース階層ツリーをすべて折り畳むには、

- [View] メニューの [Collapse Tree] をクリックするか、
- [Status] ウィンドウの左ペインにある列ヘッダの [Resource Hierarchy Tree] ボタンをダブルクリックしてください。

注記: 「9」と「0」のキーは、すべてのリソース階層ツリーに対して即座に展開 / 折り畳みを実行するホットキー / アクセラレータキーとして指定されています。

[Cluster Connect] ダイアログ

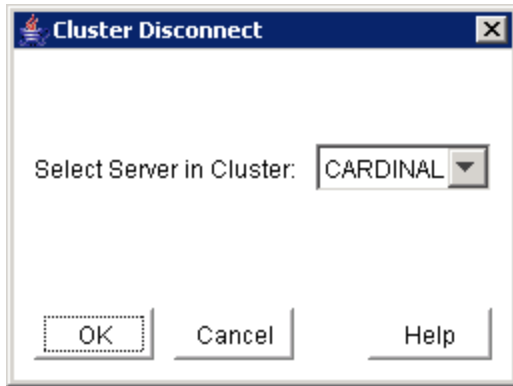


Server Name - 接続先のサーバ名。

Login - 接続先のサーバに LifeKeeper 認証情報を持つユーザのログイン名。

Password - 接続先のサーバで指定ログインを認証するパスワード。

[Cluster Disconnect] ダイアログ



Select Server in Cluster -

接続しているサーバの名前がドロップダウンリストボックスに表示されます。リストから、切断するクラスタのサーバを選択してください。切断されるクラスタ内のすべてのサーバが、確認ダイアログに表示されます。

[Resource Properties] ダイアログ

[Resource Properties] ダイアログは、[\[Edit\] メニュー](#)や[リソースコンテキストメニュー](#)から使用できます。このダイアログには、サーバ上にある特定のリソースのプロパティが表示されます。[Edit] メニューからアクセスした場合は、リソースとサーバを選択できます。リソースコンテキストメニューからアクセスした場合は、サーバを選択できます。

[General] タブ

- Tag - リソースインスタンスの名前。システムに対して一意で、管理者にリソースを示します。
- ID - リソースインスタンスに関連する文字列であり、リソースタイプのすべてのインスタンス間で一意です。関連するアプリケーションソフトウェアに対して、リソースインスタンスの内部特性のいくつかを示します。
- Switchback (管理者権限を持つユーザは編集可能) - サービス起動中のリソースが存在するサーバに障害が発生した場合に、サーバのリカバリ動作を管理する設定。この設定が[Intelligent]の場合、指定リソースの可能なバックアップとしてサーバが動作します。この設定が[Automatic]の場合、サーバはアクティブにリソースの再取得を試行します(以下の条件が満たされる場合)。
 - サーバがクラスタから離れるときには、リソース階層のサービスが既に起動している必要があります。
 - リソース階層がすべてサービス起動している場合は、低プライオリティのサーバでサービスを起動している必要があります。

注記: 自動スイッチバックのチェックは、LifeKeeper を起動したとき、またはクラスタに新しいサーバを追加したときにのみ実行されます。通常のクラスタ動作中には実行されません。

- State - リソースインスタンスの現在の状態。
 - *Active* - ローカルで in service であり、保護されています。
 - *Warning* - ローカルで in service ですが、ローカルリカバリは試行されません。
 - *Failed* - out of service、障害。
 - *Standby* - サービス停止、障害なし。
 - *LLSTATE* - LifeKeeper の起動シーケンスの一部として実行されるリソース初期化プロセスにより適切に初期化されていません。この状態のリソースは、LifeKeeper で保護されていません。
 - *UNKNOWN* - リソースの状態を特定できませんでした。GUI サーバが使用できない可能性があります。
- Reason - 存在する場合、リソースが現在の状態にある原因 (つまり、最後の状態変化の原因) を示します。例えば、galahad 上にあるアプリケーションの状態が OSU である原因は、tristan 上にある共有プライマリリソース *ordbfsaa-on-tristan* の状態が ISP か ISU であることです。共有リソースは、グループ内の 1 つのシステムでのみ同時にアクティブにできます。
- Initialization - 起動時のリソースの初期化動作を決定する設定であり、`AUTORES_ISP`、`INIT_ISP`、`INIT_OSU` などがあります。

[Relations] タブ

- Parent - このリソースに直接依存するリソースのタグ名を示します。
- Child - このリソースが依存するすべてのリソースのタグ名を示します。
- Root - このリソース階層で、親を持たないリソースのタグ名。

[Equivalencies] タブ

- Server - リソースが定義済みの同等性を持つサーバ名。
- Priority (管理者権限を持つユーザは編集可能) - このリソースについて、ターゲットサーバのフェイルオーバーの優先順位の値。
- Tag - 同等のサーバ上にあるこのリソースのタグ名。
- Type - 同等性のタイプ (`SHARED`、`COMMON`、`COMPOSITE`)。
- Reorder Priorities (管理者権限を持つユーザは編集可能) - [Up]/[Down] ボタンを使用して、選択した同等リソースの優先順位を並べ替えることができます。

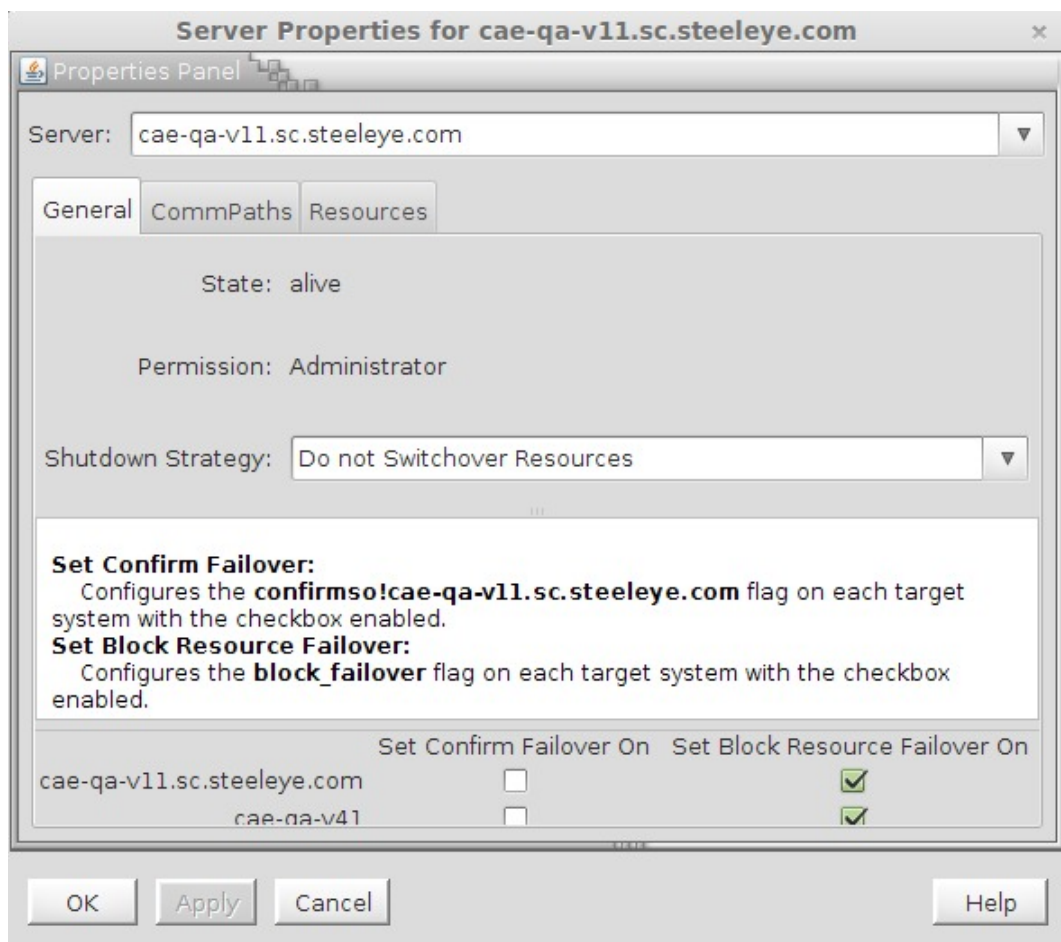
[OK] ボタンをクリックすると、変更内容が適用されてウィンドウが閉じます。[Apply] ボタンをクリックすると、変更内容が適用されます。[Cancel] ボタンをクリックすると、最後に [Apply] をクリックして以降の変更内容を保存せずに、ウィンドウが閉じます。

[Server Properties] ダイアログ

[Server Properties] ダイアログは、サーバのコンテキストメニューや[Edit]メニューから使用できます。このダイアログには、特定のサーバのプロパティが表示されます。サーバのプロパティは、[プロパティパネル](#) (有効になっている場合)にも表示されます。

このダイアログの3つのタブについて説明します。[OK] ボタンをクリックすると、変更内容が適用されてウィンドウが閉じます。[Apply] ボタンをクリックすると、変更内容が適用されます。[Cancel] ボタンをクリックすると、最後に[Apply] をクリックして以降の変更内容を保存せずに、ウィンドウが閉じます。

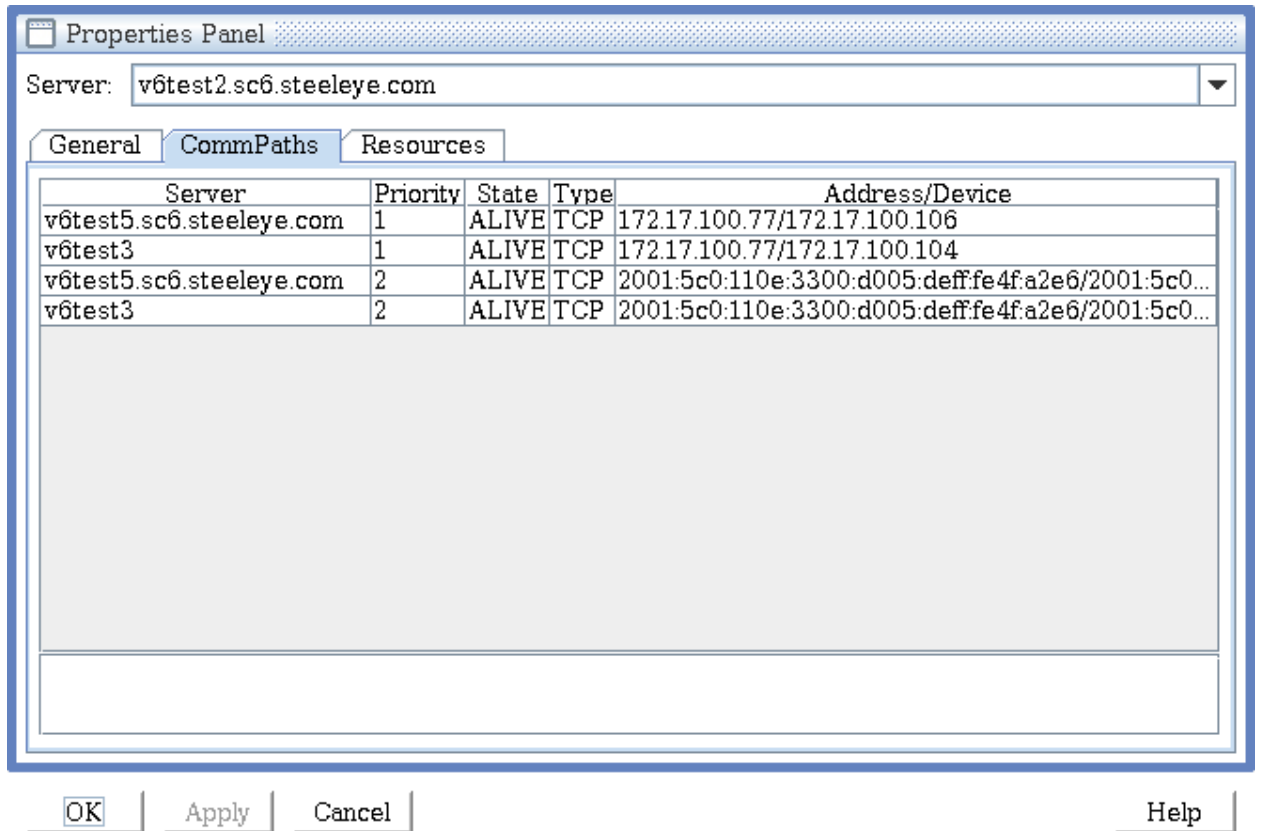
[General] タブ



- **Name** - 選択したサーバの名前。
- **State** - サーバの現在の状態。サーバの状態は以下の値をとります。

- *ALIVE* - サーバが使用可能。
- *DEAD* - サーバが使用不可。
- *UNKNOWN* - リソースの状態を特定できませんでした。GUI サーバが使用できない可能性があります。
- **Permission** - そのサーバに現在ログインしているユーザの権限レベル。権限は以下の値をとります。
 - *Administrator* - ユーザは LifeKeeper のすべての作業を実行できます。
 - *Operator* - ユーザは、LifeKeeperのリソースとサーバのステータスを監視でき、リソースのサービス起動やサービス停止ができます。
 - *Guest* - ユーザは LifeKeeper のリソースとサーバのステータスを監視できます。
- **Shutdown Strategy**(管理者権限を持つユーザは編集可能) - サーバがシャットダウンしたときに、リソースがクラスタ内のバックアップサーバにスイッチオーバーするかどうかを制御する設定。設定「*Switchover Resources*」は、リソースがクラスタ内のバックアップサーバでサービスが起動することを示します。設定「*Do not Switchover Resources*」は、リソースがクラスタ内にある別のサーバでサービスが起動しないことを示します。
- **Failover Strategy** - この設定を使用して、LifeKeeper のクラスタ内にある特定システムからのフェイルオーバーをユーザに確定するよう要求できます。この設定は、LifeKeeper の管理者のみが使用できます。オペレータとゲストには、この設定は表示されません。デフォルトでは、フェイルオーバーはすべて、ユーザの操作を必要とせず自動実行されます。ただし、confirm failover フラグが設定されると、指定システムからフェイルオーバーするには、以下のコマンドを実行して確定することが必要です。`lk_confirmso -y system`。以下のコマンドを実行して、フェイルオーバーをブロックできます。`lk_confirmso -n system`。指定期間内にこれらのコマンドのいずれかが実行されない限り、システムは事前プログラミングされたデフォルト動作を実行します。 `/etc/default/LifeKeeper` ファイル内にある2つのフラグが、この自動動作を制御します。
 - CONFIRMSODEF
これは、デフォルト動作を指定します。「0」に設定されている場合、デフォルト動作はフェイルオーバーを実行します。「1」に設定されている場合、デフォルト動作はフェイルオーバーをブロックします。
 - CONFIRMSOTO
これは秒単位で設定され、デフォルト動作を実行する前に LifeKeeper が待機する時間を示します。

[CommPaths] タブ



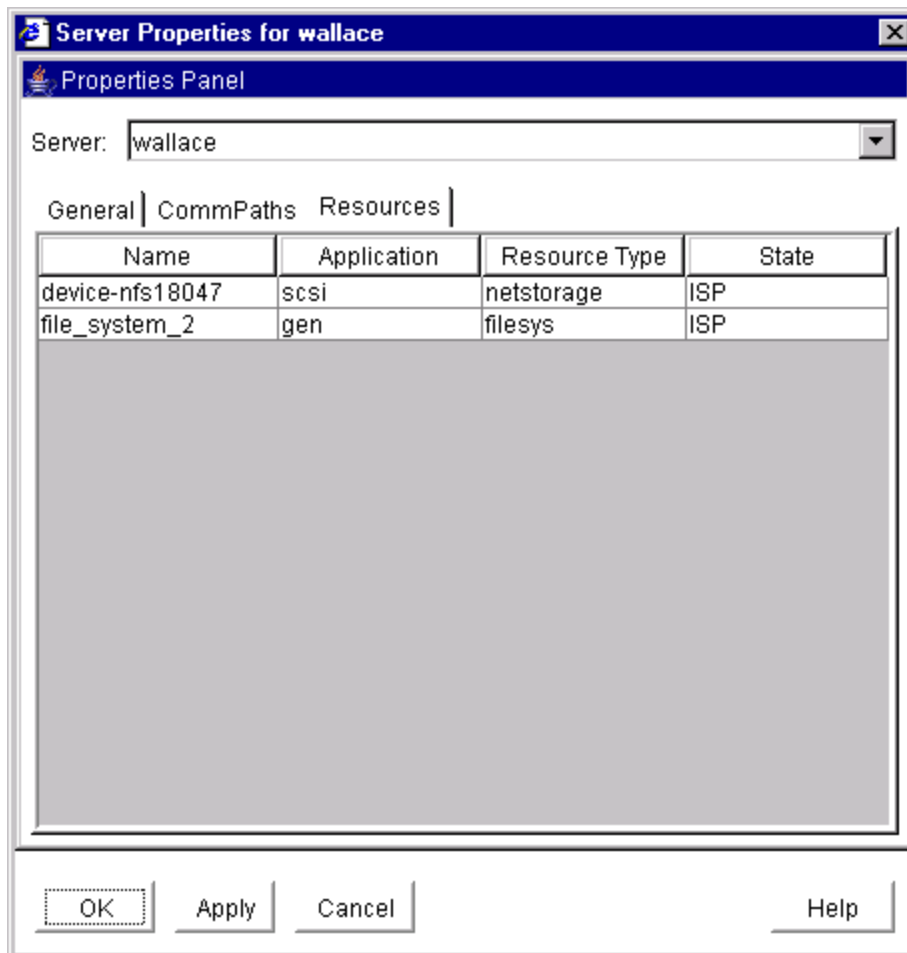
- **Server** -LifeKeeper のクラスタ内で、コミュニケーションパスが接続している他のサーバのサーバ名。
- **Priority** - 2 台のサーバ間でコミュニケーションパスを使用する順序を定義する優先順位。1 が最高の優先順位で、99 が最低の優先順位です。
- **State** -LifeKeeper の設定データベース (LCD) のコミュニケーションパスの状態。コミュニケーションパスの状態は以下の値をとります。
 - *ALIVE* - 通常の動作をしています。
 - *DEAD* - 通常の動作をしていません。
 - *UNKNOWN* - 状態を特定できませんでした。GUI サーバが使用できない可能性があります。
- **Type** -リスト内のサーバと、[Server] フィールドに指定されたサーバとの間のコミュニケーションパスの種類。TCP (TCP/IP)、または TTY。
- **Address/Device** - コミュニケーションパスが使用する IP アドレスまたはデバイス名。
- **Comm Path Status** - LifeKeeper の設定データベース (LCD) 内のコミュニケーションパスの状態に基づいて、GUI が判定したコミュニケーションパスのステータスの概要。以下に、コミュニケーションパスのステータ

[Resources] タブ

スの値を示します。これらの値は、下のパネルの詳細テキストの下に表示されます。

- *NORMAL* - すべてのコミュニケーションパスが通常の動作をしています。
- *FAILED* - 指定サーバに対するすべてのコミュニケーションパスが動作していません。
- *UNKNOWN* - コミュニケーションパスのステータスを特定できませんでした。GUIサーバが使用できない可能性があります。
- *WARNING* - 指定サーバに対する1つ以上のコミュニケーションパスが動作していません。
- *DEGRADED* - 指定サーバに対する1つ以上の冗長コミュニケーションパスが動作していません。
- *NONE DEFINED* - コミュニケーションパスが定義されていません。

[Resources] タブ



- **Name** - 選択したサーバ上にあるリソースインスタンスのタグ名。
- **Application** - リソースタイプのアプリケーション名 (gen、scsi など)。
- **Resource Type** - サービスを提供するリソースタイプ、ハードウェアのクラス、ソフトウェアのクラス、またはシステムのエンティティのクラス (app、filesys、nfs、device、disk など)。
- **State** - リソースインスタンスの現在の状態。
 - *ISP* - ローカルでサービス起動中であり、保護されています。
 - *ISU* - ローカルでサービス起動中ですが、ローカルリカバリは試行されません。
 - *OSF* - サービス停止、障害。
 - *OSU* - サービス停止、障害なし。
 - *LLSTATE* - LifeKeeper の起動シーケンスの一部として実行されるリソース初期化プロセスにより、リソースの状態が適切に初期化されていません。この状態のリソースは、LifeKeeper で保護されていません。
 - *UNKNOWN* - リソースの状態を特定できませんでした。GUI サーバが使用できない可能性があります。

オペレータの作業

以下のトピックは、オペレータの権限を必要とする高度な作業です。

リソースを In Service にする

1. 開始するには、以下の5つの方法があります。
 - in service にするリソース / サーバの組み合わせのアイコンを右クリックします。[リソースのコンテキストメニュー](#)が表示されたら、**[In Service]** をクリックします。
 - in service にするグローバルリソースのアイコンを右クリックします。[リソースのコンテキストメニュー](#)が表示されたら、**[In Service]** をクリックします。ダイアログが表示されたら、in service にするリソースが存在するサーバを **[Server]** リストから選択し、**[Next]** をクリックします。
 - [グローバルツールバー](#)の **[In Service]** ボタンをクリックします。ダイアログが表示されたら、in service にするリソースが存在するサーバを **[Server]** リストから選択し、**[Next]** をクリックします。次のダイアログで、in service にするリソースを1つ以上 **[Resource(s)]** リストから選択し、**[Next]** をもう一度クリックします。
 - [リソースのコンテキストツールバー](#)が表示される場合は、その **[In Service]** ボタンをクリックします。
 - [\[Edit\] メニュー](#)の **[Resource]** をポイントし、**[In Service]** をクリックします。ダイアログが表示されたら、in service にするリソースがあるサーバを **[Server]** リストから選択し、**[Next]** をクリックします。次のダイアログで、in service にするリソースを1つ以上 **[Resource(s)]** リストから選択し、**[Next]** をもう一度クリックします。
2. 選択したサーバとリソースを In Service にすることを示すダイアログボックスが表示されます。親リソースの in service にせずに依存する子リソースを in service にしようとする場合、このダイアログには警告も表示

されます。[In Service] をクリックして、依存する子リソースと共にリソースを in service にしてください。

3. [出力パネル](#)が有効の場合は、ダイアログが閉じ、リソースを In Service にするコマンドの結果が出力パネルに表示されます。出力パネルが無効の場合は、これらの結果を表示するダイアログが表示されたままになり、結果がすべて表示されたら **[Done]** をクリックします。in service になった追加の依存 (子) リソースが、ダイアログまたは出力パネルに表示されます。
4. リソースを in service にする動作で発生したエラーは、in service にするリソースがあるサーバの LifeKeeper ログに記録されます。

リソースを Out of Service にする

1. 開始するには、以下の4つの方法があります。
 - Out of Service にするグローバルリソース、またはリソース / サーバの組み合わせのアイコンを右クリックします。[リソースのコンテキストメニュー](#)が表示されたら、**[Out of Service]** をクリックします。
 - [グローバルツールバー](#)の [Out of Service] ボタンをクリックします。[\[Out of Service\]](#)ダイアログが表示されたら、Out of Service にするリソースを1つ以上 [Resource(s)] リストから選択し、**[Next]** をクリックします。
 - [リソースのコンテキストツールバー](#)が表示される場合は、**[Out of Service]** ボタンをクリックします。
 - [\[Edit\] メニュー](#)の **[Resource]** をポイントし、**[Out of Service]** をクリックします。[\[Out of Service\]](#)ダイアログが表示されたら、Out of Service にするリソースを1つ以上 [Resource(s)] リストから選択し、**[Next]** をクリックします。
2. 選択したリソースが Out of Service になることを示す **[Out of Service]** ダイアログボックスが表示されます。親リソースを Out of Service にせずに依存する子リソースを Out of Service にしようとする場合、このダイアログには警告も表示されます。**[Out of Service]** をクリックして、次のダイアログボックスに進みます。
3. [出力パネル](#)が有効の場合は、ダイアログが閉じ、リソースを Out of Service にするコマンドの結果が出力パネルに表示されます。出力パネルが無効の場合は、これらの結果を表示するダイアログが表示されたままになり、結果がすべて表示されたら **[Done]** をクリックします。
4. リソースを Out of Service にする動作で発生したエラーは、Out of Service にするリソースが存在するサーバの LifeKeeper ログに記録されます。

高度な作業

LCD

LifeKeeper 設定データベース

LifeKeeper 設定データベース (LCD) は、LifeKeeper が既知のすべてのリソースタイプについて、オブジェクト指向のリソース階層情報を管理し、リカバリ方向の情報を保存します。データは共有メモリにキャッシュされ、ファイルに保存されるので、システムの再起動後もデータが保持されます。LCD には、リカバリが必要なリソースインスタンスについての状態の情報、および特定の詳細情報もあります。

LCD のディレクトリ構造、保存されるデータタイプ、使用できるリソースタイプ、およびアプリケーションスクリプトの使用の詳細については、以下の関連トピックを参照してください。

関連トピック

LCDI のコマンド

LifeKeeper には、アプリケーションのリソース階層を定義するためのメカニズムが2つ用意されています。

- LifeKeeper の GUI
- LifeKeeper 設定データベースのインターフェース (LCDI) コマンド

LCDI は LifeKeeper が提供するインターフェースコマンドのセットで、使用するアプリケーションのニーズに合わせてリソース階層の設定の作成とカスタマイズができます。アプリケーションが複数のリソース (例: 2 つ以上のファイルシステム) に依存する場合、コマンドインターフェースを使用します。

コマンドの詳細については、LCDI のマニュアルページを参照してください。このトピックでは、開発シナリオを示し、GUI とコマンドの両方の機能を使用してリソース階層を作成できる方法を説明します。

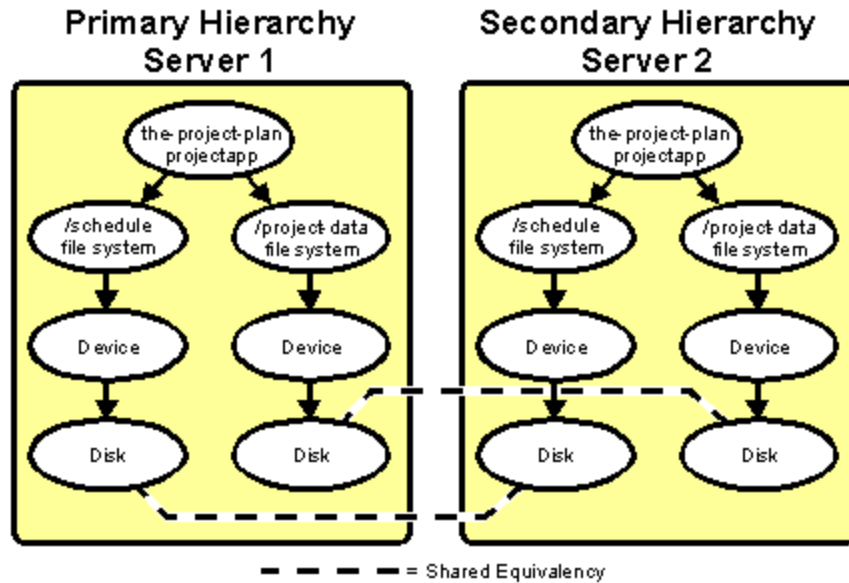
シナリオの状況

アプリケーションの例である ProjectPlan は、サーバ1 とサーバ2 が共有する SCSI ファイルシステムにデータを保存しています。サーバ1 が、アプリケーションのプライマリ階層にあります。アプリケーションには、`/project-data` と `/schedule` の2つのファイルシステムがあります。階層定義の最初の手順では、依存関係を指定します。

このアプリケーションの例には、以下の依存関係があります。

- **共有ファイルシステム。** アプリケーションは、`/project-data` と `/schedule` のファイルシステムに依存します。
- **SCSI ディスクサブシステム。** 次に、ファイルシステムは、SCSI ディスクサブシステム (デバイス、ディスク、およびホストアダプタのリソースを含む) に依存します。

結果として、階層を作成する作業は以下の図のようになります。



階層の定義

この例のアプリケーション階層を作成するために必要な作業を示します。

1. **ファイルシステムリソースの作成**。LifeKeeper の GUI には、ファイルシステムリソースを作成するメニューがあります。[ファイルシステムリソース階層の作成](#)を参照してください。

この定義作業の最後で、LCD では2つのファイルシステムリソースが以下のように定義されます。

ID	タグ	サーバ
/project-data	project-data-on-Server1	Server1
/project-data	project-data-from-Server1	Server2
/schedule	schedule-on-Server1	Server1
/schedule	schedule-from-Server1	Server2

注記: LifeKeeper で使用されるタグ名には意味はありません。単なるラベルです。表内のタグ名は LifeKeeper のデフォルト値です。

2. **リソースの定義**。この例では、以下の項目を定義する必要があります。

アプリケーション:	projectapp
リソースタイプ:	plan
インスタンス ID:	1yrplan
タグ:	the-project-plan

注記: LifeKeeper の GUI を使用して定義の大部分を作成できますが、この例の以降ではコマンドインターフェースの操作を説明します。

3. **ディレクトリの作成。** 各システムで以下のコマンドを使用して、ディレクトリ `/opt/LifeKeeper/subsys` の下に必要なアプリケーションリカバリディレクトリを作成します。

```
mkdir -p /opt/LifeKeeper/subsys/projectapp/Resources/plan/actions
```

4. **アプリケーションの定義。** 以下のコマンドで、アプリケーション `projectapp` を作成します。

```
app_create -d Server1 -a projectapp
```

```
app_create -d Server2 -a projectapp
```

5. **リソースタイプの定義。** 以下のコマンドで、リソースタイプ `plan` を作成します。

```
typ_create -d Server1 -a projectapp -r plan
```

```
typ_create -d Server2 -a projectapp -r plan
```

6. **リカバリスクリプトのインストール。** `restore` と `remove` のスクリプトを各サーバの以下のディレクトリにコピーします。

```
/opt/LifeKeeper/subsys/projectapp/Resources/plan/actions
```

7. **インスタンスの定義。** 以下のコマンドで、リソースのインスタンスタイプが `plan`、ID が `1yrplan` のリソースを定義します。

```
ins_create -d Server1 -a projectapp -r plan -I\
```

```
AUTORES_ISP -t the-project-plan -i 1yrplan
```

```
ins_create -d Server2 -a projectapp -r plan -I\
```

```
SEC_ISP -t the-project-plan -i 1yrplan
```

Server1 に作成したインスタンスの `-I AUTORES_ISP` 命令は、LifeKeeper の再起動時にそのリソースを自動的に `in service` にするように LifeKeeper に指示します。この例では、リソースの `restore` スクリプトが実行され、正常に実行された場合はリソースが `ISP` 状態になります。この動作は、ペアのリソースがすでにサービス起動している場合は実行されません。

Server2 に作成したインスタンスの `-I SEC_ISP` 命令は、LifeKeeper の再起動時にそのリソースを `in service` にしないように LifeKeeper に指示します。その代わりに、Server2 は Server1 上にあるリソースのバックアップとして機能し、プライマリのリソースまたはサーバに障害が発生したときにローカルリソースを `in service` にします。

8. **依存関係の定義。** 以下のコマンドは、アプリケーションとファイルシステムの依存関係を定義します。

```
dep_create -d Server1 -p the-project-plan -c project-data-on-System1
```

```
dep_create -d Server2 -p the-project-plan -c project-data-from-Server1
```

```
dep_create -d Server1 -p the-project-plan -c schedule-on-Server1
```

```
dep_create -d Server2 -p the-project-plan -cschedule-from-Server1
```

9. **lcdsync の実行**。以下の `lcdsync` コマンドを実行して、設定のコピーを更新するように LifeKeeper に通知します。

```
lcdsync -d Server1
```

```
lcdsync -d Server2
```

10. **リソースを In Service にする**。プライマリサーバで LifeKeeper の GUI にアクセスし、[Edit] > **[Resource]** > **[In-Service]** をクリックしてリソースを In Service にします。

LCD の設定データ

LCD には、以下の関連するデータタイプが保存されます。

- 依存関係の情報
- リソースのステータス情報
- サーバ間のイクイバレンシ情報

依存関係の情報

定義した各リソースについて、LifeKeeper は依存関係のリスト、および依存物 (あるリソースに依存するリソース) のリストを保持します。詳細については、`LCDI_relationship (1M)` と `LCDI_instances (1M)` のマニュアルページを参照してください。

リソースのステータス情報

LifeKeeper は、各リソースインスタンスのステータス情報をメモリに保持します。LCD が認識する [リソースの状態](#) は、**ISP**、**ISU**、**OSF**、**OSU**、および **ILLSTATE** です。システムイベントが発生した場合、または管理者が特定の操作を行った場合に、リソースがある状態から別の状態に変化することがあります。リソースの状態が変化した場合、ステータスの変化が、ローカルサーバの LCD、およびそのリソースのダイアログサーバ上にあるデータベースに反映されます。

サーバ間のイクイバレンシ情報

さまざまなサーバ上にある複数のリソース間に関係が存在することがあります。[イクイバレンシ情報](#) とは、別のサーバ上にある 2 つのリソースが同一の物理エンティティであることを示す関係です。2 台のサーバが、イクイバレンシ情報の関係にある 1 つのリソースを持つ場合、LifeKeeper はその動作により、2 台のサーバ上にあるリソースの 1 つのみが同時に In Service、保護 (ISP) になるようにします。両方のサーバでそのリソースインスタンスを Out of Service (**OSU** または **OSF**) にすることができますが、データの整合性の理由から、同時に In Service にできるリソースは 1 つのみです。

SCSI バス上にある複数のディスクが、同等なリソースの一例です。SCSI のロック (または予約) メカニズムにより、任意の時点でディスクデバイスのロックを所有できるのは 1 台のサーバのみです。このロック所有機能により、同時に複数のサーバによる同一ディスクリソースへのアクセスが防止されます。

さらに、階層内の依存関係により、ファイルシステムのようにディスクに依存するリソースはすべて、同時に 1 台のサーバでのみ In Service になります。

LCD のディレクトリ構造

`/opt/LifeKeeper` の下にある主なサブディレクトリを示します。

- **config** - LifeKeeper の設定ファイル。イクイバレンシ情報を含みます。
- **bin** - LifeKeeper の実行可能プログラム。is_recoverable などがあります。詳細については、[障害検出とリカバリのシナリオ](#)を参照してください。
- **subsys** - リソースとタイプ。LifeKeeper は、共有 SCSI ディスクサブシステムのリソースとタイプの定義を scsi で、汎用アプリケーションのメニュー機能を gen で提供します。アプリケーションのインターフェースを定義する場合は、subsys の下にディレクトリを作成してください。
- **events** - 警報イベント。詳細については、[LifeKeeper の警報とリカバリ](#)を参照してください。

`/opt/LifeKeeper` 内の LCD ディレクトリの構造については、[/opt/LifeKeeper 内の LCD ディレクトリの構造](#) のトピックを参照してください。

LCD のリソースタイプ

LCD は共有メモリ、および `/opt/LifeKeeper` ディレクトリの両方に保持されます。[ディレクトリ構造の図](#) に示すように、subsys には、アプリケーションインターフェースの指定に使用できるアプリケーションリソースセットが 2 つあります。

- gen - 汎用アプリケーションとファイルシステムの情報
- scsi - SCSI に固有のリカバリ情報

これらのサブディレクトリについては[リソースのサブディレクトリ](#)を参照してください。

LifeKeeper のフラグ

[ステータスの詳細表示](#) の後部近くに、システムのフラグセットがあります。共通タイプは、プロセスのロックが動作を完了するまで他のプロセスを確実に待機させるために使用する LCD のロックフラグです。LCD のロックの標準フォーマットは以下のとおりです。

```
!action!processID!time!machine:id.
```

一般的な LCD のロックフラグの例を示します。

- **!action!02833!701236710!<servername>:filesys**. ファイルシステム階層を作成すると、このフォーマットでステータス表示にフラグが生成されます。filesys の指定は、他のアプリケーションリソース階層では別のリソースタイプである場合も、一般的なアプリケーションやユーザ定義アプリケーションでは app である場合もあります。
- 他の代表的なフラグとして、**!nofailover!machine** and **shutdown_switchover** があります。**!nofailover!machine** フラグは、LifeKeeper が作成と削除を行う内部の一時フラグで、サーバのフェイルオーバーを制御します。**shutdown_switchover** フラグは、このサーバのシャットダウン方針が

スイッチオーバーに設定されたことを示し、サーバのシャットダウンによりスイッチオーバーが発生します。使用可能なフラグの詳細については、LCDI-flag (1M) を参照してください。

リソースのサブディレクトリ

scsi と **gen** のディレクトリはそれぞれ、リソースのサブディレクトリを持ちます。これらのディレクトリの内容は、LifeKeeper が提供するリソースタイプのリストです。

scsi のリソースタイプ。これらのリソースタイプは、`/opt/LifeKeeper/subsys/scsi/resources` ディレクトリにあります。実際の設定によっては、その他のディレクトリが存在する場合があります。

- **device** - ディスクパーティション、または仮想ディスクデバイス
- **disk** - 物理ディスク、または LUN
- **hostadp** - ホストアダプタ

gen のリソースタイプ。これらのリソースタイプは、`/opt/LifeKeeper/subsys/gen/resources` ディレクトリにあります。

- **filesystem** - ファイルシステム
- **app** - 汎用またはユーザ定義のアプリケーションであり、他のリソースに依存することがある

各リソースタイプのディレクトリには、以下のものが1つ以上あります。

- **インスタンス**。このファイルは、LCD に保存されている、リソースインスタンスに関する恒久的な情報を反映します。このリソースタイプに関連付けられたリソースインスタンスの記述的な情報があります。

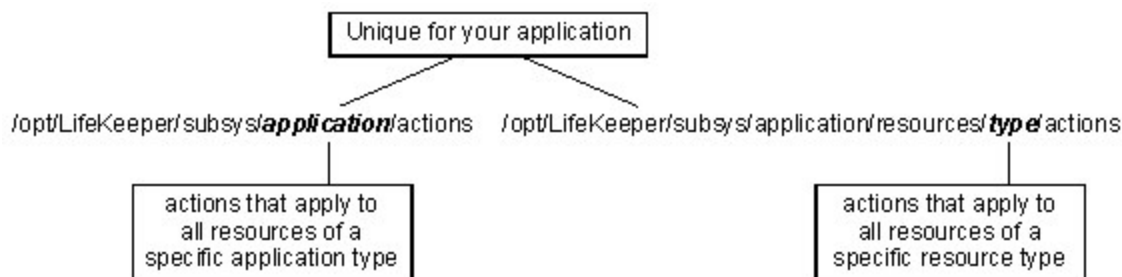
警告: インスタンスファイル(または LCD ファイル)を直接変更しないでください。リソースインスタンスの作成や操作を行うには、LifeKeeper の GUI の機能、または `ins_create`、`ins_remove`、`ins_gettag`、`ins_setas`、`ins_setinfo`、`ins_setinit`、`ins_setstate`、および `ins_list` の LifeKeeper の `LCDI_instances` コマンドのみを使用してください。これらのコマンドの詳細については、`LCDI_instances (1M)` のマニュアルページを参照してください。

- **recovery**。このオプションのディレクトリには、障害が検出されたリソースのローカルリカバリの試行に使用されるプログラムがあります。recovery ディレクトリには、sendevent に渡されるイベントクラスに対応するディレクトリがあります。ディレクトリの名前は、sendevent プログラムに渡されるクラスパラメータ (-C) と一致する必要があります。(LifeKeeper の警告とリカバリを参照)。

各サブディレクトリに、アプリケーションは対応するイベントタイプを処理するリカバリプログラムを入れることができます。これらのプログラムの名前は、sendevent の -E パラメータで渡される文字列と一致する必要があります。このオプションのディレクトリは、複数のアプリケーションに使用されるように存在することはできません。

- **actions**。このディレクトリには、特定のリソースタイプのリソースインスタンスについてのみ動作するリカバリ実行プログラムのセットがあります。使用するアプリケーションについて、アプリケーション内のすべてのリソースに適用する動作がある場合は、その動作を、**resource type** ディレクトリではなく、アプリケーションディレクトリの **actions** サブディレクトリに入れてください。

リソースの動作



リカバリ指示ソフトウェアが、リソースインスタンスの変更や復旧に使用されます。各リソースタイプの **actions** ディレクトリに、**remove** と **restore** の2つの動作が必要です。

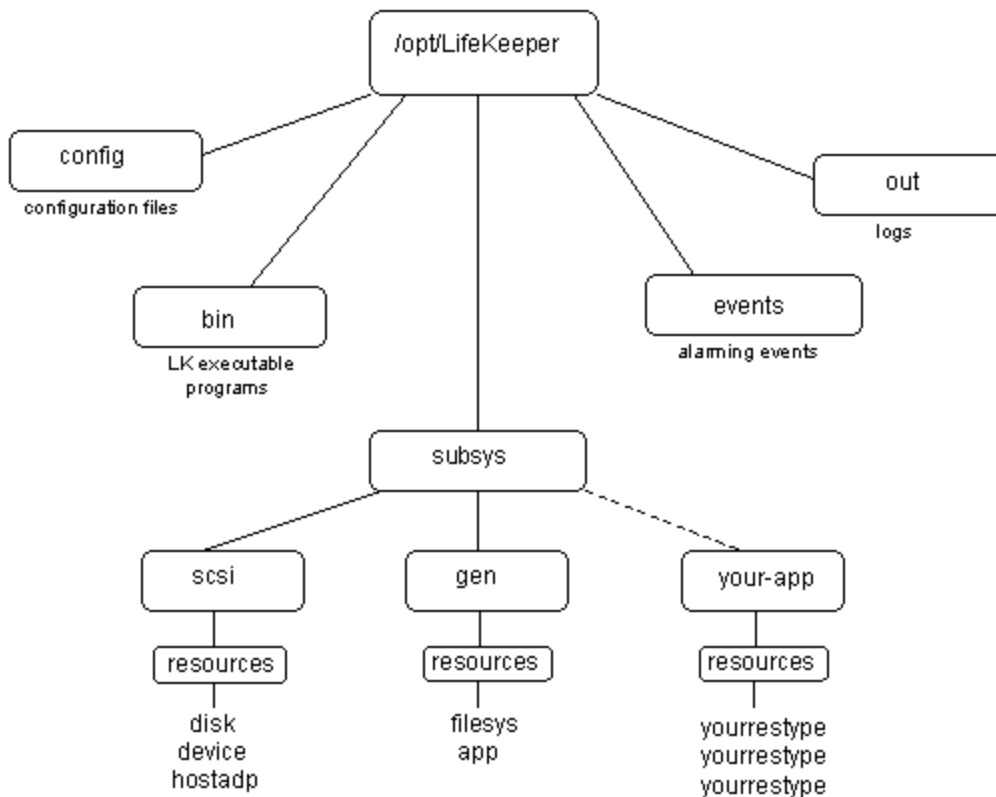
リソースの動作

リソースタイプの **actions** ディレクトリには、特定のアプリケーションの動作を記述するプログラム(多くの場合は shell スクリプト)があります。各リソースタイプについて、**restore** と **remove** の2つの動作が必要です。

remove と **restore** のプログラムは、正反対の機能を実行する必要があります。つまり、相互の動作を元に戻す必要があります。これらのスクリプトは、絶対に手動で実行しないでください。これらのスクリプトは、LifeKeeper のリカバリ動作と制御のインターフェース (LRACI) の `perform_action` shell プログラムのみが実行する必要があります (LRACI-`perform_action` (1M) マニュアルページを参照)。

`/opt/LifeKeeper` の LCD のディレクトリ構造

以下の図に、`/opt/LifeKeeper` のディレクトリ構造を示します。



LCM

The LifeKeeper Communications Manager (LCM) は、1 台以上の LifeKeeper サーバ上にあるプロセス間に信頼性の高い通信を提供します。このプロセスは、システム間の冗長コミュニケーションパスを使用できるので、1 つのコミュニケーションパスに障害が発生しても、LifeKeeper やそれが保護するリソースには障害が発生しません。LCM は、RS-232 (TTY) と TCP/IP の接続を含む多様な通信方法をサポートしています。

LCM は以下の機能を提供します。

- **LifeKeeper のハートビート。** 接続している他の LifeKeeper システムと定期的に通信して、他のシステムが動作を継続しているかどうかを判断します。LifeKeeper はハートビート信号がないことを認識することにより、他の方法では検出されないシステム全体の障害を検出できます。
- **管理サービス。** LifeKeeper の管理機能は、LCM の機能を使用してリモート管理を実行します。この機能は、シングルポイントの管理、設定の検証、および管理動作の正常性チェックに使用されます。
- **設定とステータスの通信。** LifeKeeper 設定データベース (LCD) は、リソースのステータス、可用性、および設定を LCM 機能経由で記録します。LCM の機能により、LCD はプライマリとセカンダリのシステム間で整合性のあるリソース情報を保持できます。
- **フェイルオーバー/リカバリ。** あるシステム上のリソースに障害が発生すると、LCM は LifeKeeper に、バックアップシステム上にリソースを復旧するように通知します。

LCM が提供する LifeKeeper のサービスに加えて、shell コマンドセットによりアプリケーションによる信頼性の高いシステム間通信が可能です。これらのコマンドとして、`snd_msg`、`rcv_msg`、`can_talk` などがあります。これらのコマンドの詳細については、`LCMI_mailboxes (1M)` のマニュアルページを参照してください。LCM はシステム上でリアルタイムプロセスとして動作し、システムのハートビートが送信されるなどの重要な通信が確実に実行されるようにします。

通信ステータスの情報

ステータス表示の通信ステータスの情報のセクションには、LifeKeeper が認識しているサーバとその現在の状態、および各コミュニケーションパスの情報がリストされます。

以下の例は、ステータスの簡略表示の通信ステータスのセクションのものです。

```
MACHINE NETWORK ADDRESSES/DEVICE STATE PRIO
tristan TCP 100.10.100.100/100.10.100.200 ALIVE 1
tristan TTY /dev/ttyS0 ALIVE --
```

詳細については、[ステータスの詳細表示](#)と[ステータスの簡略表示](#)のトピックの「通信ステータスの情報」セクションを参照してください。

LifeKeeper の警報とリカバリ

LifeKeeper のエラー検出と通知は、イベント警報メカニズム `sendevent` をベースにしています。**sendevent** メカニズムの重要な概念は、独立したアプリケーションが重要なコンポーネントについて警報を受信できるように登録できることです。警報を開始する側のコンポーネントと受信する側のアプリケーションのいずれも、他のアプリケーションの存在を知るように変更する必要はありません。アプリケーションに固有のエラーが、**sendevent** 機能経由で LifeKeeper のリカバリメカニズムをトリガできます。

このセクションでは、警報クラス、警報の処理、および警報ディレクトリのレイアウトを含む警報に関連するトピックを説明し、次に警報の概念を示す処理シナリオを示します。

警報クラス

`/opt/LifeKeeper/events` ディレクトリには、アラームクラスのセットがリストされます。これらのクラスは、イベントを生成するシステムの特定制サブコンポーネントに対応します (例: `filesystem`)。各警報クラスのサブディレクトリには、可能性のある警報のセットがあります (例: `badmount`、`diskfull`)。shell スクリプトまたはプログラムを適切なディレクトリに入れることで、これらの警報を受信するようにアプリケーションを登録できます。

LifeKeeper は基本的な警報通知機能を使用しています。この警報機能により、イベントについて登録されたすべてのアプリケーションで、該当する警報の発生時に `sendevent` により処理プログラムが非同期で実行されます。LifeKeeper が存在する場合、**sendevent** プロセスははじめに、LifeKeeper のリソースオブジェクトがクラスとイベントを処理できるかどうかを判断します。LifeKeeper がクラス/イベントの一致を検出した場合、適切な復旧シナリオが実行されます。

sendevent 警報機能の追加スクリプトを定義することは任意です。LifeKeeper リソースを定義すると、LifeKeeper が基本的な警報機能を提供します。その詳細は、この章の処理シナリオで後述します。

注記: リソースインスタンスのローカルリカバリは、LifeKeeper の制御下にあるアプリケーションが、中断されたりソースサービスを、イベントが発生したシステムのエンドユーザに返そうとする試行です。サーバ間リカバリでは、アプリケーションはバックアップシステムに移行できます。この種のリカバリは、ローカルリカバリが失敗したか、ローカルリカバリが不可能である場合に試行されます。

警報の処理

LifeKeeper の注意が必要な可能性のあるイベントを検出するアプリケーションまたはプロセスは、**sendevent** プログラムを実行し、各エラークラス、エラー名、および障害のあるインスタンスの引数を渡すことにより、イベントを報告できます。必須の詳細、オプションのパラメータ、および構文については、sendevent (5) のマニュアルページを参照してください。

警報ディレクトリのレイアウト

`/opt/LifeKeeper/events` ディレクトリには、2種類の内容があります。

- **LifeKeeper の指定クラス。** LifeKeeper は、`events` ディレクトリの下に `lifekeeper` と `filesys` の2つの警報クラスを用意しています。警報イベントの例としては、`diskfull` などがあります。警報クラスは、**sendevent** コマンドの `-C` オプションで渡される文字列に対応し、警報イベントは `-E` オプションで渡される文字列に対応します。Lifekeeper の警報クラスは、LifeKeeper のサブシステム内のイベント報告用に内部的に使用されます。
- **アプリケーションに固有のクラス。** 特定のアプリケーションで警報クラスの定義が必要な場合、`events` ディレクトリに他のサブディレクトリが追加されます。アプリケーションは shell スクリプトまたはバイナリプログラムをそのサブディレクトリに入れることで、これらの警報を受信するように登録します。これらのプログラムの名前は、属するアプリケーションパッケージの名前に由来します。

メンテナンス作業

以下に、LifeKeeper のメンテナンス作業を示します。

LifeKeeper の設定値の変更

LifeKeeper には、設定と設定を行った後に変更を要する値がある値が多数あります。変更を要する値がある値の例として、LifeKeeper サーバの `uname`、コミュニケーションパスの IP アドレス、IP リソースのアドレス、タグ名などがあります。これらの値を変更するには、注意して以下の手順に従ってください。

1. 以下のコマンドを使用して、すべてのサーバで LifeKeeper を停止してください。

```
/etc/init.d/lifekeeper stop-nofailover
```

コミュニケーションパスを削除したり、サーバからリソース階層を拡張解除したりする必要はありません。

2. LifeKeeper サーバの `uname` を変更する場合は、`hostname (1)` コマンドを使用してサーバのホスト名を変更してください。
3. 先に進む前に、新しいホスト名がクラスタ内のすべてのサーバで解決可能であることを確認してください。コミュニケーションパスのアドレスを変更する場合は、新しいアドレスが設定され、動作していることを確認してください (`ping` と `telnet` のユーティリティをこの確認に使用可能)。

4. LifeKeeper の複数の値を変更する必要がある場合は、クラスタ内の各サーバ上のファイルで、古い値と新しい値を以下のフォーマットで指定する必要があります。

```
old_value1=new_value1
```

```
....
```

```
old_value9=new_value9
```

5. クラスタ内のすべてのサーバで `lk_chg_value` コマンドを実行し、出力を確認して、予測しなかった変更内容による副作用が発生していないことを確認してください。変更する値が複数ある場合は、以下のコマンドを実行してください。

```
$LKROOT/bin/lk_chg_value -Mvf file_name
```

`file_name` は、手順 4 で作成したファイルの名前です。

変更する値が1つのみの場合は、以下のコマンドを実行してください。

```
$LKROOT/bin/lk_chg_value -Mvo old_value -n new_value
```

`-M` オプションは、LifeKeeper のすべてのファイルに対して変更を行わないことを指定します。

6. クラスタ内のすべてのサーバで、`-M` オプションを指定せずに `lk_chg_value` コマンドを実行して、LifeKeeper のファイルを変更してください。変更する値が複数ある場合は、以下のコマンドを実行してください。

```
$LKROOT/bin/lk_chg_value -vf file_name
```

`file_name` は、手順 4 で作成したファイルの名前です。

変更する値が1つのみの場合は、以下のコマンドを実行してください。

```
$LKROOT/bin/lk_chg_value -vo old_value -n new_value
```

7. 以下のコマンドを使用して、LifeKeeper を再起動してください。

```
/etc/init.d/lifekeeper start
```

LifeKeeper の GUI を使用してクラスタを表示する場合は、GUI を閉じてから再起動しなければならないことがあります。

例:

`Server1` と `Server2` は、2 ノードクラスタ内にある LifeKeeper サーバの `uname` です。`Server1` は、アドレス 172.17.100.48 のコミュニケーションパスを持ちます。`Server2` はアドレス 172.17.100.220 の IP リソースを持ち、この IP リソースは `Server1` に拡張されています。`Server1` について以下の値を変更します。

値	旧	新
uname	Server1	Newserver1
コミュニケーションパスのアドレス	172.17.100.48	172.17.105.49
IP リソースのアドレス	172.17.100.220	172.17.100.221

これらの変更を行うには、以下の手順を実行する必要があります。

1. 以下のコマンドを使用して、*Server1* と *Server2* の両方で LifeKeeper を停止してください。

```
/etc/init.d/lifekeeper stop-nofailover
```

2. 以下のコマンドを使用して、*Server1* の `uname` を *Newserver1* に変更してください。

```
hostname Newserver1
```

3. *Newserver1* と *Server2* の両方に、以下の内容を持つファイル `/tmp/subs` を作成してください。

```
Server1=Newserver1
```

```
172.17.100.48=172.17.105.49
```

```
172.17.100.220=172.17.100.221
```

4. 両方のサーバで以下のコマンドを実行し、出力を確認して、予測しなかった変更内容による副作用が発生していないことを確認してください。

```
$LKROOT/bin/lk_chg_value -Mvf /tmp/subs
```

5. 両方のサーバで、**-M** オプションを指定せずに `lk_chg_value` コマンドを実行して、LifeKeeper のファイルを変更してください。

```
$LKROOT/bin/lk_chg_value -vf /tmp/subs
```

6. 以下のコマンドを使用して、両方のサーバで LifeKeeper を再起動してください。

```
/etc/init.d/lifekeeper start
```

注記:

- LifeKeeper のファイルを変更せずに `lk_chg_value` による変更内容を表示するには、**-M** オプションを使用してください。`lk_chg_value` が調べるファイルを表示するには、**-v** を使用してください。タグ名を変更しない場合は、**-T** オプションを使用してください。リソース ID を変更しない場合は、**-I** オプションを使用してください。

ファイルシステムの健全性の監視

ファイルシステムの健全性の監視機能は、LifeKeeper が保護する、ファイルシステム依存のアプリケーションで障害が発生する原因となる条件を検出します。監視は、アクティブ / サービス中のリソース (つまりファイルシステム) でのみ実行されます。監視する条件は、以下の2つです。

- ファイルシステムがフル (またはほぼフル) の状態になる。
- ファイルシステムが不適切にマウント (またはアンマウント) された。

これら2つの条件のいずれかが検出されると、いくつかの動作のいずれかが実行されることがあります。

- 警告メッセージがログ記録され、システム管理者に電子メールを送信できる。
- リソースインスタンスのローカルリカバリを試行できる。
- リソースをバックアップサーバにフェイルオーバーできる。

条件の定義

フル(またはほぼフル)のファイルシステム

「ディスクがフル」の条件は検出できますが、ローカルリカバリまたはフェイルオーバーの実行で解決することはできません。管理者の操作が必要です。デフォルトでは、メッセージがログ記録されます。追加の通知機能を使用できます。例えば、電子メールをシステム管理者に送信できます。また、他の方法により、別のアプリケーションを起動して警告メッセージを送信できます。この通知機能を有効にする方法については[LifeKeeper のイベント電子メール通知の設定](#) のトピックを参照してください。

「ディスクフル」の条件に加えて、「ディスクがほぼフル」の条件を検出し、警告メッセージを LifeKeeper のログに記録できます。

「ディスクフル」のしきい値は以下のとおりです。

```
FILESYSFULLERROR=95
```

「ディスクがほぼフル」のしきい値は以下のとおりです。

```
FILESYSFULLWARN=90
```

デフォルト値は上記のとおりそれぞれ 90% と 95% ですが、`/etc/default/LifeKeeper` ファイルの調整可能なパラメータを使用して設定できます。これら 2 つのしきい値の意味は以下のとおりです。

`FILESYSFULLWARNING` - ファイルシステムがこの割合までフルになると、メッセージが LifeKeeper のログに表示されます。

`FILESYSFULLERROR` - ファイルシステムがこの割合までフルになると、メッセージが LifeKeeper のログ、およびシステムログに表示されます。ファイルシステムの通知スクリプトも呼び出されます。

アンマウントされた、または不適切にマウントされたファイルシステム

LifeKeeper は `/etc/mtab` ファイルをチェックして、LifeKeeper が保護するサービス中のファイルシステムが実際にマウントされているかどうかを調べます。さらに、`filesys` のリソース情報フィールドに保存されているマウントオプションに対してマウントオプションをチェックし、階層の作成時に使用されていた元のマウントポジションと一致するかどうかを確認します。

ファイルシステムがアンマウントされているか、不適切にマウントされていることを検出した場合、ローカルリカバリが起動され、正しいマウントオプションを使用してファイルシステムの再マウントが試行されます。

再マウントに失敗した場合、条件を解消するためにフェイルオーバーが試行されます。以下のリストに、フェイルオーバーに進行する場合がある再マウントの障害の一般的な原因を示します。

- ファイルシステムが破損している (fsck の障害)
- マウントポイントディレクトリの作成失敗
- マウントポイントがビジー
- マウントの失敗
- LifeKeeper の内部エラー

LifeKeeper が保護するシステムのメンテナンス

LifeKeeper が保護するサーバをシャットダウンしてメンテナンスを行うときには、メンテナンスの前に、バックアップサーバでシステムのリソース階層を in service にする必要があります。このプロセスにより、メンテナンスが必要なシステム上にある共有ディスクの動作がすべて停止します。

記載の順序で、以下の操作を実行してください。Server A はメンテナンスが必要なプライマリシステム、Server B はバックアップサーバです。

1. **Server B で階層を in service にしてください。**バックアップの Server B で、LifeKeeper の GUI を使用して、現在 Server A でサービス中のリソース階層を in service にします。これにより、LifeKeeper の保護下にある共有ディスクに存在している Server A のファイルシステムがアンマウントされます。詳細については、[リソースを In Service にする](#)を参照してください。
2. **Server A で LifeKeeper を停止してください。**LifeKeeper のコマンド `/etc/init.d/lifekeeper stop-nofailover` を使用して、LifeKeeper を停止します。リソースが保護されていない状態になります。
3. **Linux をシャットダウンし、Server A の電源をオフにしてください。**Server A の Linux オペレーティングシステムをシャットダウンし、サーバの電源をオフにします。
4. **メンテナンスを実行してください。**Server A で必要なメンテナンスを実行します。
5. **Server A の電源をオンにし、Linux を再起動してください。**Server A の電源をオンにし、次に Linux オペレーティングシステムを再起動します。
6. **Server A で LifeKeeper を開始してください。**LifeKeeper のコマンド `/etc/init.d/lifekeeper start` を使用して、LifeKeeper を開始します。リソースが保護されている状態になります。
7. **必要に応じて、Server A で階層を in-service にしてください。**Server A で LifeKeeper の GUI を使用して、Server B にスイッチオーバーしていたすべてのリソース階層を in service にしてください。

リソース階層のメンテナンス

システム上のその他すべての階層を LifeKeeper で保護した状態で、あるリソース階層のメンテナンスを実行できます。このためには、メンテナンスが必要な階層を Out of Service にし、メンテナンス作業の完了後にその階層を In Service にします。

リソース階層のメンテナンスを実行するには、以下の手順に従ってください。

1. **階層を Out of Service にしてください。**LifeKeeper の GUI を使用して、メンテナンスを実行する必要があるリソース階層をすべて Out of Service にします。詳細については、[リソースを Out of Service にする](#)を参照してください。
2. **メンテナンスを実行してください。**リソース階層で必要なメンテナンスを実行します。
3. **階層をリストアしてください。**LifeKeeper の GUI を使用して、リソース階層を In Service にします。詳細については、[リソースを In Service にする](#)を参照してください。

フェイルオーバー後の復旧

LifeKeeper がプライマリサーバ (Server A) からバックアップサーバ (Server B) にフェイルオーバーリカバリを実行した

後、以下の手順を実行してください。

1. **ログを確認してください。** Server B の LifeKeeper が Server A からフェイルオーバーリカバリを実行すると、フェイルオーバー中にステータスメッセージが表示されます。

実際の出力は、設定によって異なります。マウントやアンマウントの失敗に関するいくつかのメッセージが表示されることが予測されますが、これらのメッセージはリカバリの失敗を示唆しません。これらのメッセージ、および Server B でリソースを in service にするときに発生したエラーは、LifeKeeper のログに記録されます。
2. **メンテナンスを実行してください。** Server A の障害の原因を特定し、解決します。メンテナンスを実行するために、Server A の電源をオフにすることが必要な場合があります。
3. **必要に応じて、Server A を再起動してください。** メンテナンスが完了したら、必要に応じて Server A を再起動します。
4. **必要に応じて、LifeKeeper を開始してください。** Server A で LifeKeeper が動作していない場合は、コマンド `/etc/init.d/lifekeeper start` を使用して、LifeKeeper を開始してください。
5. **アプリケーションを Server A に戻してください。** 都合のよい時点で、LifeKeeper の GUI を使用して、Server A でアプリケーションを in service にします。詳細については、[リソースを In Service にする](#)を参照してください。Server A でアプリケーションが **[Automatic Switchback]** に設定されている場合は、この手順は不要なことがあります。

LifeKeeper の削除

Linux 環境での LifeKeeper パッケージのアンインストールは、rpm をサポートするグラフィカルインターフェース、またはコマンドラインから実行できます。このセクションでは、コマンドラインから rpm コマンドを使用して LifeKeeper をアンインストールする手順を詳しく説明します。rpm コマンドを使用する手順の詳細については、[rpm\(8\)](#) のマニュアルページを参照してください。

rpm ソフトウェアの詳細については、以下の Web サイトを参照してください。<http://www.rpm.org/>

以下に、LifeKeeper ソフトウェアを削除するための要件を示します。

- **アプリケーションの移動。** LifeKeeper ソフトウェアを削除する前に、サーバ上に LifeKeeper の保護を必要とするアプリケーションがないことを確認する必要があります。アプリケーションリソース階層が In Service のサーバからは、絶対に LifeKeeper を削除しないでください。LifeKeeper を削除すると、同等性、リソース階層定義、ログファイルなどの設定データがすべて削除されます。追加情報については、[リソース階層の転送](#)を参照してください。
- **LifeKeeper の開始。** LifeKeeper のリカバリキットソフトウェアを削除するときには、LifeKeeper が実行中でなければなりません場合があります。LifeKeeper が実行中でない場合、削除プロセスはクラスタ内の他の LifeKeeper サーバからリソースインスタンスを削除できず、複数のサーバが不整合の状態になることがあります。
- **すべてのパッケージの削除。** LifeKeeper Core を削除する場合、初めに LifeKeeper に依存する他のパッケージ (例: LifeKeeper のリカバリキット) を削除する必要があります。LifeKeeper のリカバリキットを削除する前に、まず関連するアプリケーションリソース階層を削除することが推奨されます。

注記: LifeKeeper のリカバリキットソフトウェアを削除する前に、まず関連する階層をそのサーバから削除することが推奨されます。この削除は、リソースの拡張解除の設定作業で実行できます。既存の階層の

拡張解除を実行せずに LifeKeeper のリカバリキットパッケージを削除した場合、現在定義され、このリカバリキットにより保護されている該当のリソース階層は、システムから自動的に削除されます。一般的なルールは以下のとおりです。リソース階層が In Service のサーバからは、絶対にリカバリキットを削除しないでください。これにより現在の階層が破壊され、リカバリキットの再インストール時に階層の再作成が必要になります。

GnoRPM からの削除

GnoRPM のウィンドウで、削除する各パッケージのアイコンを右クリックし、ポップアップメニューの **[Uninstall]** をクリックしてください(または、パッケージアイコンを選択して **[Uninstall]** ボタンをクリックできます)。

コマンドラインからの削除

サーバから LifeKeeper を削除するには、`rpm -e <packagename>` コマンドを使用して LifeKeeper のパッケージをすべて削除してください。rpm コマンドを使用する手順の詳細については、**rpm(8)** のマニュアルページを参照してください。例えば、LifeKeeper Core パッケージを削除するには、以下のコマンドを入力します。

```
rpm -e steeleye-1k
```

参考として、LifeKeeper Core パッケージクラスタに含まれるパッケージを示します。

```
steeleye-1k
steeleye-1kGUI
steeleye-1kHLP
steeleye-1kIP
steeleye-1kMAN
steeleye-1kRAW
steeleye-1kCCISS
```

ディストリビューションの有効化パッケージの削除

LifeKeeper パッケージを削除した後、SPS のインストールイメージファイルに含まれる設定スクリプトがインストールしたディストリビューションに固有の有効化パッケージを削除する必要があります。お使いの Linux ディストリビューションにより、パッケージの名前は **steeleye-1k<Linux Distribution>** のようになっています。

```
steeleye-1kRedHat
steeleye-1kSuSE
```

ファイアウォールを使用した状態での LifeKeeper の実行

以下のネットワークアクセス要件を満たす場合、LifeKeeper for Linux は、同一サーバ上にファイアウォールを設定した状態で実行できます。

注記: ファイアウォールを単に無効にする場合は、後述の[ファイアウォールの無効化](#)を参照してください。

LifeKeeper のコミュニケーションパス

コミュニケーションパスは、特定の IP アドレスを使用して、LifeKeeper クラスタ内にあるサーバペアの間に設定されます。TCP ポート 7365 は、作成時にデフォルトで各通信のリモート側により使用されますが、通信の開始側の TCP ポートは任意です。推奨方法は、そのシステムが既知のコミュニケーションパスでローカルとリモートの IP アドレスの各指定ペアについて、受信と送信の両方のトラフィックを許可するように、各 LifeKeeper サーバにファイアウォールを設定することです。

LifeKeeper GUI の接続

LifeKeeper GUI は、デフォルトの初期接続ポートであるポート 81 と 82 を含めて、特定の TCP ポートを多数使用します。また、LifeKeeper GUI は、ポート 1024 以降をオブジェクトの送受信に使用するリモートメソッド呼び出し (RMI) も使用します。これらすべてのポートが、各 LifeKeeper サーバのファイアウォールで、少なくとも GUI クライアントが動作する外部システムに対して開いている必要があります。

LifeKeeper の IP アドレスリソース

IP アドレスに関連するアプリケーションにアクセスする必要があるクライアントシステムから、LifeKeeper の階層にある IP アドレスリソースにアクセスできるように、ファイアウォールを設定する必要があります。IP アドレスリソースは LifeKeeper クラスタ内のあるサーバから別のサーバに移動できるので、すべての LifeKeeper サーバ上のファイアウォールを適切に設定する必要があります。

また、LifeKeeper は、ブロードキャスト ping のテストを使用して、IP アドレスリソースの健全性を定期的にチェックします。このテストでは、仮想 IP アドレスからブロードキャスト ping パケットを送信し、ローカルサブネット上の他のいずれかのシステムが最初に応答するまで待ちます。このテストが失敗しないようにするには、各 LifeKeeper サーバ上のファイアウォールが以下のタイプのネットワーク動作を許可するように設定する必要があります。

- 仮想 IP アドレスからのインターネット制御メッセージプロトコル (ICMP) パケットの送信 (アクティブな LifeKeeper サーバがブロードキャスト ping を送信できる)
- 仮想 IP アドレスからの ICMP パケットの受信 (他の LifeKeeper サーバがブロードキャスト ping を受信できる)
- 任意のローカルアドレスからの ICMP 応答パケットの送信 (他の LifeKeeper サーバがブロードキャスト ping に応答できる)
- 仮想 IP アドレスでの ICMP 応答パケットの受信 (アクティブな LifeKeeper サーバがブロードキャスト ping への応答を受信できる)

LifeKeeper Data Replication

LifeKeeper Data Replication を使用する場合は、複製に ndb を使用する任意のポートへのアクセスを許可するように、ファイアウォールを設定する必要があります。ndb が使用するポートは、以下の式で計算できます。

```
10001 + <mirror number> + <256 * i>
```

i は 0 から始まり、使用されていないポート番号が計算されるまで加算されます。`/etc/services` に定義されているポート、`netstat -an -inet` の出力に含まれるポート、または LifeKeeper Data Replication の他のリソースが使用中としてすでに定義されているポートは、使用中です。

例: LifeKeeper Data Replication リソースのミラー番号が 0 である場合、式は当初、使用するポートを 10001 として計算しますが、この番号は、一部の Linux ディストリビューションでは SCP 設定ポートとして `/etc/services` に定義されています。この場合、 i が 1 だけ増分されてポート番号 10257 が得られます。この番号は、これらの Linux ディストリビューションの `/etc/services` には定義されていません。

ファイアウォールの無効化

ファイアウォールを無効にする場合は、以下の手順に従ってください。

1. 以下のコマンド (お使いのファイアウォールパッケージによって異なる) を使用して、ファイアウォールを停止してください。

```
/etc/init.d/ipchains stop または
```

```
/etc/init.d/iptables stop
```

IPv6 環境を使用している場合は、かならず `ip6tables` を考慮してください。

```
/etc/init.d/ip6tables stop
```

SuSE Linux Enterprise Server を実行している場合

```
/etc/init.d/SuSEfirewall2_init stop
```

```
/etc/init.d/SuSEfirewall2_setup stop
```

2. パッケージを削除するか (`rpm -e` を使用)、以下のいずれかのコマンド (お使いのファイアウォールパッケージによって異なる) を使用して起動を無効にしてください。

```
/sbin/chkconfig --del ipchains または
```

```
/sbin/chkconfig --del iptables
```

```
/sbin/chkconfig --del ip6tables
```

SuSE Linux Enterprise Server を実行している場合は、`SuSEfirewall2` の設定を管理する必要があります。

ファイアウォール経由での LifeKeeper GUI の実行

場合によっては、LifeKeeper クラスタが会社のファイアウォール内に配置され、管理者はファイアウォールの外側にあるリモートシステムから LifeKeeper GUI を実行します。

LifeKeeper は、GUI のサーバとクライアントとの通信にリモートメソッド呼び出し (RMI) を使用します。RMI クライアントは、それぞれの方向に通信を確立する必要があります。RMI クライアントは動的ポートを使用するので、クライアントには推奨ポートを使用できません。

解決法としては、以下のように `ssh` を使用して、ファイアウォールを通過する方法があります。

リソース階層の転送

1. 社内の IT 部門が、ファイアウォール内にアクセスするために十分にセキュリティの高い shell ポートを社内ファイアウォールに開けていることを確認します。多くの場合、IT 部門がアクセスを許可するマシンは、実際にはクラスタ内のマシンではなく、そこからクラスタ内にアクセスできる中間マシンです。このマシンは、Unix または Linux が動作するマシンである必要があります。
2. 中間マシンと LifeKeeper サーバの両方が、sshd (Secure Shell デーモン) を実行していること、および X11 ポート転送が有効になっていること(これは通常、`etc/ssh/sshd_config` の `X11Forwarding yes` 行にある)を確認してください。不明の場合は、IT 部門に依頼してください。
3. X の Unix クライアントから以下のコマンドを使用して、中間マシンにトンネルを作成します。

```
ssh -X -C <intermediate machine>
```

`-C` は「トラフィックの圧縮」を意味し、低速のインターネットリンクから受信する場合に役立つことが多々あります。

4. 中間マシンから以下のコマンドを使用して、LifeKeeper サーバにトンネルを作成します。

```
ssh -X <LifeKeeper server>
```

中間マシンは LifeKeeper サーバとの間にかなり高い帯域幅の接続をもつはずなので、このコマンドには圧縮は不要です。

5. すべての操作が良好に完了した場合、以下のコマンドを実行してください。

```
echo $DISPLAY
```

「`localhost:10.0`」のような値に設定されます。値が設定されない場合、X11 の転送がいずれかの sshd 設定ファイルで無効になっています。

6. 以下のコマンドを実行して、LifeKeeper サーバから単純な `xterm` をポップアップ表示できることを確認してください。

```
/usr/X11R6/bin/xterm
```

7. `xterm` が表示された場合、以下のコマンドを使用して、LifeKeeper で `lkGUIapp` を実行できます。

```
/opt/LifeKeeper/bin/lkGUIapp
```

8. GUI コンソールが表示されるまで待ってください。Java は多くのグラフィックス動作を使用し、低速リンクで伝播するには時間がかかります(圧縮している場合でも)。しかし、最終的には GUI コンソールが表示されます。

リソース階層の転送

LifeKeeper サーバで定期的なメンテナンスやその他の作業を実行する必要がある場合、LifeKeeper の GUI を使用して In Service のリソースを別のサーバに移動できます。サーバ A からサーバ B に In Service のリソース階層を転送するには、GUI を使用してサーバ B でリソース階層を in service にします。サーバ A のリソースがすべて、対応するバックアップサービスで In Service になるまで、操作を繰り返します。手順については、[リソースを In Service にする](#)を参照してください。

サーバ A のリソースがすべて、バックアップサーバでアクティブになった後、アプリケーションの処理に影響を与えることなく、サーバ A をシャットダウンできます。ただし、メンテナンスの期間中、クラスタ内にあるサーバ数によっては、リソースが LifeKeeper で保護されないことがあります。

テクニカルノート

お使いの LifeKeeper 環境に関する設定と動作上の問題に関する以下のテクニカルノートをお読みになることを強く推奨します。

LifeKeeper の機能

項目	説明
ライセンス	LifeKeeper を使用するには、各サーバに一意の実行時ライセンスキーが必要です。これは物理サーバおよび仮想サーバの両方に適用されます。ライセンスキーは、LifeKeeper Core ソフトウェア、および LifeKeeper リカバリキットの各パッケージにそれぞれ必要です。インストールスクリプトで、サーバの Host ID を取得して表示するライセンスユーティリティパッケージをインストールします。ライセンスがインストールされた後、ユーティリティが、使用可能な Entitlement ID、または Host ID (Entitlement ID が使用できない場合) を返します。Host ID およびソフトウェアに付属のアクティベーション ID を使用して SIOS Technology Corp. の Web サイト からライセンスキーを取得してください。
大型クラスタのサポート	LifeKeeper は、最大 32 台のサーバを持つ大型クラスタの設定をサポートします。ただし、LifeKeeper 以外の多くの要因が、クラスタ内でサポートされるサーバの台数に影響することがあります。この要因として、ストレージの相互接続、オペレーティングシステム、ストレージソフトウェアの制限などがあります。サポートされる最大クラスタサイズを調べるには、ベンダ固有のハードウェアとソフトウェアの設定情報を参照してください。
国際化とローカライズ	LifeKeeper for Linux v5.2 以降は、リソース名とタグ名でのワイド / マルチバイト文字の使用をサポートしていますが、ネイティブの言語メッセージサポートは含まれていません。Java のプロパティファイルのロケール固有バージョンを作成することにより、LifeKeeper の GUI をローカライズできますが、現在フルにローカライズされているのは英語バージョンのみです。ただし、LifeKeeper の GUI に表示される多くのメッセージは LifeKeeper Core から来ているので、GUI のローカライズは、ユーザにとって、Core ソフトウェアがフルにローカライズされるまでの単なる部分的な解決法です。 追加情報については、 制限または既知の問題の「言語環境の影響」 も参照してください。
LifeKeeper の MIB ファイル	LifeKeeper は、LifeKeeper クラスタ内で発生するイベントを記述する SNMP トラップを送出するように設定できます。この機能の設定に関する詳細については、lk_configsnmp(8) のマニュアルページを参照してください。LifeKeeper のトラップを記述する MIB ファイルは、 <code>/opt/LifeKeeper/include/LifeKeeper-MIB.txt</code> に記載されています。
Watchdog	LifeKeeper は、Watchdog 機能をサポートしています。この機能は、SIOS Technology Corp. により Red Hat EL 5.5 の 64 ビット、および Red Hat EL 6 + softdog でテスト済みです。
STONITH	LifeKeeper は、STONITH 機能をサポートしています。この機能は、SIOS Technology Corp. により IBM x3550 x86_64 アーキテクチャ上の SLES 11、および RHEL5.5 の 64 ビットでテスト済みです。

XFS ファイルシステム	XFS ファイルシステムは、ファイルシステムのチェックと修正に fsck ユーティリティを使用しません。その代わりに、ログの再生をマウントに依存します。整合性の問題についての懸念がある場合は、システム管理者がファイルシステムを out of service にしてシステムをアンマウントし、xfs_check(8) と xfs_repair(8) を実行して問題を解決する必要があります。
IPv6	SIOS は、ip コマンドの使用に移行し、ifconfig コマンドを使用しなくなりました (詳細については IPv6 の既知の問題 を参照)。

チューニング

項目	説明												
IPC セマフォと IPC 共有メモリ	<p>LifeKeeper には、プロセス間通信 (IPC) セマフォと IPC 共有メモリが必要です。以下の Linux カーネルオプションの Red Hat のデフォルト値は、<code>/usr/src/linux/include/linux/sem.h</code> にあり、LifeKeeper の多数の設定をサポートするのに十分な値です。</p> <table border="1" data-bbox="422 777 966 976"> <thead> <tr> <th>必要なオプション</th> <th>Red Hat 6.2 のデフォルト値</th> </tr> </thead> <tbody> <tr> <td>SEMOPM</td> <td>14 32</td> </tr> <tr> <td>SEMUME</td> <td>20 32</td> </tr> <tr> <td>SEMMNU</td> <td>60 32000</td> </tr> <tr> <td>SEMMAP</td> <td>25 32000</td> </tr> <tr> <td>SEMMNI</td> <td>25 128</td> </tr> </tbody> </table>	必要なオプション	Red Hat 6.2 のデフォルト値	SEMOPM	14 32	SEMUME	20 32	SEMMNU	60 32000	SEMMAP	25 32000	SEMMNI	25 128
必要なオプション	Red Hat 6.2 のデフォルト値												
SEMOPM	14 32												
SEMUME	20 32												
SEMMNU	60 32000												
SEMMAP	25 32000												
SEMMNI	25 128												
システムファイルテーブル	<p>LifeKeeper がバックアップシステムに正常にフェイルオーバーするためには、システムリソースが使用可能である必要があります。例えば、システムファイルテーブルがフルの場合、LifeKeeper が新しいプロセスを開始してリカバリを実行することができない可能性があります。エンタプライズパッチを持つカーネル (LifeKeeper がサポートするものを含む) では、file-max、つまりシステムで開いているファイルの最大数は、デフォルトでシステムメモリサイズの 1/10 に設定されます。これは、LifeKeeper の多数の設定をサポートするのに十分な値です。file-max 値をデフォルト値よりも低く設定すると、予期しない LifeKeeper の障害が発生することがあります。</p> <p>file-max 値は、以下のコマンドで取得できます。</p> <pre>cat /proc/sys/fs/file-nr</pre> <p>このコマンドは、3つの値を返します。1番目の値はファイルテーブルのエントリのこれまでの最大値 (システムがこれまでに検出した最大値)、2番目の値は現在のファイルテーブルのエントリ数、3番目の値は file-max 値です。</p> <p>file-max を調整するには、<code>/etc/sysctl.conf</code> の「<code>fs, file-max</code>」値を追加 (または変更) し (フォーマットについては <code>sysctl.conf(5)</code> を参照)、</p> <pre>sysctl -p</pre> <p>次にこのファイルを実行して、システムを更新します。<code>/etc/sysctl.conf</code> の値は、再起動後も保持されません。</p>												

LifeKeeper の動作

項目	説明
カーネルデバッグ (kdb) init s	LifeKeeper が保護するサーバでカーネルデバッグ (kdb) を使用したり init s に移動する前に、そのサーバで LifeKeeper をシャットダウンするか、LifeKeeper が保護するリソースをバックアップサーバにスイッチオーバーする必要があります。LifeKeeper の SCSI 予約デーモン (lkscsid と lkccissd) を有効にした状態で (デフォルトで有効)、 kdb を使用すると、予期しないパニックが発生することがあります。
ロックしている共有デバイスでのシステムパニック	LifeKeeper はロックを使用して、共有 SCSI バス上にある他のサーバがアクセスしないように共有データを保護します。他のサーバがデバイスをロックしたことにより LifeKeeper がデバイスにアクセスできない場合、致命的なエラーが発生し、即座に対処する必要があります。対処しない場合、データが破損するおそれがあります。この条件が検出された場合、LifeKeeper はシステムにパニックを発生させる機能を有効にします。 共有デバイスが予約された状態で、LifeKeeper が「 <code>/etc/init.d/lifekeeper stop-nofailover</code> 」以外の方法により停止した場合 (init s の実行で発生することがある)、他のサーバがリソースを復旧するときに LifeKeeper のロックメカニズムによりカーネルのパニックがトリガされることがあります。この方法で LifeKeeper を停止する前に、リソースをすべて out-of-service にする必要があります。
nolock オプション	When using storage applications with locking and following recommendations for the NFS mount options, SPS requires the additional <code>nolock</code> option be set, e.g. <code>rw,nolock,bg,hard,nointr,tcp,nfsvers=3,timeo=600,rsz=32768,wsz=32768,actimeo=0.</code>
Out-of-Service 階層の復旧	LifeKeeper サーバの障害発生後のリカバリの一部として、障害が発生したサーバに設定されているリソース階層のうち、障害発生時にいずれかのサーバで <i>in-service</i> ではないものは、その時点で優先順位が最高の <i>alive</i> のサーバで復旧されます。これは、障害が発生したサーバ、復旧中のサーバ、クラスタ内の他のサーバを含め、 <i>out of service</i> の階層が最後にどこで <i>in service</i> だったかには無関係です。

サーバの設定

Linux ファイア ウォールと SELinux の 共存	<p>ファイアウォールとSELinux が、インストール時に有効になります。インストールの完了後、SELinux を無効にし、ファイアウォールを変更する必要があります。</p> <p>SELinux のモードが「有効」または「許可」の場合、LifeKeeper はインストールされず、機能しません。Red Hat のSELinux を無効にするには、ホストシステムのコンソールから <code>system-config-securitylevel-tui</code> ツールを実行してください。SELinux for SLES 11 SP1 が提供されていますが、これも無効にする必要があります(http://www.novell.com/linux/releasen.../SUSE-SLES/11/)。</p> <p>AppArmor (このセキュリティモデルを使用するディストリビューションの場合) は有効にすることができます。</p> <p>ホストのファイアウォールが有効の場合、LifeKeeper は機能します。ただし、絶対に必要な場合以外は、ファイアウォールは無効にし、LifeKeeper が保護するリソースは別の保護ファイアウォール内に配置してください。</p> <p>LifeKeeper を、ファイアウォールを有効にしたホストと共存させる必要がある場合、LifeKeeper はコミュニケーションパス、GUI、IP、およびデータ複製に特定のポートを使用します。Linux のファイアウォール機能を使用する場合、LifeKeeper が使用する特定のポートを開く必要があります。</p> <p>Red Hat のファイアウォールを無効にしたり変更したりするには、ホストシステムのコンソールから <code>system-config-securitylevel-tui</code> ツールを実行してください。SUSE のファイアウォールを無効にしたり変更したりするには、<code>yast2</code> を実行し、[Security and User]、[Firewall] を順に選択してください。詳細については、ファイアウォールを使用した状態での LifeKeeper の実行 を参照してください。</p>
Suid マ ウントオ プション	<p>suid マウントオプションは、<code>root</code> としてマウントするときのデフォルトであり、マウントコマンドにより <code>/etc/mtab</code> に書き込まれることはありません。LifeKeeper 環境では、suid マウントオプションは不要です。</p>

サーバの設定

項目	説明
BIOS のアップデート	使用可能な最新の BIOS を常にすべての LifeKeeper サーバにインストールする必要があります。

LifeKeeper 8.2.0 以降の GUI 要件

LifeKeeper GUI クライアントでユーザを正常に認証するには、64 ビットバージョンの PAM 関連のパッケージが必要です。

[Confirm Failover] と [Block Resource Failover] の設定

以下の説明、例、および考慮事項をよく読んで理解してから、お使いの LifeKeeper 環境で **[Confirm Failover]** または **[Block Resource Failover]** を設定してください。これらの設定は、コマンドライン、または LifeKeeper の GUI の **[Properties]** パネルから使用できます。

Confirm Failover On:

定義 - システム A からシステム B へのフェイルオーバーの手動確認を有効にします (システム A はプロパティが [\[Properties\]](#) パネルに表示されるサーバで、システム B はチェックボックスの左にあるシステム)。あるシステムでこのオプションをオンに設定した場合、障害発生が検出されたシステムについて LifeKeeper がフェイルオーバーリカバリを実行するには、システム管理者による手動確認が必要になります。

フェイルオーバーを確認するには、`lk_confirmso` コマンドを使用してください。デフォルトでは、このコマンドを実行するまで管理者には 10 分の猶予時間があります。この時間は、`/etc/default/LifeKeeper` の **CONFIRMSOTO** 設定で変更できます。管理者が 10 分以内に `lk_confirmso` コマンドを実行しない場合、フェイルオーバーは続行されるか、ブロックされます。デフォルトでは、フェイルオーバーが続行されます。この動作は、`/etc/default/LifeKeeper` の **CONFIRMSODEF** 設定で変更できます。

例: 自動フェイルオーバーをすべてブロックする場合は、[\[Properties\]](#) パネルの **[Confirm Failover On]** オプションを設定し、さらに **CONFIRMSODEF** を 1 (フェイルオーバーをブロック)、**CONFIRMSOTO** を 0 (フェイルオーバー動作が決定されるまで待機しない) に設定してください。

この設定を選択するタイミング:

この設定は、設定に冗長ハートビートコミュニケーションパスを含まない多くのディザスタリカバリ、その他の WAN 設定で使用されます。

通常のサイト (非マルチサイトクラスタ) では、あるサーバで [\[Properties\]](#) ページを開き、**[Confirm Failover]** フラグをオンに設定するサーバを選択してください。

マルチサイト WAN の構成の場合: フェイルオーバーの手動確認を有効にしてください。

マルチサイト LAN の構成の場合: フェイルオーバーの手動確認を有効にしないでください。

マルチサイトクラスタ環境では、非ディザスタシステムから DR システムを選択し、`[Set Confirm Failover]` フラグチェックボックスをオンにします。クラスタ内の非ディザスタサーバのそれぞれについて、[\[Properties\]](#) パネルを開いてこの設定を選択する必要があります。

Block Resource Failover On:

定義 - デフォルトでは、リソースのすべての障害について復旧イベントが発生し、ローカルシステムの障害リソースの復旧が試行されます。ローカルリカバリが失敗した場合、または有効になっていない場合は、リソースが定義されている、優先順位が次に最も高いシステムに、LifeKeeper がローカル履歴を転送します。ただし、宛先として指定したシステムでこの設定を選択している場合、リソース障害に起因するリソースの転送はすべてブロックされます。

この設定が有効の場合、以下のメッセージがログに記録されます。

Local recovery failure, failover blocked, MANUAL INTERVENTION REQUIRED

条件 / 考慮事項:

マルチサイト設定では、設定内のすべてのサーバについて、フェイルオーバーのブロックを選択しないでください。

注記: この設定は、システム全体の障害が発生した場合のフェイルオーバー動作には影響しません。リソースの障害に起因するフェイルオーバーのみをブロックします。

NFS クライアントのオプション

LifeKeeper で保護する NFS サーバを設定するときには、NFS クライアントがこのサーバに接続する方法が、フェイルオーバー時に再接続する速さに大きな影響を与えます。

NFS クライアントをマウントするときの考慮事項

NFS サーバは、クライアントコンピュータにネットワークベースのストレージシステムを提供します。このリソースを使用するには、クライアントシステムは、NFS サーバによりエクスポートされた、既に NFS であるファイルシステムを「マウント」する必要があります。NFS クライアントを LifeKeeper が保護する NFS リソースに接続する方法について、いくつかのオプションをシステム管理者は考慮する必要があります。

UDP または TCP の選択

NFS プロトコルは、ユーザデータグラムプロトコル (UDP) と伝送制御プロトコル (TCP) のいずれかを活用できます。NFS は従来、クライアント / サーバの通信に UDP プロトコルを使用してきました。この理由の 1 つは、NFS が UDP プロトコルを使用してステートレス方式で動作するほうが容易だからです。この「ステートレス」であることが、高可用性のクラスタ化では重要です。これは、保護されている NFS サーバリソースがクラスタホスト間で切り替えられた場合に、クライアントを容易に認識できるからです。一般的に、LifeKeeper が保護する NFS リソースを操作するときには、UDP プロトコルが TCP よりも良好に動作する傾向があります。

/etc/exports の Sync オプション

LifeKeeper が保護する NFS リソースの場合、エクスポートオプションとして「sync」を指定することが推奨されます。「sync」オプションは、ディスクに書き込みを実行してから肯定応答を NFS クライアントに送信するように NFS に指示します。もう 1 つのオプションである「async」も使用できますが、このオプションを使用するとデータが破損するおそれがあります。これは、ディスクに書き込みを実行する前に NFS 書き込みの肯定応答をクライアントに送信するからです。NFS クライアントも、NFS ファイルシステムのマウント時にオプションとして「sync」を指定できます。

Red Hat EL6 (および Fedora 14) クライアントと Red Hat EL6 NFS サーバの使用

Red Hat EL6 用 NFS サーバのバグと思われるものにより、Red Hat EL6 (および Fedora 14) を実行する NFS クライアントは、NFS のバージョン (*nfsvers*) および UDP の両方をマウントコマンドに指定できません。これと同じ動作が、Ubuntu10.10 クライアントでも確認されています。この動作は、Red Hat EL6 NFS を使用する Red Hat EL5 クライアントでは確認されておらず、Red Hat EL5 NFS サーバを使用するすべてのクライアントで確認されていません。Red Hat EL6 (Fedora 14) クライアントと Red Hat EL 6 NFS サーバを使用するための NFS マウントディレクトリの最善の組み合わせは以下のとおりです。

```
mount <protected-IP>:<export> <mount point>
-o nfsvers=2,sync,hard,intr,timeo=1
```

- この組み合わせでは、LifeKeeper が保護する NFS サーバがスイッチオーバーまたはフェイルオーバーを実行する場合に、クライアントの再接続時間が最短になります。

Red Hat EL5 NFS クライアントと Red Hat EL6 NFS サーバの使用

Red Hat EL5 を実行する NFS クライアントと Red Hat EL6 NFS サーバを使用するときに、再接続時間が短い最善のオプションの組み合わせは以下のとおりです。

```
mount <protected-IP>:<export> <mount point>
-o nfsvers=3, sync, hard, intr, timeo=1, udp
```

クラスタの例

拡張したマルチクラスタの例

Hierarchies	pat	mike	wallace	gromit	batman	bullwinkle
Backup Not StandB						
file_system_2	40 St...	50 St...	1 Acti...	10 St...	60 St...	70 U...
device-nfs180	40 St...	50 St...	1 Acti...	10 St...	60 St...	70 U...

トラブルシューティング

メッセージカタログ (場所は、SIOS テクニカルドキュメンテーション Web サイトの「エラーコードの検索」の下) には、操作、管理、GUI など、SIOS Protection Suite for Linux の使用中に遭遇する可能性のあるすべてのエラーコードが列挙されています。また、エラーコードの原因に関する追加の説明や、問題解決のために必要な処置についても、必要に応じて記載されています。この完全なリストを検索すると、受信したエラーコードを見つけることができます。また、以下の個別のメッセージカタログに直接アクセスすることもできます。

- Core メッセージカタログ
- DB2 メッセージカタログ
- DMMP Kit メッセージカタログ
- Recovery Kit for EC2 メッセージカタログ
- ファイルシステムキットメッセージカタログ
- Gen/App Kit メッセージカタログ
- GUI メッセージカタログ
- IP Kit メッセージカタログ
- Oracle Listener Kit メッセージカタログ
- Oracle Kit メッセージカタログ
- SCSI Kit メッセージカタログ
- DataKeeper Kit メッセージカタログ

上記のメッセージカタログに加え、以下のトピックでも、直面する可能性がある問題や制限事項のトラブルシューティングについて詳細を説明します。

SPS が開始するフェイルオーバーの一般的な原因

障害が発生した場合、SPS には 2 つのリカバリ方式があります。ローカルリカバリとサーバ間リカバリです。ローカルリカバリが失敗した場合、「フェイルオーバー」が実行されます。フェイルオーバーは、アクティブだったアプリケーション、サーバ、システム、ハードウェアコンポーネント、ネットワークの障害や異常終了が発生したときのバックアップサーバへの自動的な切り替えと定義されます。フェイルオーバーとスイッチオーバーは基本的には同じ動作ですが、スイッチオーバーが人の介入を必要とするのに対し、フェイルオーバーは自動で、[通常は警告なしで実行されます](#)。この自動的なフェイルオーバーは、さまざまな理由で発生します。以下は、SPS が開始するフェイルオーバーの最も一般的な例の一覧です。

サーバレベルでの原因

サーバの障害

SPS には、設定内の各サーバに、ペアのサーバが動作していることを定期的に通知する組み込みのハートビート信号があります。サーバがハートビートメッセージを受信しなかった場合に、障害として検出されます。

- プライマリサーバが電源を喪失するか、電源がオフになる。
- 過負荷による CPU 使用率 – 非常に高負荷の I/O の下では、これらの遅延と低メモリ状態によってシステムが無応答になり、SPS がこれをサーバのダウンとして検知し、フェイルオーバを開始することがあります。
- Quorum/Witness - プライマリサーバが Quorum を失った場合、Quorum/Witness の I/O フェンシングメカニズムの一環として、(設定に基づいて)「[fastboot](#)」、「[fastkill](#)」、または「[osu](#)」が実行され、フェイルオーバが開始されます。フェイルオーバ時を決定する際、プライマリサーバで障害が発生しクラスタを構成できなくなったことが確認できた場合にのみ、witness サーバによりバックアップサーバ上でリソースを in service にすることができます。全体のアクセスや、パフォーマンス、in-service のノードが影響を受けない場合は、これによってノード間で発生する単純な通信障害から発生するフェイルオーバを回避します。

関連トピック

[サポートストレージ一覧](#)

[サーバ障害リカバリのシナリオ](#)

[LifeKeeper ハートビートの調整](#)

[Quorum/Witness](#)

通信障害 / ネットワーク障害

SPS は、サーバ間で 5 秒ごとにハートビートを送信します。通信障害によって 2 回のハートビートが途絶し、3 回目のハートビートで再開した場合、SPS は一切処置を行いません。コミュニケーションパスの切断継続時間がハートビート 3 回分になった場合は、SPS はそのコミュニケーションパスを切断と判定します。ただし、他方の冗長的なコミュニケーションパスも切断と判定されるまではフェイルオーバを開始しません。

- プライマリサーバへのネットワーク接続の喪失。
- ネットワークの遅延。
- TCP コミュニケーションパス上のネットワークトラフィックが大きくなると、偽のフェイルオーバや LifeKeeper の初期化の問題など、予期せぬ動作が生じる可能性があります。
- STONITH を使用しているときに SPS がノードとの通信障害を検出すると、そのノードの電源が切断され、フェイルオーバが発生します。
- NIC の障害。
- ネットワークスイッチの障害。
- 手動によるネットワーク接続の解除。

関連トピック
コミュニケーションパスの作成
LifeKeeper ハートビートの調整
ネットワーク設定
ネットワーク設定の確認
SNMP 経由の LifeKeeper イベントの転送
ネットワーク関連のトラブルシューティング (GUI)
ファイアウォールを使用した状態での LifeKeeper の実行
STONITH

スプリットブレイン

単一のコミュニケーションパスを使用しているときに、そのコミュニケーションパスで障害が発生した場合、複数のシステム上で同時に SPS の階層が in service になることがあります。これは、偽のフェイルオーバーまたは「**スプリットブレイン**」シナリオと呼ばれます。「スプリットブレイン」シナリオでは、各サーバが、アプリケーションを制御できると認識しているため、データにアクセスしようしたり共有ストレージデバイスにデータを書き込もうとする場合があります。スプリットブレインシナリオを解決するために、SPS では、サーバの電源をオフにしたり、再起動したり、階層を out-of-service にすることで、すべての共有データに対するデータの整合性を保証することができます。また、TCP コミュニケーションパス上のネットワークトラフィックが大きくなると、偽のフェイルオーバーや LifeKeeper が適切に初期化できなくなるなど、予期しない動作が生じる可能性があります。

以下に、スプリットブレインが発生する可能性のあるシナリオを示します。

- 上記のいずれかの通信障害
- LifeKeeper の不正なシャットダウン
- サーバリソースの枯渇
- すべてのネットワークパスの喪失
- DNS またはその他のネットワークの問題
- システムのロックアップ / 解除

リソースレベルでの原因

SPS には、個々のアプリケーションおよび関連アプリケーションのグループを監視する機能があり、定期的にローカルリカバリを実行したり、保護下のアプリケーションに障害が発生したときに通知することができます。関連し合うアプリケーションの例としては、主アプリケーションが下位のストレージまたはネットワークリソースに依存する階層などがあります。SPS は、これらの保護下のリソースのステータスと健全性を監視します。SPS は、リソースが障害状態にあると判断すると、外部の介入なしに現在のシステム (in-service のノード) でリソースまたはアプリケーションをリストアしようとします。このローカルリカバリが失敗した場合、リソースのフェイルオーバーが開始されます。

アプリケーションの障害

- アプリケーションの障害が検出されたが、ローカルリカバリプロセスが失敗。
- 削除の失敗 - リソースのフェイルオーバープロセスでは、該当する重要なアプリケーションのすべての機能を提供するために、プライマリサーバ上のサービスから特定のリソースを削除し、選択したバックアップサーバでそのリソースを in service にする必要があります。この削除プロセスが失敗した場合は、プライマリサーバの再起動が実行され、その結果、完全なサーバフェイルオーバーに移行します。

削除の失敗の例

- ファイルシステムをアンマウントできない
- 保護対象のアプリケーション (Oracle、mySQL、Postgres など) をシャットダウンできない

関連トピック

[ファイルシステムの健全性監視](#)

[リソースエラーリカバリのシナリオ](#)

ファイルシステム

- ディスクフル - SPS のファイルシステムの健全性監視は、ファイルシステムリソースのフェイルオーバーになる可能性のある、ファイルシステムのディスクフル状態を検出できます。
- アンマウントされた、または不適切にマウントされたファイルシステム - in-service 状態で LK 保護下のファイルシステムをユーザが手動でアンマウントまたはオプションを変更。
- 再マウントの障害 - 以下のリストに、フェイルオーバーに進行する可能性がある再マウントの障害の一般的な原因を示します。
 - ファイルシステムが破損している (fsck の障害)
 - マウントポイントディレクトリの作成失敗
 - マウントポイントがビジー
 - マウントの失敗
 - SPS の内部エラー

関連トピック

[ファイルシステムの健全性監視](#)

IP アドレスの障害

IP Recovery Kit によって IP アドレスの障害が検出されると、結果として生じる障害によって IP ローカルリカバリスクリプトが起動されます。SPS は最初に、その IP アドレスを現在のネットワークインターフェース上で in service

に戻そうとします。ローカルリカバリを試みが失敗すると、SPS はその IP アドレスと依存関係を持つすべてのリソースのバックアップサーバへのフェイルオーバーを実行します。フェイルオーバー中に、削除プロセスによって現在のサーバ上の該当する IP アドレスの設定が解除され、バックアップサーバ上でそのアドレスを設定できるようになります。この削除プロセスに失敗すると、システムは再起動します。

- IP 競合
- IP コリジョン
- DNS の名前解決の障害
- NIC やスイッチの障害

関連トピック

[切り替え可能な IP アドレスの作成](#)

[IP ローカルリカバリ](#)

リザベーションコンフリクト

- 保護されたデバイスへのリザベーションの喪失または盗難
- 保護されたリソースデバイスへのリザベーションまたは制御の回復が不可能 (手動のユーザ介入、HBA またはスイッチの障害が原因)

関連トピック

[SCSI リザベーション](#)

[リザベーションの無効化](#)

SCSI デバイス

- 保護された SCSI デバイスを開くことができない。デバイスに障害が発生しているか、システムからデバイスが削除されている可能性があります。

既知の問題と制限

下記に、LifeKeeper for Linux で明らかになっている制限または既知の問題を機能領域ごとに示します。

インストール

説明

リリース 7.4 以降では、SIOS 製品 RPM パッケージの再割り当てはサポートされません。

説明

Linux の依存関係

オプションの Recovery Kit を含めて SIOS Protection Suite for Linux のインストールを正常に完了するには、前提条件である多数のパッケージをインストールしておく必要があります。**注記:** 依存関係にあるこれらのパッケージのインストールが正常に完了していない場合、SIOS Protection Suite for Linux を起動する機能、および SIOS Protection Suite for Linux の GUI をロードする機能にも影響します。

スクリプトの失敗を防ぐには、これらのパッケージをインストールしてから、インストール設定スクリプトを実行してください。依存関係の詳細については、[Linux の依存関係](#)を参照してください。

新しいデバイスがスキャンされているときに nbd ドライバがロードされると、multipathd デモンはエラーログにエラーを記録します

解決方法: これらのエラーをログに記録しないようにするには、`/etc/multipath.conf` の blacklist に `devnode "^nbd"` を追加します。

NFS Setup Logging が不完全です

ISO イメージ (sps.img) からインストール設定スクリプトを実行する場合、NFS のスクリプトパッチプロセスの結果は、LifeKeeper インストールログ (`/var/log/LK_install.log`) でキャプチャされません。対応策はありません。

RHEL 5.9、CentOS 5.9、および Oracle Linux 5.9 では、Java 1.6.0_33 のインストール手順で軽微なエラーが発生

SIOS Protection Suite のインストール/設定フェーズで Java 1.6.0_33 コンポーネントがインストールされるときに、軽微なエラーが発生します。このエラーは、Java パッケージで発生するものです。致命的ではありません。Java package はインストールされ、フルに機能します。このエラーは RHEL 5.9 およびその派生カーネルに固有のものであり、[Red Hat の既知の問題](#)です。メッセージは、以下ようになります。

```
[<fffffffff8012ee59>] security_ops_task_setrlimit+0x87/0x96
[<fffffffff8009dbc9>] do_prlimit+0xd7/0x1d2
[<fffffffff8009ed12>] sys_setrlimit+0x36/0x43
[<fffffffff8005d29e>] tracesys+0xd5/0xdf
```

説明

mksh SIOS Protection Suite for Linux の必須設定との矛盾 ksh

mksh パッケージがインストール済みの場合、SIOS Protection Suite for Linux の設定は、パッケージの矛盾があることを示して、失敗します。SIOS Protection Suite for Linux には ksh パッケージが必須です。

対応策: RHEL、CentOS、または Oracle Linux では、mksh パッケージをアンインストールして ksh パッケージをインストールしてください。ksh パッケージをインストールした後、SIOS Protection Suite for Linux の設定を再実行してください。

例:

1. mksh パッケージをアンインストールしてください。

```
yum remove mksh
```

2. ksh パッケージをインストールしてください。

```
yum install ksh
```

3. 設定スクリプトを再実行してください。

説明

JRE 7 update 67 64-bit のインストール時にエラーが表示されます

LifeKeeper のインストーラーは JRE 7 update 67 64-bit をインストールします。既に JRE 7 update 67 32-bit がインストール済みの環境である場合に、下記のようなエラーが表示され、インストール済みの JRE 7 update 67 32-bit のパッケージに対して、JRE 7 update 67 64-bit のパッケージが上書きインストールされる場合があります。

```
Java is installed, but the package may not be compatible with SPS for Linux.
You should install the recommended version of java.
```

```
Do you wish to install the Java Runtime Environment v1.7.0_67 on your system
(y/n) [y] ?
```

```
The following packages will be installed:
```

```
jre-7u67-linux-x64.rpm
```

```
Preparing... #####
```

```
jre #####
```

```
Unpacking JAR files...
```

```
rt.jar...
```

```
Error: Could not open input file: /usr/java/jre1.7.0_67/lib/rt.pack
```

```
jsse.jar...
```

```
Error: Could not open input file: /usr/java/jre1.7.0_67/lib/jsse.pack
```

```
charsets.jar...
```

```
Error: Could not open input file: /usr/java/jre1.7.0_67/lib/charsets.pack
```

```
localedata.jar...
```

```
Error: Could not open input file: /usr/java/jre1.7.0_67/lib/ext/localedata.pack
```

```
jfxrt.jar...
```

```
Error: Could not open input file: /usr/java/jre1.7.0_67/lib/jfxrt.pack
```

```
Installation was successful.
```

```
Press ENTER to continue...
```

対応策:

LifeKeeper をインストールする前に JRE 7 update 67 32-bit をアンインストールしていただくか、JRE 7 update 67 32-bit のインストールパスを /usr/java/jre1.7.0_67 以外に変更してください。

LifeKeeper Core

説明

言語環境の影響

一部の LifeKeeper スクリプトは、Linux システムユーティリティの出力を解析し、一定のパターンに従って情報を抽出します。英語圏以外のロケールで一部のコマンドが実行されている場合、予測されたパターンは変更され、LifeKeeper スクリプトは必要な情報の取得に失敗します。このため、`/etc/default/LifeKeeper` では、言語環境変数 `LC_MESSAGES` が POSIX「C」のロケール (`LC_MESSAGES=C`) に設定されています。言語を英語にして Linux をインストールする必要はありません (インストールメディアで利用できる任意の言語を選択可能)。`/etc/default/LifeKeeper` の `LC_MESSAGES` の設定は LifeKeeper にのみ影響します。`/etc/default/LifeKeeper` の `LC_MESSAGES` の値を変更する場合、LifeKeeper の動作に悪影響を及ぼす可能性があることに注意してください。悪影響は、メッセージカタログがさまざまな言語とユーティリティに対応してインストールされているかどうか、および LifeKeeper が予期していないテキスト出力をそれらが生成するかどうかによって異なります。

ファイルシステムラベルは、大規模な設定で使用しないことを推奨します

ファイルシステムラベルを使用すると、大きなクラスタの場合、起動時にパフォーマンスが低下する可能性があります。ラベルを使用するには、システムに接続されるすべてのデバイスをスキャンする必要があり、通常はその結果として問題が生じます。SAN に接続されているシステム、特にデバイスへのアクセスがブロックされている LifeKeeper が導入されているシステムの場合、このスキャンは非常に遅くなる可能性があります。

Red Hat システムでこのパフォーマンスの問題を防ぐには、`/etc/fstab` を編集し、ラベルをパス名に置き換えます。

gen/app リソースで構文エラーが発生する可能性があります

コアのアップグレードをせずに、`steeleye-ikGUI` パッケージのみアップグレードした場合、`gen/app` リソースで構文エラーが発生します。`steeleye-ikGUI` パッケージには、同じバージョンまたはそれ以降のバージョンのコアを必要とする `gen/app` GUI コンポーネントへの更新が含まれています。

注記: LifeKeeper をアップグレードする際に、GUI とコアパッケージを最新バージョンにアップグレードする必要があります。GUI パッケージと一緒にコアをアップグレードした場合、エラーは発生しないはずですが。

lkscsid は、次のものを発行するとシステムを停止します sendevent

`lkscsid` がディスク障害を検出すると、デフォルトでは `sendevent` を LifeKeeper に発行し、障害から復旧しようとします。`sendevent` はまず、ローカルで障害から復旧しようとします。それに失敗すると、ディスクの階層を別のサーバに切り替えて障害から復旧しようとします。一部のバージョンの Linux (RHEL 5 および SLES11) では、`lkscsid` は `sendevent` を発行できないため、代わりにすぐにシステムを停止します。これは、`/dev/sda` などの SCSI デバイスノードを使用している階層にのみ影響します。

RHEL 6 64-bit ではセットアップが失敗します

Red Hat Enterprise Linux 6 64-bit には互換性の問題があります。

解決方法: [Linux の依存関係](#) のトピックに記載されているパッケージをインストールしてください。SIOS Protection Suite をインストールする前にこれらのパッケージがインストールされていない場合、インストール作業が正常に終了しません。

説明

DataKeeper の Create Resource が失敗します

特定の環境 (IDE ディスクエミュレーションを使用する仮想環境、HP CCISS ストレージを装備したサーバなど) で DataKeeper を使用している場合、ミラーの作成時にエラーが発生することがあります。

```
ERROR 104052: Cannot get the hardware ID of the device "dev/hda3"
```

この原因は、LifeKeeper が該当ディスクを認識せず、デバイスに関連付けられている一意の ID を取得できないからです。

対応策: ディスクのパターンを DEVNAME device_pattern ファイルに追加します。次に例を示します。

```
# cat /opt/LifeKeeper/subsys/scsi/resources/DEVNAME/device_pattern
/dev/hda*
```

API アクセスに対するホスト名の指定

LifeKeeper サーバ認証情報の格納に使用するキー名は他の LifeKeeper サーバのホスト名と完全に一致する必要があります (そのサーバに対する hostname コマンドで表示されます)。ホスト名が FQDN の場合、認証キーは FQDN である必要があります。ホスト名が短縮名の場合、キーも短縮名にする必要があります。

対応策: [credstore](#) によって格納されたホスト名がホスト名と完全に一致していることを確認します。

8.2.0 より前のバージョンの LifeKeeper の lkbakups を使用する場合は、それ以降のバージョンでリストアするときに /etc/default/LifeKeeper を手動で更新する必要があります。

LifeKeeper/SPS の現在のバージョンでは、ロギングなどの主要なコアコンポーネントに対して大幅な機能強化が加えられています。これらの機能強化は /etc/default/LifeKeeper ファイルの設定に影響します。旧バージョンの LifeKeeper/SPS で作成した lkbakup を新しいバージョンの LifeKeeper/SPS でリストアすると、これらの設定値が正しくなくなり、矛盾が発生します。

解決方法: lkbakup をリストアする前に、/etc/default/LifeKeeper を保存してください。lkbakup からリストアした後、以下のチューニングパラメータにマージします。

8.0.0 より前のバージョンから取得した lkbakup を使用して、8.0.0 以降でリストアする場合:

```
LKSYSLOGTAG=LifeKeeper
LKSYSLOGSELECTOR=local6
```

9.0より前のバージョンから取得した lkbakup を使用して、9.0以降でリストアする場合:

```
PATH=/opt/LifeKeeper/bin:/usr/java/jre1.8.0_51/bin:/usr/java/bin:/usr/java/jdk1.8.0_51/bin:/bin:/usr/bin:/usr/sbin:/sbin
```

詳細については、[syslog でのロギング](#)を参照してください。

説明

リソースの作成後に lkbakup をリストアすると、破損したイクイバレンシが残されず

作成したリソースの設定ファイルは lkbakup 中に保存されます。lkbakup でバックアップした後に初めて作成したリソースは、このバックアップからリストアする際に適切に把握されない可能性があります。

解決方法: 新しいリソースを初めて追加する前に lkbakup からリストアしてください。新しいリソースが lkbakup の後で追加された場合、リストアの前に削除するか、リソースの階層のインスタンスを削除し、リストアの後で階層を再拡張してください。**注記:** 特定のリソースを初めて作成する際に lkbakup を実行することを推奨します。

フェイルオーバー時に誤った順序でリソースが削除されます

階層が共通リソースインスタンスを別のルート階層と共有している場合、カスケードフェイルオーバーまたはリソースフェイルオーバーの間、リソースは誤った順序で削除されることがあります。

解決方法: 共通ルートを作成すると、階層のリソースの削除がトップダウンで実行されます。

1. restore と remove を常に進める gen/app を作成します。
2. 現在のルートをすべて、この新しい gen/app の子にします。

注記: restore および remove スクリプトに `/bin/true` を使用すると、これが可能になります。

AppArmor を有効にした SLES11 ホストのコンソールには、LifeKeeper syslog の EMERG 重大度メッセージは表示されません。

LifeKeeper は、SLES11 の AppArmor syslog-ng 設定によって不許可にされた `/var/run/utmp` にアクセスしています。

解決方法: AppArmor を有効にした SLES11 のコンソールに LifeKeeper syslog の EMERG 重大度メッセージを表示するには、`/etc/apparmor.d/sbin.syslog-ng` に以下のエントリを追加してください。

```
/var/run/utmp kr
```

`sbin.syslog-ng` に追加すると、(再起動することなく) 既存の AppArmor の定義を以下の定義で置き換えることができます。

```
apparmor_parser -r /etc/apparmor.d/sbin.syslog-ng
```

以下のコマンドで EMERG syslog エントリを送信し、AppArmor が更新されたことを確認してください。

```
logger -p local6.emerg "This is a syslog/lk/apparmor test."
```

説明

RHEL 6.0 は推奨されません

RHEL 6.0 の使用はできるだけ避けてください。RHEL 6.0 を使用する場合、以下のような問題 (これらに限定されません) を解決するために OS のアップデートが必要になることを理解してください。

- EMC CLARiiON において、DMMP はケーブルの抜線からの復旧に失敗します (RHEL 6.1 で修正済み)。
- md のリカバリプロセスがハングします (RHEL 6.1 のカーネルの最初のアップデートで修正済み)

注記: DataKeeper 設定では、オペレーティングシステムを更新すると、SIOS Protection Suite for Linux の再インストール / アップグレードが必要になります。

ネストされたファイルシステム階層を削除すると、メッセージ「Object does not exist」が生成されます

解決方法: これは問題ではないので、メッセージを無視してかまいません。

ネストされたマウントの作成時に filesyshier が正しくないタグを返します

ネストされたファイルシステムのリソースがデータベースに存在する場合、ファイルシステムキットは、親とネストされた子の両方にファイルシステムを作成します。ただし、filesyshier は子のタグのみを返します。これにより、アプリケーションは子の依存関係を作成しますが、親の依存関係は作成しません。

解決方法: 1 つのマウントポイント内で複数のファイルシステムがネストされている場合、dep_create または UI の [Create Dependency] を使用して、親アプリケーションタグの追加の依存関係を手動で作成しなければなりません。

DataKeeper: ネストされているファイルシステムの作成は DataKeeper で失敗します

既存のファイルシステムを複製するために DataKeeper ミラーを作成するときに、ファイルシステムがこの構造内でネストされている場合、ファイルシステムをアンマウントしてからファイルシステムリソースを作成する必要があります。

対応策: ネストされているファイルシステムを手動でアンマウントしてから、個々のネストされているマウントの再マウント / 作成を行ってください。

lkstart on SLES 11 SP2 generates inssserv message

SLES 11 SP2 で lkstart の実行中に、以下の inssserv メッセージが生成されます。

```
inssserv: Service syslog is missed in the runlevel 4 to use
service steeleye-runit
```

LifeKeeper と steeleye-runit のスクリプトは、デフォルトで実行レベル 4 で起動するように設定されていますが、依存する init スクリプトの syslog はそのように設定されていません。システムの実行レベルが 4 に変更された場合、syslog は終了し LifeKeeper はロギングができなくなります。

解決方法: システムの実行レベルを絶対に実行レベル 4 に変更しないでください。

説明

Amazon EC2 環境でのシャットダウンストラテジーの問題

シャットダウンストラテジーを”Do not Switchover Resources”と設定しても、アクティブノードをシャットダウン、または再起動した場合にスイッチオーバーが発生することがあります。これは、以下のシステム設定で発生する可能性があります。

条件

プラットフォーム: Amazon EC2

OS: RHEL7系のOS、Oracle Enterprise Linux、またはCentOS

サーバのシャットダウンストラテジーの設定

http://jpdocs.us.sios.com/Linux/9.0/LK4L/TechDoc/index.htm#configuration/optional_configuration/tasks/setting_server_shutdown_strategy.htm

スイッチオーバーが発生する上記の問題を回避させる場合、以下に掲載する手順でアクティブノードのシャットダウン、または再起動を行ってください。

1. アクティブノードをシャットダウン、または再起動する前に、すべてのスタンバイノードで以下のコマンドを実行してください。

```
# /opt/LifeKeeper/bin/flg_create -f !nofailover\![アクティブノードのホスト名]
```

注記: [アクティブノードのホスト名]は、”/opt/LifeKeeper/bin/lcdunname”で出力するアクティブノードのホスト名に置き換えてください。

2. アクティブノードをシャットダウン、または再起動してください。

Amazon EC2 環境におけるSplit Brainの発生

ノードをシャットダウン、または再起動した場合にSplit Brainが発生することがあります。これは、以下のシステム設定で発生する可能性があります。

条件

プラットフォーム: Amazon EC2

OS: RHEL7系のOS、Oracle Enterprise Linux、またはCentOS

この事象は、LifeKeeperが起動した時点で、ネットワークが有効ではない場合に発生します。ネットワークが有効になるまでLifeKeeperの起動を遅延 (sleep) させることが対策となります。

ファイルシステムリソースに保護されたデバイスのマウントポイントの変更は、データ破損を引き起こす可能性があります

ファイルシステムリソースに保護されたデバイスのマウントポイントは変更しないでください。変更した場合は、ファイルシステムリソースが仕様通りにデバイスを処理することができず、デバイスが2つのノードにマウントされて、データ破損を引き起こす可能性があります。

インターネット /IP ライセンス

説明

/etc/hosts の設定の依存関係**/etc/hosts 設定:**

インターネットベースのライセンス (IPv4 アドレス) を使用している場合、/etc/hosts の設定はライセンスの検証に悪影響を与える可能性があります。LifeKeeper の起動に失敗した場合は、以下のようなメッセージが出力されます。

```
Error in obtaining LifeKeeper license key:
  Invalid host.
  The hostid of this system does not match the hostid specified in the
  license file.
```

リストされているインターネットホスト ID が正しい場合、/etc/hosts の設定が原因の可能性もあります。/etc/hosts エントリを正しく一致させるには、IPv6 エントリの前に IPv4 エントリを記載する必要があります。/etc/hosts 設定が原因かどうかを確認するには、次のコマンドを実行します。

```
/opt/LifeKeeper/bin/lmutil lmhostid -internet -n
```

記載された IPv4 アドレスが、インストールされたライセンスファイルの IPv4 アドレスと一致しない場合、正しいアドレスを返すために、/etc/hosts を変更し、IPv4 エントリを IPv6 エントリの前に配置する必要があります。

GUI

説明

GUI を終了した後で Web ブラウザを介して再接続すると、GUI ログインプロンプトが再表示されない場合があります

GUI アプレットを終了するか切断してから同じ Web ブラウザセッションから再接続しようとする、ログインプロンプトが表示されない場合があります。

対応策: Web ブラウザを閉じ、開き直してからサーバに接続してください。Firefox を使用している際は、すべての Firefox を閉じ、再び開きます。

RHEL 5 の kGUIapp が、対応していないテーマに関するエラーを報告します

GUI アプリケーションクライアントを起動すると、以下のコンソールメッセージが表示される場合があります。このメッセージは RHEL 5 および FC6 Java プラットフォームのルックアンドフィールに由来するもので、GUI クライアントの動作に悪影響を及ぼすことはありません。

```
/usr/share/themes/Clearlooks/gtk-2.0/gtkrc:60:Engine "clearlooks"
is unsupported, ignoring
```

説明

ネットワークが切断され、再接続された後で、GUI は IP リソースの状態をすぐに更新しません

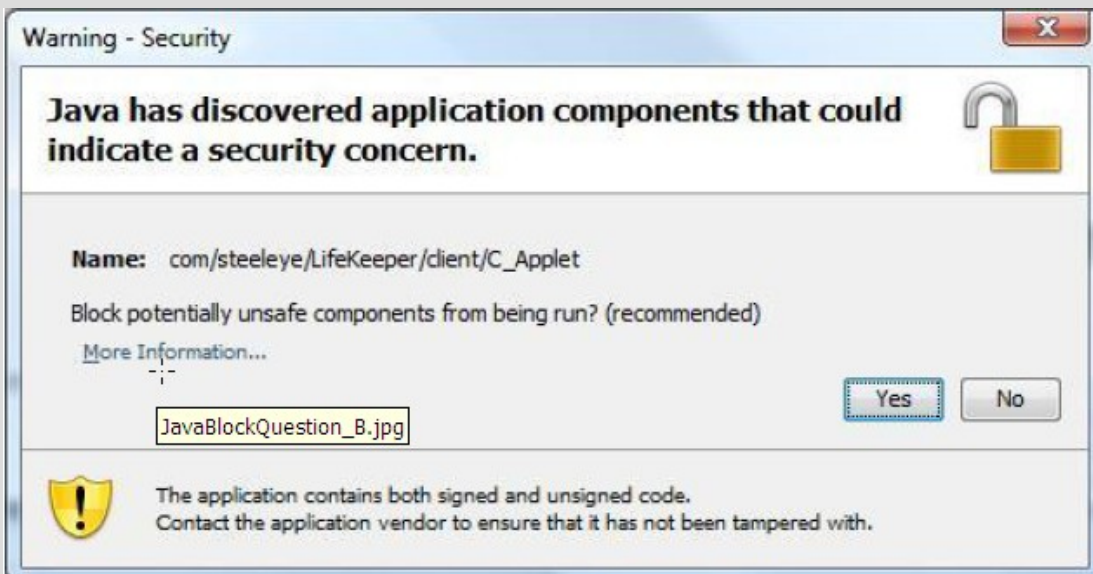
クラスタ内のサーバ間のプライマリネットワークが切断され、再接続されると、RMI/TCP レイヤーの問題のため、リモート GUI クライアントの IP リソースの状態が更新されるまで 1 分 25 秒かかる場合があります。

説明

Java 署名/未署名混合コードの警告 - LifeKeeper Java GUI クライアントアプレットをリモートシステムからロードすると、以下のセキュリティ警告が表示されることがあります。



[Run] をクリックすると、以下のダイアログが表示されます。



ブロックするかどうかを確認するメッセージが表示されます。[No] をクリックすると、LifeKeeper GUI の動作が可能になります。

推奨される処置: セキュリティ警告の数を減らすには、2つのオプションがあります。
SIOS Protection Suite for Linux テクニカルドキュメンテーション

1. **[Always trust content from this publisher]** ボックスをチェックし、**[Run]** をクリックします。次に LifeKeeper GUI Java クライアントを読み込むときには、警告メッセージが表示されません。

説明

ポート 778 が使用中の場合、steeleye-lighttpd プロセスの開始に失敗します

steeleye-lighttpd の起動時にプロセスがポート 778 を使用している場合、steeleye-lighttpd の起動に失敗し、GUIへの接続障害が発生する。

解決方法: クラスタ内のすべてのノードで以下の設定を行い、LifeKeeper をすべてのノードで再起動します。

以下の行を `/etc/default/LifeKeeper` に追加します。

```
API_SSL_PORT=port_number
```

port_number は使用する新しいポートです。

データレプリケーション

説明

LVM 構成における DataKeeper for Linux 非同期モードに関する注意事項

複数の非同期ミラー上に LVM を構成すると、データの整合性が保てず、さらには kernel panic が発生する場合があります。

DataKeeper 上に LVM を構成する場合には、DataKeeper ミラー 1 つだけ、または複数の同期ミラー、いずれかの構成でなければなりません。

両方のサーバに重要な I/O トラフィックを持つシンメトリックなアクティブ SDR 設定で、netraid デバイス (ミラー) にマウントされたファイルシステムが応答を停止し、結果的にシステム全体がハングします

Linux バッファキャッシュの単一スレッドの特性により、バッファキャッシュフラッシングデーモンは、リモートでコミットする必要があるバッファをフラッシュアウトしようとしてハングする可能性があります。フラッシングデーモンがハングすると、クリアされていないバッファの数がシステムで許容されている上限 (`/proc/sys/kernel/vm/bdflush` で設定) を超えると、クリアされていないバッファを持つ Linux システムのすべてのアクティビティが停止します。

これは、リモートシステムがリモートバッファを消去できなくなるような事態でないかぎり、通常は深刻な問題ではありません。LifeKeeper はネットワークの障害を検出し、そのときにレプリケーションを停止するので、ハング状態は消去されます。ただし、リモートシステムがローカルシステムにもレプリケートされた場合 (つまり、相互がシンメトリカルにレプリケートされた場合)、両方のシステムでこのフラッシングデーモンのハング状態が発生すると、永久にデッドロックする可能性があります。

デッドロックを解除するには、両方のシステムの nbd-client デーモンを手動で停止します (これにより、ミラーが切断されます)。ただし、このデッドロックを完全に防止する場合は、シンメトリックアクティブレプリケーションを推奨しません。

説明

ミラーが切断され、/var/log/messages にたくさんのエラーが記述されます

この問題は、(Red Hat EL 6.x や CentOS 6 で) 意図的に障害を発生させるストレステストを実行しているとき (特に、ミラーターゲットシステムで実行されている `nbds_server` プロセスを停止しているときに) ときおり見られます。ディストリビューションの最新カーネル (Red Hat EL 6.0 または 6.1 の場合は `kernel-2.6.32-131.17.1` など) にアップグレードすると、この問題が発生するリスクの軽減に役立つ場合があります。ソースシステムを再起動すると、この問題はなくなります。

CentOS 6 に付属のデフォルトカーネル (2.6.32-71) では、(ミラーの過負荷に過ぎない場合でも) この問題はさらに頻繁に発生する可能性があります。残念なことに、CentOS はこの状態を改善するカーネル (2.6.32-131.17.1) をまだリリースしていません。SIOS は、CentOS 6 で入手可能になり次第、2.6.32-131.17.1 カーネルへのアップグレードを推奨しています。

注記: SPS 8.1 以降、Red Hat Enterprise Linux システムでカーネルのアップグレードを実行する際、インストールイメージから `setup` スクリプト (`./setup`) を再実行する必要はなくなりました。カーネルが適切な Red Hat package (rpm ファイル) からインストールされている限り、モジュールはアップグレードしたカーネルで特別な操作を必要とせずに、自動的に使用可能になります。

大きなミラーサイズを備えた md_raid1 プロセスではトップで高い CPU 使用率がレポートされます

`mdX_raid1` プロセス (X はミラー番号) では、非常に大きなミラー (500GB 以上) を操作している際に、一部の OS ディストリビューションで高い CPU 使用率がトップで報告されることがあります。

解決方法: CPU の使用率を減らすには、LifeKeeper 設定 `LKDR_CHUNK_SIZE` でチャンクサイズを 1024 に変更し、この新しい設定を使用するためにミラーを削除して再作成します。

DataKeeper リソースで lkbbackup を使用する場合は、全同期が必要です

`lkbbackup` では `instance` と `mirror_info` ファイルを保存しますが、リソースが存在しないときにソースおよびターゲットの状態は保証されないため、`lkbbackup` から restore した後で、DataKeeper ミラーの全同期を実行することがベストプラクティスです。

初期の Red Hat/CentOS 6.x のカーネルバージョンでは、LifeKeeper のログに「Failed to remove device」というメッセージを出してミラーの再同期がハングすることがあります

バージョン 2.6.32-131.17.1 より前のカーネルバージョン (更新前の RHEL 6.1 カーネルバージョン 2.6.32-131.0.15 など) では、レプリケーションに使用する md ドライブに問題があります。この問題によって、`nbds` デバイスがミラーから解放されなくなり、「Failed to remove device」メッセージが複数記録され、ミラーの再同期が中止されます。この状態を解消するには、システムの再起動が必要です。この問題は、ミラー作成後の最初の再同期中、およびミラーが高負荷のときに発生したことがあります。

解決方法: カーネル 2.6.32-131.17.1 にはこの問題の修正が含まれていることが確認されています。2.6.32-131.17.1 より前のカーネルバージョンの Red Hat または CentOS 6 で DataKeeper をお使いの場合は、このバージョンまたは最新のバージョンにアップデートすることを推奨します。

IPv6

説明

SIOS は、`ifconfig` コマンドから `ip` コマンドの使用に移行しました。この変更のため、外部スクリプトを使用するお客様も、同様の変更を行うことを推奨します。`ifconfig` コマンドを発行し、結果を解析して特定のインターフェースを探す代わりに、スクリプトは「`ip -o addr show`」を使用し、結果を解析して、「`inet`」および「`secondary`」という語を含む行を検索します。

```
# ip -o addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
  \   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
1: lo    inet 127.0.0.1/8 scope host lo
1: lo    inet6 ::1/128 scope host
  \     valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
  \   link/ether d2:05:de:4f:a2:e6 brd ff:ff:ff:ff:ff:ff
2: eth0    inet 172.17.100.77/22 brd 172.17.103.255 scope global eth0
2: eth0    inet 172.17.100.79/22 scope global secondary eth0
2: eth0    inet 172.17.100.80/22 scope global secondary eth0
2: eth0    inet6 2001:5c0:110e:3364::1:2/64 scope global
  \     valid_lft forever preferred_lft forever
2: eth0    inet6 2001:5c0:110e:3300:d005:deff:fe4f:a2e6/64 scope
global dynamic
  \     valid_lft 86393sec preferred_lft 14393sec
2: eth0    inet6 fe80::d005:deff:fe4f:a2e6/64 scope link
  \     valid_lft forever preferred_lft forever
```

`ip` コマンドの上記の結果では、以下の行に `eth0` インターフェースの仮想 IP アドレスが含まれます。

```
2: eth0    inet 172.17.100.79/22 scope global secondary eth0
2: eth0    inet 172.17.100.80/22 scope global secondary eth0
```

説明

/etc/sysconfig/network-scripts/ifcfg-<nicName> の「IPV6_AUTOCONF = No」が再起動または起動の際に考慮されません

起動時に、自動設定されたステートレスな IPv6 アドレスがネットワークインターフェースに割り当てられます。IPV6_AUTOCONF=No が設定されているインターフェースのステートレス IPv6 アドレスでコミュニケーションパスが作成された場合、任意のシステムリソースが ifdown <nicName>;ifup <nicName> などのインターフェースを管理する際にアドレスが削除されます。

IPV6_AUTOCONF が No に設定されていたため、プライマリサーバを再起動した後で、自動設定された IPv6 アドレスを使用しているコミュニケーションパスは復旧せず、無効のままでした。

解決方法: スタティックな IPv6 アドレスのみを使用してください。自動設定された IPv6 アドレスを使用すると、再起動後に通信が失われたり、NIC が変更されたりする可能性があります。

自動設定された IPv6 アドレスをコミュニケーションパスの作成に使用できますが、システム管理者は以下の条件を認識する責任があります。

- 自動設定されたステートレス IPv6 アドレスがネットワークインターフェース (NIC) の MAC アドレスに準拠していること。コミュニケーションパスが作成され、関連 NIC が後で置き換えられた場合、自動設定された IPv6 アドレスは異なるものになり、LifeKeeper はコミュニケーションパスが無効になっていることを正しく表示します。コミュニケーションパスを再作成する必要があります。
- RHEL 5.6 では、ホスト操作のあらゆる側面で一貫した IPv6 自動設定を確保するための動作を実行するには、sysctl.conf、net.ipv6.* 命令 ('if/ip' ユーティリティで参照される ifcfg-<nic> の明示的な IPV6_AUTOCONF 設定、およびシステムが起動して init レベルを切り替える際に NIC 制御に影響する /etc/sysctl.conf) に加え、個々のインターフェース設定ファイルを正確に設定するために、詳細かつ具体的なドメインの知識が必要になります。

IP:IPv6 のソースアドレス変更設定ではソースアドレスが設定されません

IPv6 IP リソースのソースアドレスを設定しようとすると、何も変更されていない場合でも成功となります。

対応策: 現在のところ、対応策はありません。今後のリリースで対応する予定です。

IP:無効な IPv6 アドレス設定が IP リソース作成で許可されます

オクテットに 4 文字を超える文字が含まれている場合、2001:5c0:110e:3368:000000:000000001:61:14 という形式の IPv6 アドレスが許容されます。

対応策: 正しい形式の IPv6 アドレスを入力してください。

IPv6 アドレス設定からホストに接続できません

lkGUIapp は、解決可能なホスト名または IP アドレスの場合でも、IPv6 の 16 進数アドレス設定からホストに接続できません。lkGUIapp が接続するには、IPv4 設定のノードが必要です。IPv6 コミュニケーションパスは完全にサポートされています。

説明

bonding NIC に割り当てられているものの、「暫定的な」状態のアドレスでは、IPv6 リソースが ISP としてレポートされます

LifeKeeper の IPv6 保護リソースは、IPv6 リソースが bonding インターフェース上にある SLES システムでは「In Service Protected」(ISP: in service の保護)と不正に識別されます。これは、'active-backup' (1) および 2.6.21 以前の Linux カーネルとは別のモードです。IPv6 の bonding リンクは、解決できないアドレスを持つ「暫定的な」状態のままになります。

対応策: bonding インターフェースモードを 'active-backup' (1) に設定します。または、'active-backup' (1) 以外のモードの場合、リンク状態を「tentative (暫定的)」から「valid (有効)」に設定する更新したカーネルで操作します。

注記: SPS 8.1 以降、Red Hat Enterprise Linux システムでカーネルのアップグレードを実行する際、インストールイメージから setup スクリプト(./setup)を再実行する必要はなくなりました。カーネルが適切な Red Hat package (rpm ファイル) からインストールされている限り、モジュールはアップグレードしたカーネルで特別な操作を必要とせずに、自動的に使用可能になります。

Apache

説明

Apache キットは IPv6 をサポートしていません。httpd.conf で IPv6 を識別しません

httpd.conf ファイルの「Listen」命令エントリに IPv6 アドレスを割り当てると、問題が発生します。

解決方法: Apache Recovery Kit で IPv6 がサポートされるまで、リソース作成後に IPv6 アドレスを httpd.conf ファイルに指定できません。

Oracle Recovery Kit

説明

Oracle Recovery Kit には Connection Manager および Oracle Names 機能のサポートが含まれていません

LifeKeeper Oracle Recovery Kit には、Oracle Connection Manager と Oracle Names という Oracle Net 機能のサポートが含まれていません。Oracle Connection Manager は、同じサービスにアクセスする必要がある多数の接続を管理するルーティングプロセスです。Oracle Names はサービスアドレスの一括格納を管理する Oracle 固有の命名サービスです。

LifeKeeper Oracle Recovery Kit は、送信されてきたクライアント接続要求をリスニングし、サーバへのトラフィックを管理する Oracle Net Listener プロセスを保護します。Oracle Listener に固有の LifeKeeper 設定情報については、『LifeKeeper for Linux Oracle Recovery Kit 管理ガイド』を参照してください。

説明

Oracle Recovery Kit は Oracle 10g の ASM またはグリッドコンポーネント機能をサポートしていません

以下の情報は Oracle 10g データベースインスタンスのみを対象とします。Oracle 10g で提供されている Oracle Automatic Storage Manager (ASM) 機能は現在、LifeKeeper ではサポートされていません。また、10g のグリッドコンポーネントは LifeKeeper Oracle Recovery Kit によって保護されません。Raw デバイス、ファイルシステム、論理ボリュームに対するサポートは、現在の LifeKeeper for Linux Oracle Recovery Kit に含まれています。グリッドコンポーネントに対するサポートは、gen/app リカバリキットを使用して LifeKeeper の保護機能に追加できます。

Oracle Recovery Kit は NFS バージョン 4 をサポートしていません

Oracle Recovery Kit は共有データベースストレージ用に NFS バージョン 3 をサポートしています。NFSv4 ファイルロックメカニズムのため、NFS バージョン 4 は、現時点ではサポートされていません。

フェイルオーバー後にプライマリサーバで Oracle Listener が in service のままになります

ネットワーク障害により、アプリケーションがバックアップサーバにフェイルオーバーした後、プライマリサーバで Listener プロセスがアクティブのままになります。正しいデータベースへの接続は影響を受けませんが、その Listener プロセスを終了したいと考えるかもしれません。

NFS Server Recovery Kit

説明

最上位の NFS リソース階層は hanfs リソースのスイッチバックタイプを使用します

障害からサービス状態に復旧する際に NFS リソース階層がプライマリサーバにスイッチバックするかどうかを制御するスイッチバックタイプは、hanfs リソースで定義されます。

一部のクライアントが nfs ファイルロックを再取得できません

NFS クライアントとして動作しているとき、一部の Linux カーネルは、「NFS ロックが解除されているので、再取得する必要がある」という、NFS サーバからの通知に正常に応答しません。そのため、これらのシステムが、LifeKeeper に保護されている NFS ファイル共有のクライアントである場合、これらのクライアントで保持されている NFS ロックは、フェイルオーバーまたはスイッチオーバーの際に失われます。

NFS マウントオプションの推奨値で NFS ロックを使用する場合、以下のオプションを設定することを推奨します。

```
rw, nolock, bg, hard, nointr, tcp, nfsvers=3, timeo=600, rsize=32768, wsize=32768, actimeo=0
```

NFS v4 の変更は SLES 11 nfs サブシステムの操作と互換性はありません

SLES 11 の非 NFS v4 リモートエクスポートのマウンティングによって、rpc.statd が開始されます。NFS v4 ルートエクスポートを保護するクラスタ内の out of service ノードでは、rpc.statd の開始に失敗します。

解決方法: NFS v4 ルートエクスポートを保護しているクラスタで NFS v2/v3 と混在させないでください。

説明

IPv6 では NFS v4 を設定できません

IPv6 仮想 IP は NFSv4 階層にまとめられます。

解決方法: NFSv4 リソースの作成時に IPv6 仮想 IP リソースを使用しないでください。

NFS v4: 拡張解除後に階層を再拡張できません

エクスポートポイントがすでにターゲットサーバでエクスポート済みなので、拡張に失敗します。NFS v4 階層がサーバ A で作成され、サーバ B に拡張され、サーバ B で in service になった後にサーバ A から拡張解除された場合、NFS v4 階層のサーバ A への再拡張は失敗します。

解決方法: サーバ A で「`exportfs -ra`」というコマンドを実行し、残された追加エクスポート情報をクリーンアップします。

NFSv3: Red Hat 6.x および CentOS 6.x ではファイルロックのスイッチオーバーに失敗します

サーバのフェイルオーバーまたはスイッチオーバーの際に、ファイルロックのフェイルオーバーは、Red Hat 6.x または CentOS 6.x のシステムでは機能しません。NFSv3 のロックフェイルオーバーは現在、これらの OS バージョンではサポートされていません。

解決方法: NFSv4 で有効なロックフェイルオーバー機能を使用します。

Oracle Recovery Kit は NFSv4 をサポートしていません

Oracle Recovery Kit は共有データベースストレージ用に NFSv3 をサポートしています。NFSv4 ファイルロックメカニズムのため、NFSv4 は、現時点ではサポートされていません。

NFS サブシステムの停止と起動が、SIOS Protection Suite で保護される NFS エクスポートに悪影響を及ぼします

SIOS Protection Suite for Linux が NFS エクスポートを保護しているときに、NFS サブシステム (Red Hat の `/etc/init.d/nfs` または SuSE の `/etc/init.d/nfsserver`) が停止された場合、NFS の停止動作ですべてのディレクトリのエクスポート解除が実行されるため、SIOS Protection Suite for Linux が保護しているすべてのエクスポート済みディレクトリが影響を受けます。SIOS Protection Suite for Linux の NFS quickCheck スクリプトが停止した NFS のプロセスを検出し、ローカルリカバリを実行してプロセスを再起動してから、ディレクトリを再エクスポートします。ただし、SIOS Protection Suite の NFS ARK がすべてを復旧するために、保護されたエクスポートのすべてについて quickCheck が実行されます。例えば、SIOS Protection Suite for Linux の NFS ARK により 5 個のエクスポートが保護されている場合、そのキットが保護するすべてのエクスポート済みディレクトリを復旧するために、quickCheck が 5 回実行されます。デフォルトの quickCheck 時間は 2 分なので、すべてのエクスポート済みディレクトリを復旧するには、10 分かかります。

対応策: SIOS Protection Suite の NFS ARK がシステム上のエクスポート済みディレクトリをアクティブに保護しているときには、NFS サブシステムを停止しないでください。NFS サブシステムを停止する必要がある場合は、すべての NFS リソースをスタンバイノードに切り替えてから NFS サブシステムを停止してください。`exportfs` コマンドの使用も検討する必要があります。このコマンドラインユーティリティは 1 つのディレクトリのエクスポート / エクスポート解除の機能を装備しているので、NFS サブシステム全体を停止する必要がなくなります。

SAP Recovery Kit

説明

SAP 階層の削除または拡張解除に失敗します

同じ IP リソースを階層内の複数の場所に格納している SAP 階層を削除または拡張解除すると、リソースが削除されず、コアダンプが発生することもあります。

この問題を修正するには、拡張解除または削除操作に失敗した後で、残ったリソースを LifeKeeper GUI から手動で削除します。サーバからコアファイルを削除するという方法もあります。

[Handle Warnings] により -e 行 1 で構文エラーが発生します

[Handle Warnings] のデフォルトの動作 [No] を [Yes] に変更すると、エラーを受信します。

解決方法: このオプションをデフォルト設定の [No] のままにしてください。**注記:** 黄は過渡的な状態であり、ほとんどの場合は障害を表すものではないため、この設定はデフォルト選択の [No] のままにすることを強くお勧めします。

同じ設定を選択すると、Update Wizard のボタンが非表示になります

現在の設定を変更せずに [Handle Warnings] を更新しようとする、戻る必要があることを示す次の画面で [Done] ボタンが表示されません。

res_state に変更を加えると、監視が無効になります

[Protection Level] を [BASIC] に設定し、SAP を手動でダウンさせた場合 (メンテナンスなどの目的で)、「FAILED」とマークされ、監視が停止します。

解決方法: 監視を再開する場合、LifeKeeper はリソースを手動ではなく開始する必要があります。

ERS が Core/CI の親ではない場合、サービス中の ERS がリモートホストで失敗します

追加 SAP リソースの依存関係なしで ERS リソースを作成すると、スイッチオーバー時に初期のサービス中状態が失敗します。

解決方法: CI/Core インスタンス (SCS または ASCS) の親として ERS を作成してから、in-service を再試行します。

LVM Recovery Kit

説明

LVM 構成における DataKeeper for Linux 非同期モードに関する注意事項

複数の非同期ミラー上に LVM を構成すると、データの整合性が保てず、さらには kernel panic が発生する場合があります。

DataKeeper 上に LVM を構成する場合には、DataKeeper ミラー 1 つだけ、または複数の同期ミラー、いずれかの構成でなければなりません。

説明

IKID の使用はディスク全体で LVM を上書きした状態と互換性がありません

IKID を使用して、LVM 物理ボリュームとして設定されているディスクで一意的なディスク ID を生成すると、IKID および LVM 情報が格納されているディスク上の場所で競合が発生します。これにより、IKID および pvcreate が使用された順番に従って、IKID または LVM 情報のどちらかが上書きされます。

対応策: IKID を LVM と組み合わせて使用する必要がある場合は、ディスクをパーティション化し、ディスク全体ではなくディスクパーティションを LVM 物理ボリュームとして使用します。

LVM のアクションが RHEL 6 では遅くなります

RHEL 6 で一部の LVM コマンドを実行している際、前のリリースよりもパフォーマンスが遅くなる場合があります。LVM リソースを含む階層の restore および remove に要する時間がやや長くなることとして現れることがあります。

Raw および LVM の Recovery Kit を組み合わせた設定は、RHEL 6 環境ではサポートされません

Raw リソースを作成するとき、Raw Recovery Kit は Raw デバイスの major 番号および minor 番号に基づいてデバイスファイルを探します。その結果、`/dev/dm-*` がそのデバイスになります。ただし、LVM Recovery Kit はこのタイプの `/dev/dm-*` を処理できないため、GUI で「raw device not found」エラーが発生します。

DMMP Recovery Kit

説明

DMMP:スタンバイサーバで発行された write がハングすることがあります

別のサーバでリザーブされている DMMP デバイスに write が発行されると、IO が永久に（またはデバイスがもう一方のサーバでリザーブされなくなるまで）ハングすることがあります。もう一方のサーバでデバイスが解放され、write が発行されると、データが破損することがあります。

この問題の原因は、DMMP での IO 再試行に従ってパス確認が実行される方法にあります。「no_path_retry」が 0 (失敗) に設定されている場合、このハングは発生しません。別のサーバでパスがリザーブされているときにデバイスの path_checker が失敗しても (MSA1000)、この問題は発生しません。

対応策: 「no_path_retry」を 0 (失敗) に設定します。しかしこれにより、一時的なパスの障害が原因で、IO の障害が発生する可能性もあります。

説明

DMMP:複数のイニシエータが ATP_C をサポートする SAS アレイで正しく登録されていません

LifeKeeper は、複数の SAS イニシエータが SAS アレイに接続される設定をネイティブにサポートしていません。こうした設定では、各イニシエータが正常に登録されないため、1つのイニシエータのみが IO を発行できるようになります。マルチパスドライバ (DMMP など) が未登録のイニシエータに IO を発行すると、エラーが発生します。

解決方法: SAS ストレージ情報に基づいてパス ID が設定されるように、`/etc/default/LifeKeeper` の設定値を以下のように設定してください。

```
MULTIPATH_SAS=TRUE
```

RHEL 6 の場合、LifeKeeper は EMC Clariion に接続されているリザベーションをサポートできません

DMMPリカバリーキットのパラメータ設定が必要な場合、複数の異なるストレージを同時に使用することはできません。

DB2 Recovery Kit

説明

DB2 Recovery Kit が不要なエラーをレポートします

DB2 が共有ディスクにインストールされている場合、DB2 リソースを拡張する際に以下のメッセージが表示されることがあります。

```
LifeKeeper was unable to add instance "%s" and/or its variables to the DB2 registry.
```

このメッセージは、DB2 リソースの拡張動作には悪影響を及ぼしません。

MD Recovery Kit

説明

MD Recovery Kit は、「homehost」で作成されたミラーをサポートしません

LifeKeeper MD Recovery Kit は、「homehost」機能で作成されたミラーでは正常に機能しません。「homehost」が設定された場合、LifeKeeper は、不正なフォーマットの一意の ID を使用するので、in-service 操作が失敗します。SLES 11 システムでは、「homehost」はミラーの作成時にデフォルトで設定されます。「homehost」に対応している mdadm のバージョンは、別のディストリビューションやバージョンでも使用可能と思われれます。この機能を無効にするには、ミラー作成時にコマンドラインで `--homehost=""` を指定します。「homehost」設定を使用して作成されたミラーがすでに存在している場合は、設定を無効にしてミラーを再作成する必要があります。「homehost」設定を使用して作成されたミラーで LifeKeeper 階層がすでに構築されている場合、階層を削除し、「homehost」を無効にしてミラーを構築した後で再作成する必要があります。

MD Recovery Kit は LVM デバイスで作成された MD デバイスをサポートしていません

LifeKeeper MD Recovery Kit は、LVM デバイスで作成された MD デバイスを正常に処理しません。MD デバイスの作成時に LifeKeeper が認識できない名前が付けられます。

/etc/mdadm.conf の MD Recovery Kit 設定ファイルエントリがコメントアウトされていません

/etc/mdadm.conf の LifeKeeper 設定ファイルエントリ破砕起動後にコメントアウトする必要があります。これらのファイルエントリはコメントアウトされていません。

大規模な設定ではローカルリカバリが実行されません

大規模な設定 (6 以上の階層) では、ローカルリカバリがトリガされた場合 (`sendevent`)、すべての階層がチェックされず、ローカルリカバリが失敗することがあります。

起動時にミラーが自動的に開始されます

一部のシステム (RHEL 6 を実行しているシステムなど) では、起動時に自動的にミラーを開始する設定ファイル (/etc/mdadm.conf) に AUTO エントリがあります (例: `AUTO +imsm +1.x -all`)。

解決方法: LifeKeeper では、ミラーを自動的に開始しないようにする必要がありますので、このエントリを編集し、起動時に自動的に開始しないように指定する必要があります。前の例 (`AUTO +imsm +1.x -all`) は、`imsm` メタデータおよび `1.x` メタデータから他のすべてを除いたものを使用して作成したミラーを自動的に開始するようにシステムに指示しています。このエントリを「`AUTO -all`」に変更し、あらゆるものからすべてを除いて自動的に開始するように (つまり、何も自動的に開始されないように) システムに指示する必要があります。

重要: クリティカルなシステムリソース (root など) が MD を使用している場合、それらのミラーが他の方法で開始され、LifeKeeper で保護されているミラーは開始されないようにしてください。

SAP DB/MaxDB Recovery Kit

説明

MaxDB を共有ストレージ上でインストールおよび設定すると、拡張に失敗します

対応策: MaxDB のローカルコピーをバックアップノード上でプライマリシステムと同じディレクトリにインストールしてください。

Sybase ASE Recovery Kit

説明

ユーザ名 / パスワードの問題:

- デフォルトのユーザ名がパスワードで保護されている場合、すべての検証が完了するまで create UI はこれを検出しません

Sybase リソースを作成する際に、ユーザ名の入力が必要されます。フロントのヘルプには、ユーザが指定されない場合はデフォルトの「sa」が使用されるというメッセージが表示されます。ただし、この時点ではデフォルトのアカウントに対するパスワード検証は実行されません。SIOS Protection Suite が Sybase リソースを作成しようとする際に、パスワードの検証 / 入力が行われていないためにリソース作成が失敗します。パスワード検証は [user/password] ダイアログで発生しますが、ユーザプロンプトに有効なユーザが実際に入力された場合のみです。デフォルトのユーザ名を使用する場合でも、create 操作時にそのユーザ名を指定する必要があります。

- ユーザ名が指定されない場合、パスワードの入力要求が省略されます

ユーザ名を入力しない場合、[user/password] ダイアログはパスワードの入力要求を省略します。UI オプションを使用してユーザ / パスワードを更新する際に Sybase のユーザ名を入力しない場合、デフォルトの「sa」が使用され、そのアカウントのパスワード検証は実行されません。これにより、無効な認証情報のエラーが発生して、データベースの監視が失敗します。デフォルトのユーザ名を使用する場合でも、update 操作時にそのユーザ名を指定する必要があります。この問題を解決するには、以下の手順を実行してください。

1. 必要な Sybase データファイルが、目的のサーバから現在アクセス可能であることを確認します。多くの場合、プライマリ上のローカルリカバリの障害を監視するため、これはバックアップサーバです。
2. このサーバのコマンドラインから Sybase データベースインスタンスを開始してください(データベースを手動で開始する方法の詳細は、Sybase 製品のドキュメンテーションを参照)。
3. コマンドラインを使用して、LKROOT/bin ディレクトリ(ほとんどのインストールでは /opt/LifeKeeper/bin)に移動 (cd) してください。
4. bin ディレクトリで次のコマンドを実行してください。

```
./ins_setstate -t <SYBASE_TAG> -S ISP
```

<SYBASE_TAG> は Sybase リソースのタグ名です。

5. コマンドの実行が完了した後、ただちに UI から **Update User/Password Wizard** を実行し、有効なユーザ名を入力してください (Sybase のデフォルトの「sa」を使用する場合も同様)。注記: **Update User/Password Wizard** にアクセスするには、Sybase リソースインスタンスを右クリックし、[Change Username/Password] を選択してください。
6. ローカルサーバ上で階層が更新された後、リソースがすべてのノードで in service にすることができることを確認してください。

- Sybase のローカルユーザ名が 8 文字以上の場合、バックアップサーバの保護が失敗します

Sybase のユーザ名は 8 文字未満にする必要があります。Sybase のローカルユーザ名が 8 文字以上の場合、リソース作成に使用されるプロセスとユーザ ID のチェック、および監視が失敗します。また、これにより、有効な Sybase Backup Server のインスタンスを保護の対象として選択できなくなります。この問題は、オペレーティングシステムが 8 文字以上のユーザ名を各種コマンド (ps など) の UID に変換することに起因します。7 文字以下のユーザ名を使用する必要があります。

説明

リソース作成問題:

- Sybase のデフォルトのインストールプロンプトは、12.5 に基づきます。SIOS Protection Suite でリソースを作成する際に、Sybase のインストール場所のデフォルトプロンプトが、Sybase バージョン 12.5 からの相対パスとして表示されます。リソース作成のプロンプトには、Sybase の正しいインストール場所を手動で入力するか、参照する必要があります。

拡張の問題:

- 拡張時に Sybase のタグプロンプトが編集可能ですが変更しないでください拡張時に Sybase のタグプロンプトが編集可能ですが、これは推奨されません。サーバごとに異なるタグを使用すると、コマンドラインによるリモート管理で問題が発生するおそれがあります。
- 拡張時に Sybase インストール先ディレクトリのプロンプトが編集可能ですが変更しないでください Sybase Recovery Kit は、各常時におけるインストール先ディレクトリの変更をサポートしていません。ターゲット上のインストール先ディレクトリが異なる場合、`extend` 操作が失敗することがあり、基本操作 (`restore`、`remove`、および `monitoring`) がネイティブに悪影響を受けることがあります。Sybase Recovery Kit は、プライマリのインストール先ディレクトリを使用してデフォルト値を設定し、リソースが拡張されるすべてのサーバ上でこのデフォルト値が使用されます。

[Properties] ページの問題:

- [Properties] ペインに [update user/password] の画像が表示されません正しい画像の代わりに小さい四角がツールバーに表示されます。この四角を選択すると、**User/Password Update Wizard** が起動されます。

ログの問題:

- Sybase Kit の `create` と `extend` のログメッセージが記録されません `create` と `extend` 操作のエラーおよび情報は、操作時に UI ウィンドウにのみ表示されます。トラブルシューティングに必要な場合、これらのメッセージを UI からコピーして保存できます。
- Sybase Kit と Core でログのフォーマットが異なります。Sybase Kit のログ出力の表示は、Core の他のログ出力の表示と一貫性がありません。これは、アプリケーション保護のエラーの原因にはなりませんが、ログビューアやログ解析ツールを使用する場合に特別な解析やロジックが必要になることがあります。

Sybase Monitor Server は、Sybase 15.7 と SIOS Protection Suite の組み合わせではサポートされていません。Sybase 15.7 で Sybase Monitor Server プロセスを設定する場合、Generic Application (`gen/app`) リソースを使用してこのサーバプロセスを保護する必要があります。

Remove コマンドが SLES 11 上の Sybase から認識されません

バックアップサーバで Sybase を in service にする場合、はじめにプライマリサーバで out of service にする必要があります。Sybase からこのコマンドを認識するようにするには、行に「`locales/locales.dat`」を追加し、SIOS Protection Suite が「`POSIX`」を「`vendor_locale`」として使用して、リモートで Sybase コマンドを実行するようにします。

例:

```
locale = POSIX, us_english, utf8
```

GUIトラブルシューティング

LifeKeeper GUI をリモートシステムから設定する際に問題が発生した場合は、以下のいずれかのトピックを参照してください。

[Java プラグイントラブルシューティング](#)

[アプレットトラブルシューティング](#)

[ネットワーク関連トラブルシューティング\(GUI\)](#)

ネットワーク関連トラブルシューティング (GUI)

LifeKeeper は GUI クライアントとサーバの通信に Java RMI (Remote Method Invocation) を使用します。問題となりうる要素の一部は RMI に関連し、それ以外は一般的なネットワークの設定に関する問題です。

Windows プラットフォームでの論理接続の遅延

Sun FAQ から:

最も蓋然性が高いのは、ホストのネットワーク設定が誤りというものです。RMI は Java API ネットワーククラス、特に `ava.net.InetAddress` を使用します。これは、アドレスマッピングおよびホスト名へのアドレスに対して両方のホストに TCP/IP ホスト名のルックアップを実行させます。Windows では、ルックアップ機能はネイティブ Windows ソケットライブラリで実行されるので、遅延は RMI ではなく、Windows ライブラリで発生するものです。ホストが DNS を使用するように設定されている場合、これは、通信に関連するホストについて認識しないという DNS サーバの問題となる可能性があります。その場合、DNS ルックアップのタイムアウトが発生します。このケースに当てはまる場合は、ファイル `windows\system32\drivers\etc\hosts` で関連ホスト名/アドレスをすべて指定してください。通常のホストファイルのフォーマットを次に示します。

IP アドレス サーバ名称

例:

`208.2.84.61 homer.somecompany.com homer`

これで、最初のルックアップにかかる時間を短縮できるはずです。

また、サブネットマスクとゲートウェイアドレスの設定が誤っていると、接続の遅延や障害を引き起こす可能性があります。これらの設定が正しいことをネットワーク管理者に確認してください。

モデムからの実行:

サーバが存在するネットワークにモデムで (PPP または SLIP を使用して) 接続する場合、コンピュータは一時的な IP 番号を操作用に取得します。この一時的な番号は、ホスト名がマップしたものではない可能性があります (ホスト名が何かにマップしている場合)。そのため、この場合は、IP のみで通信するようにサーバに指示する必要があります。これには、モデム接続ウィンドウを開いて一時的な IP 番号を取得します。この番号を使用して、GUI クライアントのホスト名プロパティを設定します。

プライマリネットワークインターフェースのダウン:

プラグインでブラウザのホスト名を設定するには、**[Java Plug-In Control Panel]**を開き、**[Java Run Time Parameters]**に以下の値を追加してクライアントのホスト名を設定します。

```
-Djava.rmi.server.hostname=<MY_HOST>
```

HotJava ブラウザのホスト名を設定するには、hotjava コマンドラインに以下の値を追加します。

```
-Djava.rmi.server.hostname=<MY_HOST>
```

たとえば、以下ようになります。

```
-Djava.rmi.server.hostname=153.66.140.1
```

プライマリネットワークインターフェースのダウン:

LifeKeeper GUI は、GUI クライアントと GUI サーバの通信を維持するために Remote Method Invocation (RMI) を使用します。ほぼどのような場合でも、プライマリネットワークインターフェースを介してサーバへの接続が確立されます。つまり、サーバのプライマリインターフェースがダウンした場合、接続は失われ、GUI クライアントに [Unknown] というサーバの状態が表示されます。

この問題の唯一の解決策は、サーバのプライマリインターフェースを再び有効にすることです。また、RMI の制限のため、マルチホームサーバ (複数のネットワークインターフェースを備えたサーバ) でこの問題を解決することはできません。

ホストへのルートが存在しない例外:

ホストに接続できなかったため、ソケットをリモートホストに接続できませんでした。これは通常、ネットワークのローカルサーバとリモートホストの間のリンクの一部がダウンしたか、ホストがファイアウォールの後ろにあることを意味します。

不明なホストの例外:

LifeKeeper GUI クライアントとサーバは、通信に Java RMI (Remote Method Invocation) 技術を使用します。RMI が正常に動作するために、クライアントとサーバは解決可能なホスト名または IP アドレスを使用する必要があります。解決不可能な名前、WINS 名、修飾されていない DHCP 名を使用した場合、Java は UnknownHostException を送じます。

このエラーメッセージは、以下の条件でも発生する可能性があります。

- サーバ名が存在しない場合。サーバ名の誤記がないか確認してください。
- 設定された DHCP サーバが、RMI サーバが実際に存在するドメインではなく、リゾルバドメインのドメイン名になるように RMI サーバの完全修飾ドメイン名を設定している場合。この場合、サーバの DHCP ドメインの外側の RMI クライアントは、不正なドメイン名のためにサーバにアクセスできません。
- サーバが、Windows Internet Naming Service (WINS) を使用するように設定されたネットワーク上にある場合。DNS にのみ依存しているホストは、WINS の下に登録されたホストにアクセスできない場合があります。
- RMI クライアントとサーバがファイアウォールをはさんだ反対側にある場合。ファイアウォールの外側に RMI

Windows から:

クライアント、内側にサーバがある場合、クライアントはサーバに対してリモート呼び出しを実行できません。

LifeKeeper GUI を使用している場合、クライアントによって提供されたホスト名はサーバから解決できるものであり、サーバからのホスト名はクライアントによって解決できるものである必要があります。LifeKeeper GUI はこの例外を捕捉し、ユーザに警告します。クライアントがサーバのホスト名を解決できない場合、この例外が捕捉され、メッセージ 115 が表示されます。サーバがクライアントのホスト名を解決できない場合、この例外が捕捉され、メッセージ 116 が表示されます。どちらのメッセージにも、実行が試された未修飾ホスト名を指定する Java 例外の一部が含まれています。

下記に、ホスト名の解決が正常に機能していることをテストまたは検証するために使用できる手順をいくつか示します。

Windows から:

1. Linux サーバとの通信の確認

DOS プロンプトから、ホスト名を使用してターゲットを ping します。

```
ping <TARGET_NAME>
```

たとえば、以下ようになります。

```
ping homer
```

ターゲットの修飾されたホスト名と IP アドレスをリストする応答が表示されるはずですが。

2. 正しい設定の確認

- DNS の設定を確認するか、ネットワークに DNS サーバをインストールします。
- *ControlPanel->Network->Protocols->TCP/IP* の設定を確認します。これらの設定が正しいことをネットワーク管理者に確認してください。

[DNS] タブのホスト名は、ローカルネームサーバで使用されているものと一致している必要があります。これは、GUI エラーメッセージで指定したホスト名とも一致している必要があります。

- ローカルホストおよびその接続先となる LifeKeeper サーバのエントリを含める形で hosts ファイルを編集してください。

Windows 95/98 システムでは、hosts ファイルは以下ようになります。

```
%windir%\HOSTS (for example, C:\WINDOWS\HOSTS).
```

注記: Windows 95/98 では、hosts ファイルの最後のエントリがキャリッジリターン (CR) またはラインフィード (LF) で終わっていない場合、hosts ファイルはまったく読み取られません。

Windows NT システムでは、hosts ファイルは以下ようになります。

```
%windir%\System32\DRIVERS\ETC\HOSTS  
(for example, C:\WINNT\System32\DRIVERS\ETC\HOSTS).
```

Linux から:

たとえば、システムが *HOSTCLIENT.MYDOMAIN.COM* と呼ばれ、IP アドレスとして *153.66.140.1* を使用している場合、*hosts* ファイルに次のエントリを追加します。

```
153.66.140.1 HOSTCLIENT.MYDOMAIN.COM HOSTCLIENT
```

3. GUI クライアントで使用するホスト名プロパティを設定してください。プラグインでブラウザからホスト名を設定するには、**[Java Plug-In Control Panel]** を開き、**[Java Run Time Parameters]** に以下の値を追加してクライアントのホスト名を設定します。

```
Djava.rmi.server.hostname=<MY_HOST>
```

4. Microsoft のネットワーク関連のバッチを www.microsoft.com で確認してください。

Linux から:

1. ホスト名または IP アドレスを使用して Linux からターゲットサーバを ping し、他のサーバとの通信を確認します。

```
ping <TARGET_NAME>
```

たとえば、以下ようになります。

```
ping homer
```

ターゲットの修飾されたホスト名をリストする応答が表示されるはずですが。

2. ホスト名または IP アドレスで **ping** を実行し、クラスタ内の各サーバでローカルホストが解決可能であることを確認します。DNS が実装されていない場合、*/etc/hosts* ファイルを編集し、ローカルホスト名のエントリを追加します。このエントリで、ローカルサーバの IP アドレスまたはデフォルトエントリ (127.0.0.1) をリストできます。
3. DNS が NIS の前に指定されていることを確認します。*/etc/nsswitch.conf* の *hosts* 行で DNS を NIS の前に置く必要があります。また、*/etc/resolv.conf* は正しく設定された DNS サーバを指す必要があります。
4. DNS を実装しない場合、または他の方法がうまくいかない場合は、*/etc/hosts* ファイルを編集し、ホスト名のエントリを追加します。
5. GUI クライアントで使用するホスト名プロパティを設定してください。これは、管理者ごとに変更する必要があります。

プラグインでブラウザからホスト名を設定するには、**[Java Plug-In Control Panel]** を開き、**[Java Run Time Parameters]** に以下の値を追加してクライアントのホスト名を設定します。

```
-Djava.rmi.server.hostname=<MY_HOST>
```

HotJava ブラウザからホスト名を設定するには、*hotjava* コマンドラインに以下の値を追加します。

```
-Djava.rmi.server.hostname=<MY_HOST>
```

たとえば、以下ようになります。

```
-Djava.rmi.server.hostname=153.66.140.1
```

```
-Djava.rmi.server.hostname= homer.somecompany.com
```

X Window Server に接続できない:

LifeKeeper GUI アプリケーションを telnet セッションから実行している場合、GUI クライアントが LifeKeeper サーバで X Window Server にアクセスできることを確認する必要があります。LifeKeeper サーバは GUI クライアントのホスト名またはネットワークアドレスを解決できる必要があります。

LifeKeeper サーバに対して telnet を実行して LifeKeeper GUI アプリケーションを実行した場合、DISPLAY 環境変数にはクライアントのホスト名と表示番号を含める必要があります。たとえば、Server1 というサーバに Client1 というクライアントから telnet を実行した場合、DISPLAY 環境変数は Client1:0 に設定される必要があります。LifeKeeper GUI アプリケーションを実行した場合、Client1 の DISPLAY 名に出力が送信されます。Client1 が X Window Server にアクセスできない場合、例外が発生して LifeKeeper GUI アプリケーションは失敗します。

LifeKeeper GUI をアプリケーションとして起動したときに、X Window Server に接続できない、またはクライアント DISPLAY 名を開くことができないというエラーが発生した場合は、以下の手順を実行してください。

1. ホスト名または IP アドレスを使用して表示変数を設定します。たとえば、以下のようになります。

```
DISPLAY=Client1.somecompany.com:0
```

```
DISPLAY=172.17.5.74:0
```

2. xhost または xauth コマンドを使用し、クライアントが LifeKeeper サーバで X Window Server に接続できることを確認します。
3. クライアント用の DNS エントリを追加するか、クライアント用のエントリを LifeKeeper サーバのローカルホストファイルに追加します。LifeKeeper サーバからクライアントに対して、ホスト名または IP アドレスを使用して ping を実行し、クライアントとの通信を確認します。

コミュニケーションパスの稼働と停止

コミュニケーションパスの停止と稼働が繰り返される場合 (LifeKeeper GUI で Alive、Dead、Alive というように表示される場合)、ハートビートの設定がクラスタ内のすべてのサーバで同じ値に設定されていない可能性があります。

この状態は、いずれか一方のサーバにある LifeKeeper デフォルトファイル `/etc/default/LifeKeeper` で設定名に誤記がある場合にも発生する可能性があります。

推奨される対策

1. クラスタ内のすべてのサーバで LifeKeeper を停止します。
2. クラスタ内の各サーバで、`/etc/default/LifeKeeper` にある `LCMHBEATTIME` 設定と `LCMNUMHBEATS` 設定の値とスペルを確認します。設定値、スペルミスの無いことを各ノードで確認します。
3. クラスタ内のすべてのサーバで LifeKeeper を再起動します。

不完全なリソースの作成

インスタンスの一部のみが作成された状態で、リソース設定プロセスが中断された場合、階層を再設定する前に、手動でクリーンアップする必要があります。LifeKeeper GUI を使用し、一部が作成されたリソースを削除してください。手順については、[すべてのサーバからの階層の削除](#)を参照してください。階層リストにこれらのリソースが含まれていない場合、`ins_remove(LCDI-instances(1M))`を参照) および `dep_remove(LCDI-relationship(1M))` を使用し、部分的な階層をクリーンアップしなければならない可能性もあります。

不完全なリソースの優先順位の変更

LifeKeeper の階層は、親子の関係によって関連付けられたすべてのリソースとして定義されています。複数の親を持つリソースの場合、GUI と階層のすべてのリソースを区別することは一概に簡単とも言えなくなります。階層の整合性を保持するには、サーバごとに階層内のすべてのリソースに対して優先順位を変更する必要があります。[OK] または [Apply] ボタンを押した後で選択される階層のすべてのルートリソースを表示することで、GUI はこの要件を強制します。この時点で、すべてのルートを受け付けるか、操作をキャンセルするかを選択できます。ルートのリストを受け付けた場合、新しい優先順位の値が階層内のすべてのリソースに割り当てられます。

その階層の [Resource Properties] ダイアログが表示されている間、他の変更を階層に加えていることを確認する必要があります。[Resource Properties] ダイアログの優先順位を編集する前に、LifeKeeper に加えられた変更が動的にダイアログで更新されます。ただし、変更を加えると、基本的な変更が LifeKeeper で加えられた場合でも、ダイアログの値は凍結されます。[Apply] または [OK] ボタンをクリックした後でのみ、変更が加えられたことが通知されるので、優先順位の変更操作は要求どおりに進みません。

複数の優先順位の変更を伴う優先順位の変更操作時に、復旧できないエラーの可能性を最低限に抑えるには、プログラムは、一度に1つのサーバに対して個別に行われる一連の変更として、複数の優先順位の変更操作を実行します。また、操作時に優先順位の競合を防ぐために、必要に応じて一時的な値がプロパティに割り当てられます。この一時的な値は、最大許容値 999 を超えるもので、優先順位の変更中に一時的に GUI に表示されることもあります。操作が完了すると、一時的な値はすべて、要求された値に置き換えられます。エラーが発生し、優先順位の値をロールバックできない場合、一時的な優先順位の値の一部がそのまま残る可能性もあります。この場合は、下記の推奨手順に従って階層を修復してください。

一貫した状態への階層のリストア

優先順位の変更操作の間にエラーが発生し、操作を完了できない場合、優先順位は不整合の状態のまま残る可能性があります。エラーは、システムやコミュニケーションパスの障害を含め、さまざまな理由で発生します。操作が開始された後や完了する前にエラーが発生し、プログラムが前の優先順位にロールバックできなかった場合、操作中にエラーがあったこと、および前の優先順位を restore できなかったことを示すメッセージが表示されます。この場合、以下の処置を実行し、階層を一貫性のある状態に restore する必要があります。

1. 可能であれば、問題の原因を特定します。システムまたはコミュニケーションパスの障害を確認します。優先順位管理プログラムの実行中に、その他の操作が行われていないことを確認します。
2. 可能であれば、問題の原因を修正してから先に進みます。たとえば、階層を修復する前に、障害が発生したシステムまたはコミュニケーションパスを restore する必要があります。
3. [Resource Properties] ダイアログから操作を再試行します。

4. [Resource Properties] ダイアログから変更できない場合は、コマンドライン `hry_setpri` を使用して階層を修復するとより簡単かもしれません。このスクリプトを使用すると、一度に1つのサーバに対して優先順位を変更できます。このスクリプトは、GUI からは実行できません。
5. 修復を実行したら、階層が存在するすべてのサーバに対して `eqv_list` コマンドを実行し、階層のすべてのリソースに対して返された優先順位の値を調べ、LifeKeeper データベースがすべてのサーバで一貫していることを確認します。
6. 最終的に、階層を修復できない場合は、階層を削除して再作成する必要がある可能性もあります。

階層の設定中に共有ストレージが見つからない

リソースの階層を設定中に、LifeKeeper が「No shared storage」(共有ストレージがありません) というメッセージをレポートする状況がいくつかあります。

考えられる原因: ストレージを共有するサーバー間でコミュニケーションパスが定義されていません。共有ストレージデバイスで階層が設定されている場合、LifeKeeper は、クラスタ内の別のサーバを少なくとも1つ検証し、その共有ストレージにアクセスできることを確認します。

推奨される対策: LifeKeeper GUI または `lcdstatus (1M)` を使用し、コミュニケーションパスが設定されており、アクティブになっていることを確認します。

考えられる原因: ストレージを共有するサーバー間でコミュニケーションパスが機能していません。

推奨される対策: LifeKeeper GUI または `lcdstatus (1M)` を使用し、コミュニケーションパスが設定されており、アクティブになっていることを確認します。

考えられる原因: Linux が共有ストレージにアクセスできない。この原因としては、ドライバがロードされていないことや、ドライバがロードされたときにストレージの電源が入っていないこと、あるいはストレージデバイスが正しく設定されていないことなどが考えられます。

推奨される対策: `/proc/scsi/scsi` でデバイスが正しく定義されていることを確認します。

考えられる原因: LifeKeeper を起動する前にストレージが Linux で設定されていない。LifeKeeper の起動時に、すべての SCSI デバイスがスキャンされ、デバイスのマッピングが判別されます。LifeKeeper の起動後にデバイスが設定された (電源がオンにされた、接続された、またはドライバがロードされた) 場合、デバイスを設定して使用できるようにするには、LifeKeeper を停止してから再起動する必要があります。

推奨される対策: `$LKROOT/subsys/scsi/resources/hostadp/device_info` にデバイスがリストされていることを確認してください。`$LKROOT` は、デフォルトでは `/opt/LifeKeeper.` です。デバイスがこのファイルにリストされていない場合、LifeKeeper はそのデバイスを使用しません。

考えられる原因: ストレージがサポートされていない。サポートストレージ一覧には、LifeKeeper で動作がテストされ、サポートされている具体的な SCSI デバイスが列挙されています。ただし、このリストに含まれているのは、す

で知られているデバイスなので注意してください。LifeKeeper の要件を満たしているものの、SIOS Technology Corp. がテストしていないデバイスが存在する可能性もあります。

推奨される対策: `$LKROOT/subsys/scsi/resources/hostadp/device_info` にデバイスがリストされていることを確認してください。`$LKROOT` は、デフォルトでは `/opt/LifeKeeper` です。デバイスがこのファイルにリストされているものの、デバイス名の後に来る ID が「NU-」で始まる場合、LifeKeeper はデバイスから一意の ID を取得できなかったことを示します。一意の ID がない場合、LifeKeeper はデバイスが共有されているかどうかを判別できません。

考えられる原因: ストレージでは、デバイスを LifeKeeper で使用できるようにする前に、特定の LifeKeeper ソフトウェアをインストールする必要があります。たとえば、Raw I/O サポートを有効にするための `steeleye-ikRAW` キット、データレプリケーションを有効にするための `steeleye-ikDR` ソフトウェアなどです。

推奨される対策: 必要な LifeKeeper パッケージが各サーバにインストールされていることを確認します。ソフトウェアの要件については、[SPS for Linux リリースノート](#) を参照してください。

補足のヒント:

`test_ik(1M)` ツールを使用すると、ストレージおよび通信の問題のデバッグに役立ちます。

LifeKeeper サーバ障害からの復旧

LifeKeeper クラスタ内のサーバに、オペレーティングシステムの再インストールを（したがって LifeKeeper の再インストールも）必要とする障害が発生した場合、クラスタの各サーバからリソース階層を再拡張する必要があります。ただし、再インストールしたサーバとの共有イクイバレンシ関係がクラスタのサーバにある場合、LifeKeeper は、再インストールしたサーバへ既存のリソース階層を拡張することを許可しません。また、再インストールされたサーバには階層が実際には存在していないため、再インストールしたサーバから階層を拡張解除することもできません。

推奨される対策:

1. リソース階層が設定されている各サーバで、`eqv_list` コマンドを使用してすべての共有イクイバレンシのリストを取得します（詳細については、`LCDI-relationship` を参照してください）。

下記の例では、`server1` および `server2` に対する IP リソースの `iptag` のコマンドおよび結果の出力を示します。ここでは、`server2` が再インストールされたサーバ、`server1` が設定された階層です。

```
eqv_list -f:
server1:iptag:server2:iptag:SHARED:1:10
```

2. リソース階層が設定された各サーバで、`eqv_remove` を使用して、階層の各リソースのイクイバレンシ関係を手動で削除します（詳細については、`LCDI-relationship` を参照してください）。

たとえば、上記の手順 1 の例を基に、`server1` に対して以下のコマンドを実行します。

```
eqv_remove -t iptag -S server2 -e SHARED
```

3. 2つ以上のサーバがあるクラスタでは、これらのリソース階層のイクイバレンス関係が定義されているクラスタ内の各サーバに対して手順 1と2を繰り返します。
4. 最後に、GUIを使用し、リソース階層がin-serviceになっているサーバから再インストールされたサーバに各リソース階層を拡張します。

停止できないプロセスからの復旧

プロセスが停止不可の場合、LifeKeeperは共有ディスクパーティションをアンマウントできない可能性があります。そのため、リソースを別のシステムでIn Serviceにすることができません。停止できないプロセスから復旧する唯一の方法は、システムを再起動することです。

手動リカバリ時のパニックからの復旧

手動スイッチオーバー時にPANICになると、リカバリが不完全に終わる可能性があります。PANICまたはその他の大きなシステム障害が手動スイッチオーバー時に発生した場合、バックアップシステムへの完全自動リカバリは保証できなくなります。In Serviceになる必要があるすべてのリソースがIn Serviceであることをバックアップシステムで確認してください。In Serviceではないリソースがあった場合は、LifeKeeper GUIを使用して、そのリソースを手動でIn Serviceにします。手順については、[リソースをIn-Serviceする](#)を参照してください。

Out-of-Service 階層の復旧

LifeKeeperサーバの障害からの復旧の一環として、障害が発生したサーバで設定されているものの、サーバの障害時にどのサーバでもIn Serviceではなかったリソース階層が、障害時に最優先でAliveになったサーバで復旧されます。これは、障害が発生したサーバ、復旧中のサーバ、階層内の他のサーバを含め、Out of Serviceの階層が最後にどこでIn Serviceだったかには無関係です。

リソースタグ名の制限

タグ名の長さ

LifeKeeper内のすべてのタグは、256文字以内にする必要があります。

有効な特殊文字

- _ . /

タグの最初の文字に「.」および「/」を使用することはできません。

無効な文字

+ ; : ! @ # \$ * = 「スペース」

シリアル (TTY) コンソールの警告

シリアルコンソールデータパスの一部が信頼できない場合、または Out of Service になった場合、シリアル (RS-232 TTY) コンソールを使用するユーザは LifeKeeper サービスで深刻な問題に直面する可能性があります。操作中に、LifeKeeper はコンソールメッセージを生成します。設定に (標準的な VGA コンソールではなく) シリアルコンソールがある場合、これらのコンソールメッセージが確実に配信されるようにするために、LifeKeeper からエンドユーザーターミナルへのデータパス全体が機能している必要があります。

ターミナルの電源オフ、モデムの未接続、ケーブルのゆるみなど、データパスがつながっていない場合、Linux STREAMS ファシリティは、コンソールメッセージをキューに入れます。STREAMS キューがいっぱいになった場合、Unix カーネルは、STREAMS バッファキューにメッセージを入れる余地ができるまで LifeKeeper を保留にします。これにより、LifeKeeper がハングすることもあります。

注記: LifeKeeper 環境のシリアルコンソールは可能なかぎり避け、VGA コンソールを使用することを推奨します。シリアルコンソールを使用する必要がある場合、シリアルコンソールがオンになっていること、ケーブルとオプションのモデムが正しく接続されていること、メッセージが表示されていることを必ず確認してください。

システムが init 状態 S に遷移しているという警告

LifeKeeper が動作している場合、システムを直接、init 状態 S に切り替えしないでください。Linux の init システムの操作が原因で、こうした遷移が全 LifeKeeper プロセスの即時停止につながり、突発的な障害を発生させる可能性があります。この場合は、代わりに LifeKeeper を (lkstop で) 手動停止するか、システムを最初に init 状態 1 にしてから init 状態 S にしてください。

共有ストレージでスレッドがハングしているというメッセージ

デバイス確認スレッドがそれほど迅速に処理を完了していない場合、スレッドがハングしているというメッセージが LifeKeeper ログに記録されることがあります。これにより、リソースがあるサーバから別のサーバに移動し、さらに悪いケースでは、サーバが停止する可能性があります。

説明

(/etc/default/LifeKeeper)の FAILFASTTIMER は、各デバイスが正常に動作していること、および特定のシステムによって所有されているすべてのリソースがそのシステムからアクセス可能で、そのシステムに所有されていることを確認するための秒数を定義します。FAILFASTTIMER は、この所有権を確定し、データの信頼性を最大限に確保するために、可能なかぎり小さくする必要があります。ただし、デバイスがビジー状態で、負荷がピークの場合、指定した時間内で応答できない可能性もあります。デバイスの操作が FAILFASTTIMER よりも長かかっている場合、LifeKeeper はデバイスがハングしている可能性を検討します。FAILFASTTIMER の時間を 3 回繰り返してもデバイスが応答しない場合、LifeKeeper は、デバイスに障害が発生したものとみなして、リカバリを実行します。リカバリプロセスは、SCSIERROR の設定で定義します。SCSIERROR の設定によっては、ローカルリカバリを実行し、失敗した場合はスイッチオーバーを実行するために sendevent が発行されることもあります。この操作がない場合、システムが停止するおそれもあります。

推奨される対策:

ハングメッセージがまれにエラーログに出力され、もうハングしていないというメッセージがそれに続く場合、さらに括弧の数が常に1つの場合、それほど警戒する理由はありません。ただし、このメッセージが頻繁にログに記録され、数が2または3の場合、以下の2つの処置が必要になる可能性があります。

- ストレージの負荷を減らすことを試みる。ストレージの処理に FAILFASTTIMER (デフォルトでは、5秒または15秒を3回)の3倍の時間がかかっている場合、ストレージに対する負荷を考慮し、I/Oの長い遅延を避けるために負荷を分散する必要があります。これにより、LifeKeeperは、デバイスを頻繁に確認できるようになり、さらにそのデバイスを使用しているアプリケーションのパフォーマンスも向上します。
- 負荷を減らすことができない場合、FAILFASTTIMERをデフォルトの5秒から増やすことができます。この値は、できる限り低く抑える必要があります。そのため、メッセージがまったく表示されなくなるか、まれにしこ表示されなくなるまで、少しずつ値を増やしてください。

注記: FAILFASTTIMERの値が変更された場合、新しい値を有効にするために、LifeKeeperを終了し、再起動する必要があります。

Chapter 3: SIOS DataKeeper for Linux

はじめに

SIOS DataKeeper for Linux は、LifeKeeper 環境に統合 データミラーリング機能を提供します。この機能により、LifeKeeper リソースが共有 / 非共有ストレージ環境で動作可能になります。

[SIOS DataKeeper for Linux によるミラーリング](#)

[SIOS DataKeeper の仕組み](#)

SIOS DataKeeper for Linux によるミラーリング

SIOS DataKeeper for Linux は、共有ストレージを使用せずに可用性の高いクラスタ(SIOS LifeKeeper を使用)を構築したいお客様や、ビジネスに不可欠なデータをサーバ間でリアルタイムに複製したいお客様に別の方法を提供します。

SIOS DataKeeper は、同期または非同期のボリュームレベルのミラーリングを使用して、プライマリサーバ(ミラーソース)から 1 台以上のバックアップサーバ(ミラーターゲット)にデータを複製します。

DataKeeper の特長

SIOS DataKeeper には、以下の特長があります。

- TCP/IP ベースのローカルエリアネットワーク(LAN)またはワイドエリアネットワーク(WAN) 経由で、リモートの場所に高い信頼性、効率、整合性でデータをミラーリングできます。
- 同期と非同期のミラーリングをサポートします。
- 複製はファイルシステムの下ブロックレベルで実行されるので、関与するアプリケーションに対して透過的です。
- LifeKeeper と共に使用した場合、複数ターゲットへのカスケードフェイルオーバーも含めて、複数ターゲットへの同時ミラーリングをサポートします。
- 内蔵のネットワーク圧縮により、ワイドエリアネットワークでの最大スループットが向上します。
- 主要なファイルシステムをすべてサポートします(ファイルシステムのジャーナリングサポートの詳細については、[SPS for Linux リリースノート](#)の製品説明を参照してください)。
- ミラーリングしたデータにフェイルオーバーの保護を提供します。
- LifeKeeper のグラフィカルユーザインターフェースに統合されています。

- 他の LifeKeeper Application Recovery Kit をフルにサポートします。
- システムリカバリ時に、プライマリサーバとバックアップサーバとの間でデータを自動的に再同期します。
- 障害発生時には、仮想のシステムコンポーネントの健全性を監視し、ローカルリカバリを実行します。
- I/O フェンス用の Stonith デバイスをサポートします。詳細については、[STONITH](#) のトピックを参照してください。

同期ミラーリングと非同期ミラーリングの違い

同期ミラーリングと非同期ミラーリングの違いを理解すると、アプリケーション環境に適切なミラーリング方法を選択することができます。

同期ミラーリング

SIOS DataKeeper は、プライマリサーバとバックアップサーバに同時にデータを書き込む同期ミラーリング方法を使用して、リアルタイムミラーを実現します。書き込み動作のたびに、DataKeeper は書き込みをターゲットデバイスに転送し、リモート確認を受信してから I/O 完了を通知します。同期ミラーリングの長所は、データの保護レベルが高いことです。これは、常にデータのすべてのコピーを確実に同一にしているからです。ただし、リモート確認を待つために、パフォーマンスが低下することがあり、これは特に WAN 環境で発生します。

非同期ミラーリング

非同期ミラーリングでは、それぞれの書き込みがソースデバイスに対して行われ、次に、コピーがターゲットデバイスに送信されるキューに入れられます。これはつまり、任意の時点で、ソースからターゲットデバイスへの送信を待っている多数の書き込みトランザクションが存在する可能性があります。非同期ミラーリングの長所は、書き込みがプライマリディスクに到達した時点で確認されるため、パフォーマンスが高いことです。ただし、プライマリディスクに障害が発生した場合、非同期書き込みキュー内にある書き込みはターゲットに送信されないため、信頼性が低くなります。この問題を緩和するために、SIOS DataKeeper はプライマリディスクに対する個々の書き込みについて intent ログファイルにエントリを作成します。また、多量の書き込みが継続的に行われた場合、キューに書き込まれたデータの送出手を優先するため一時的に I/O パフォーマンスが低下する場合があります。

intent ログとは、プライマリとターゲットのミラー間で同期していないデータブロックを示すビットマップです。サーバの障害発生時に intent ログを使用すると、データ全体の再同期を回避できます。



補足: 非同期ミラーリングを作成するとき、DataKeeper はターゲットデバイスへの書き込みキューの数をデフォルトの 256 にセットするため、write-behind を値なしでセットします。LKDR_ASYNC_LIMIT パラメータに 2 以上の値がセットされている場合には、その値が用いられます。

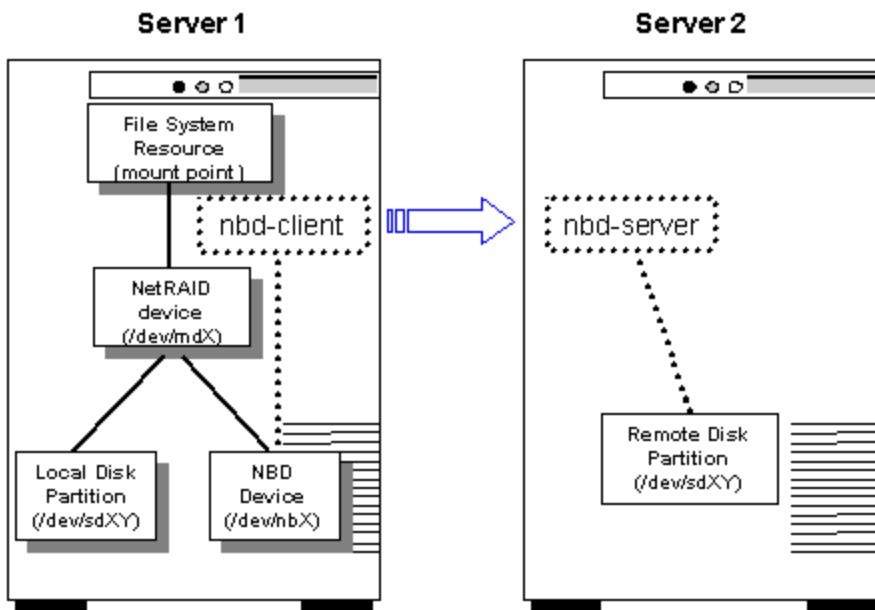
一度ターゲットデバイスへの書き込みキューがこの値に達すると、書き込みキューがこの値を下回るまでの間ミラーリングは同期モードとして動作します。

ブロックサイズが 4K である場合、この値がデフォルト (非同期書き込みキューの上限 256) であるならば、最大で 1 MB の転送待ちデータが発生します。

注記: intent ログは、同期と非同期の両方のミラーモードで使用できます。ただし、非同期ミラーリングの intent ログは、2.6.16 以降の Linux カーネルでのみサポートされます。

SIOS DataKeeper の仕組み

SIOS DataKeeper は、NetRAID デバイスを作成して保護します。NetRAID デバイスは RAID1 のデバイスであり、下図に示すようにローカルのディスクまたはパーティション、およびネットワークブロックデバイス(NBD)で構成されます。



LifeKeeper がサポートするファイルシステムは、その他すべてのストレージデバイスと同様に、NetRAID デバイスにマウントできます。この場合、ファイルシステムは複製されたファイルシステムと呼ばれます。LifeKeeper は、NetRAID デバイスと複製されたファイルシステムの両方を保護します。

ファイルシステムは、DataKeeper リソース階層を作成することにより作成されます。NetRAID デバイスを別のサーバに拡張するとNBD デバイスが作成され、2 台のサーバ間にネットワーク接続が確立されます。NBD 接続が確立されるとただちに、SIOS DataKeeper がデータの複製を開始します。

nbd-client プロセスがプライマリサーバで実行され、バックアップサーバで動作している nbd-server プロセスと接続します。

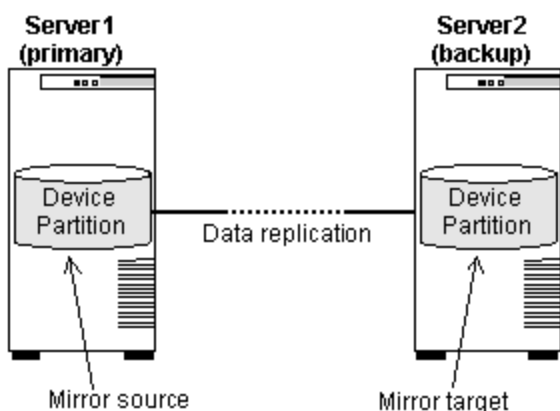
同期 (および再同期)

DataKeeper リソース階層は作成されてから拡張されるまでの間、デグレードモードです。つまり、データはローカルのディスクまたはパーティションにのみ書き込まれます。階層をバックアップ(ターゲット)システムに拡張すると、SIOS DataKeeper が2つのシステム間でデータを同期し、以降の書き込みはすべてターゲットに複製されます。どの時点でもデータが「非同期」になった場合(システムまたはネットワークの障害が発生した場合)、SIOS DataKeeper はソースとターゲットのシステムでデータを自動的に再同期します。intentログ(ビットマップファイル)を使用するようにミラーが設定されている場合、SIOS DataKeeper はintentログを使用して非同期の

データを特定するので、全体の再同期は不要です。intentログ(ビットマップファイル)を使用するようにミラーが設定されていない場合は、データ複製の中断後に全体の再同期が実行されます。

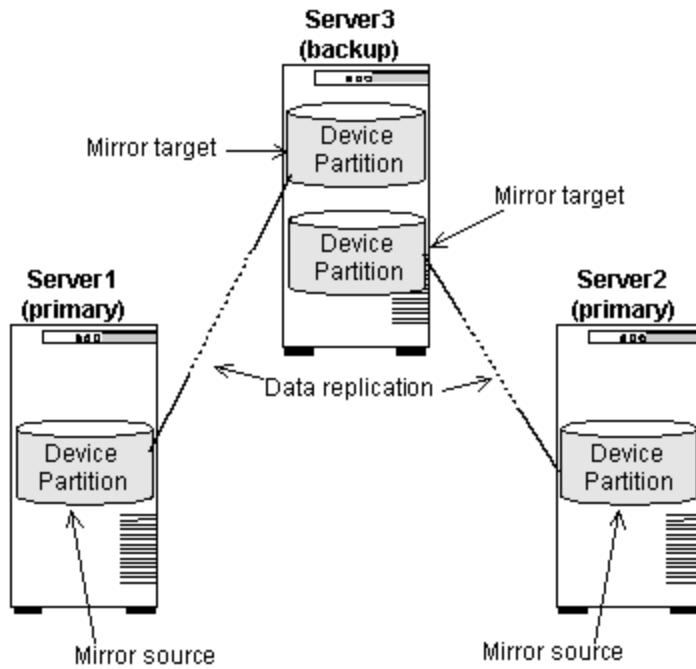
標準ミラーの構成

最も一般的なミラーの構成では、下図に示すように2台のサーバがあり、各サーバのローカルのディスクまたはパーティションとの間にミラーが確立されます。サーバ1は、ミラーソースを持つプライマリサーバです。サーバ2は、ミラーターゲットを持つバックアップサーバです。



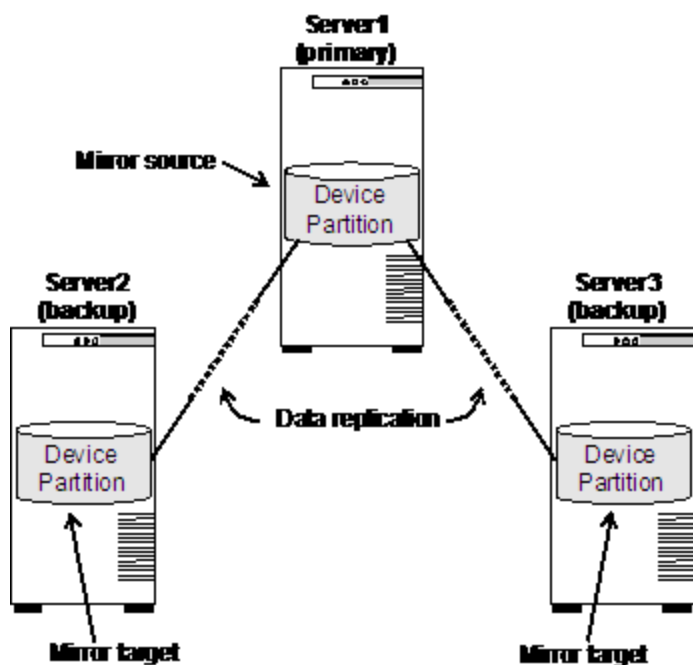
N+1 Configuration

前述した標準ミラーの構成の変形として一般的に使用される構成では、クラスタ内にある2台以上のサーバが共通のバックアップサーバにデータを複製します。この場合は、下図に示すように、各ミラーソースがバックアップサーバの個別のディスクまたはパーティションに複製する必要があります。



複数ターゲットの構成

適切な Linux のディストリビューションとバージョン 2.6.7 以降のカーネルと共に使用した場合、下図に示すように、SIOS DataKeeper は、プライマリサーバの 1 つのディスクまたはパーティションから複数のバックアップシステムにデータを複製することもできます。

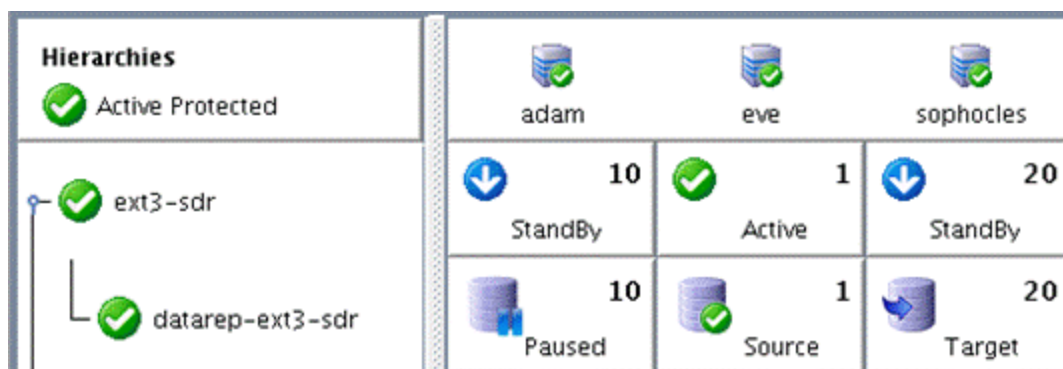


特定のソースのディスクまたはパーティションを最大 7 つのミラーターゲットに複製でき、各ミラーターゲットは別のシステムに存在する必要があります。つまり、ソースのディスクまたはパーティションを、同一ターゲットシステム上にある複数のディスクまたはパーティションにミラーリングすることはできません。

このタイプの構成では、LifeKeeper のカスケードフェイルオーバー機能を使用でき、保護するアプリケーションとそのデータに対して複数のバックアップシステムを提供できます。

SIOS DataKeeper リソース階層

以下の例に、LifeKeeper の GUI に表示される典型的な DataKeeper リソース階層を示します。



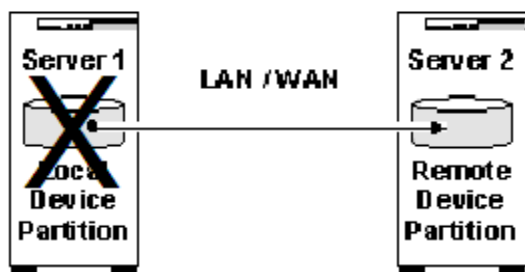
リソース *datarep-ext3-sdr* は NetRAID リソースであり、親リソース *ext3-sdr* はファイルシステムリソースです。本書の以降の部分では、「DataKeeper リソース」は両方のリソースを合わせたものを指すことに注意してください。ファイルシステムリソースは NetRAID リソースに依存するので、NetRAID リソースに対する動作はその上にあるファイルシステムにも影響します。

フェイルオーバーのシナリオ

以下の4つの例で、SIOS DataKeeperを使用するフェイルオーバーで何が起きるかを説明します。これらの例では、LifeKeeper for Linux クラスタは、サーバ1(プライマリサーバ)とサーバ2(バックアップサーバ)の2台のサーバで構成されます。

シナリオ 1

サーバ1からサーバ2へのミラーが正常に完了し、その後サーバ1が動作不能に陥る。



結果: フェイルオーバーが発生します。サーバ2がプライマリサーバの役割を担当し、サーバ1が再び動作可能になるまでデグレードモード(バックアップなし)で動作します。サーバ1が再び動作可能になると、SIOS DataKeeperがサーバ2からサーバ1への再同期を開始します。2.6.18以前のカーネルでは、全体の再同期が実行されます。2.6.19以降のカーネル、またはRed Hat Enterprise Linux 5.4の2.6.18-164以降のカーネル(またはRed Hat 5.4以降のサポートする派生カーネル)では、部分的な再同期が実行されます。つまり、ソースとターゲットにあるビットマップファイルに記録された変更部分についてのみ同期が必要です。

注記: SIOS DataKeeperは、現在ミラーソースとして動作しているサーバに以下のフラグをセットします。

```
$LKROOT/subsys/scsi/resources/netraid/$TAG_last_owner
```

サーバ1がサーバ2にフェイルオーバーすると、このフラグがサーバ2にセットされます。このため、サーバ1が動作を再開すると、SIOS DataKeeperはこの最終オーナーフラグをサーバ1から削除します。その後、サーバ2からサーバ1にデータの再同期を開始します。

シナリオ 2

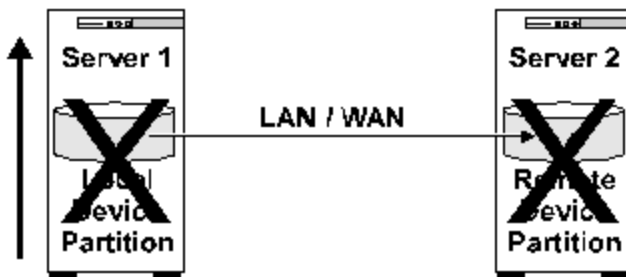
シナリオ1で、サーバ2(プライマリサーバである状態)が、サーバ1(この時点ではバックアップサーバ)との再同期中に動作不能になる。

シナリオ 3

結果: 再同期プロセスが正常に完了しなかったため、データが破損している可能性があります。この結果、LifeKeeper は DataKeeper リソースをサーバ 1 にフェイルオーバーしません。サーバ 2 が動作可能になった場合にのみ、LifeKeeper はサーバ 2 で DataKeeper リソースをサービス中 (ISP) にします。

シナリオ 3

サーバ 1 (プライマリ) とサーバ 2 (ターゲット) の両方が動作不能になる。サーバ 1 (プライマリ) が最初に動作可能になる。



結果: サーバ 1 は、DataKeeper リソースを in service にしません。この理由は、停止してからオンラインに戻ったソースサーバは、ターゲットと通信できないからです。ソースサーバは以下のタグをセットします。

```
$LKROOT/subsys/scsi/resources/netraid/$TAG_data_corrupt
```

これは、正しくない方向へのデータ同期を防止する安全策です。この場合、サーバ 1 でミラーを強制的にオンラインにする必要があります。つまり、サーバ 1 の `data_corrupt` フラグを削除し、リソースを in service にします。[ミラーを強制的にオンラインにする](#)を参照してください。

注記: `$TAG_data_corrupt` フラグを削除する前に、サーバ 1 が最終のプライマリサーバであることを確認する必要があります。サーバ 1 が最終のプライマリサーバでない場合、データが破損する可能性があります。これは、`last_owner` フラグの有無で確認できます。

シナリオ 4

サーバ 1 (プライマリ) とサーバ 2 (ターゲット) の両方が動作不能になる。サーバ 2 (プライマリ) が最初に動作可能になる。



シナリオ4

結果: LifeKeeper は、サーバ2 の DataKeeper リソースを ISP にしません。サーバ1 が動作可能になると、LifeKeeper はサーバ1 の DataKeeper リソースを ISP にします。

Chapter A: インストールと設定

DataKeeper リソースを設定する前に

以下のトピックには、DataKeeper リソースの作成と管理を行う前に考慮が必要な情報があります。また、3種類のDataKeeper リソースについても説明しています。LifeKeeper Core のリソース階層を設定する手順については、[LifeKeeper の設定](#) セクションを参照してください。

ハードウェアとソフトウェアの要件

SIOS DataKeeper をインストールするには、LifeKeeper の構成が次の要件を満たしている必要があります。

ハードウェアの要件

- **サーバ** - LifeKeeper for Linux をサポートする 2 台以上のサーバ。
- **IP ネットワークインターフェースカード** - 各サーバにネットワークインターフェースカードが 1 つ以上必要です。ただし、LifeKeeper クラスタには 2 つのコミュニケーションパスが必要です。独立した 2 つのサブネットを使用する 2 つの分離した LAN ベースのコミュニケーションパスが推奨され、これらの 1 つ以上をプライベートネットワークとして構成する必要があります。ただし、TCP と TTY を組み合わせて使用することもできます。

注記: ソフトウェアミラーリングの特性により、サーバ間のネットワークトラフィックが多くなる可能性があります。このため、SIOS DataKeeper のデバイス用に個別のプライベートネットワークを実装することが推奨されます。この実装には、各サーバに追加のネットワークインターフェースカードが必要になることがあります。

- **ディスクまたはパーティション** - ソースとターゲットのディスクまたはパーティションとして動作する、プライマリサーバとバックアップサーバのディスクまたはパーティション。ターゲットのディスクまたはパーティションは、ソースのディスクまたはパーティション以上のサイズである必要があります。

注記: SIOS Data Replication 7.1.1 のリリースから、パーティションが作成されていないディスク全体 (`/dev/sdd`) の複製が可能になりました。旧バージョンの SIOS Data Replication では、ディスクを複製するには、パーティションを作成する必要がありました (`/dev/sdd1` のような 1 つの大きいパーティションの場合でも)。SIOS Data Replication 7.1.1 からこの制限が取り除かれました。

ソフトウェアの要件

- **オペレーティングシステム** - SIOS DataKeeper は、Linux カーネル 2.6 をベースにする主要な Linux のディストリビューションと共に使用できます。サポートするディストリビューションのリストについては、SPS for Linux リリースノートを参照してください。非同期ミラーリングとインテントログは、2.6.16 以降の Linux カーネルを使用するディストリビューションでのみサポートされます。複数のターゲットのサポート (複数のミラーターゲットのサポート) には、2.6.7 以降の Linux カーネルが必要です。

- **LifeKeeper Installation スクリプト** - 多くの場合、以下のパッケージをインストールする必要があります (特定の SIOS DataKeeper の要件については、SPS for Linux リリースノートの「製品要件」セクションを参照してください)。

HADR-generic-2.6

SIOS DataKeeper をインストールする前に、LifeKeeper クラスタの各 サーバにこのパッケージをインストールする必要があります。HADR パッケージは SPS のインストールイメージファイル内にあり、Installation の **setup** スクリプトにより自動的に適切なパッケージがインストールされます。

- **LifeKeeper ソフトウェア** - 各サーバに同じバージョンの LifeKeeper Core をインストールする必要があります。また、使用を計画している同じバージョンの Recovery Kit も各サーバにインストールする必要があります。特定の SPS の要件については、SPS for Linux リリースノートを参照してください。
- **SIOS DataKeeper ソフトウェア** - SPS クラスタの各サーバには SIOS DataKeeper ソフトウェアが必要です。SIOS DataKeeper のインストールとアンインストールの手順については、SPS for Linux インストールガイドを参照してください。

全般的な設定

- ターゲットのディスクまたはパーティションのサイズ(バックアップサーバ上)は、ソースのディスクまたはパーティションのサイズ(プライマリサーバ上)以上である必要があります。
- DataKeeper リソースを作成して拡張すると、同期プロセスによりターゲットのディスクまたはパーティションに存在するデータが削除され、ソースのパーティションにあるデータに置き換えられます。

ネットワーク設定

- 各ペアのサーバ間でデータのレプリケーション用に選択するパスは、あらかじめそれらのサーバ間の LifeKeeper コミュニケーションパスとしても設定されている必要があります。ネットワークパスを変更する方法については、データレプリケーションパスの変更を参照してください。
- DataKeeper リソースを設定するときには、ローカルリカバリを有効にしている LifeKeeper IP リソースがすでに使用しているインターフェース/アドレスの使用は避けてください。例えば、LifeKeeper IP リソースがインターフェース *eth1* に構成されており、インターフェース *eth2* でのローカルリカバリが有効にされている場合、*eth1* と *eth2* のいずれについても DataKeeper リソースによる使用を避ける必要があります。ローカルリカバリを有効にすると、バックアップインターフェースへのスイッチオーバー中にインターフェースが無効になるので、SIOS DataKeeper に障害が発生することがあります。
- このリリースの SIOS DataKeeper は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、DataKeeper リソースの上に存在する他の LifeKeeper リソースにも適用されます。
- Fusion-io を使用する場合のネットワーク設定情報については、[Fusion-io を使用するクラスタ化](#)の「ネットワーク」セクションを参照してください。

データレプリケーションパスの変更

LK 7.1 から、`lk_chg_value` を使用して、ミラーのレプリケーションパスの IP アドレスを変更できるようになりました。例えば、ミラーのレプリケーションパスを IP アドレスの 192.168.0.1 から 192.168.1.1 に変更するには、以下の

操作を行ってください。

1. `/etc/init.d/lifekeeper stop-nofailover` (`lk_chg_value` は、LifeKeeper の動作中は実行できません)。
2. `lk_chg_value -o 192.168.0.1 -n 192.168.1.1`
3. `/etc/init.d/lifekeeper start`

このIPアドレスを使用するミラーに含まれるすべてのサーバで、これらのコマンドを実行してください。

注記: このコマンドは、該当アドレスを使用するコミュニケーションパスも変更します。

ネットワーク帯域幅の要件の特定

SIOS DataKeeper をインストールする前に、現在の構成の複製に仮想マシンを使用するか、物理的な Linux サーバを使用するかによりネットワーク帯域幅の要件を特定する必要があります。仮想マシン (VM) を使用する場合は、[Linux システム\(物理または仮想\)の更新頻度の測定](#) 方法を使用して、複製を計画している仮想マシンの変化率を測定してください。この値は、仮想マシンの複製に必要なネットワーク帯域幅を表します。

ネットワーク帯域幅の要件を特定した後、ネットワークが最大のパフォーマンスを発揮するように構成してください。ネットワーク帯域幅の要件が、現在使用できるネットワーク能力を超えている場合は、以下のオプションを1つ以上検討しなければならない可能性があります。

- SIOS DataKeeper (または可能な場合はネットワークハードウェア) の圧縮を有効にする。
- ネットワーク能力を増強する。
- 複製するデータ量を低減する。
- 一時データおよびスワップファイル用に、複製しないローカルのストレージリポジトリを作成する。
- 毎日、ピーク時以外に複製を手動でスケジュールする。

Linux システム(物理または仮想)での変化率の測定

DataKeeper for Linux は、使用できるネットワーク内でデータを複製できます。マルチサイト、すなわち広域ネットワーク (WAN) 構成では、「ソースパーティションが1日中更新されるときに、パーティションを正常に複製してミラーをミラーリング状態に維持するために十分な帯域幅があるか」という質問に対して特別な検討が必要です。

ミラーがミラーリング状態でない場合にはパーティションのスイッチオーバーは許可されないため、ミラーをミラーリング状態に維持することが重要です。

SIOS DataKeeper はデータを非同期キューに追加することにより、短期間に急増した書き込み動作を処理します。ただし、長期間にわたって複製されるすべてのボリュームのディスク書き込み動作の合計が、平均して DataKeeper とネットワークが送信できる変化量を下回ることを確認してください。

ネットワーク能力が不十分なためにディスクの変化率に対処できず、非同期キューがいっぱいになった場合、ミラーは同期動作に戻ります。これによりソースサーバのパフォーマンスに悪影響を及ぼすことがあります。

基本変化率の測定

以下のコマンドを使用して、ミラーリングするファイルまたはパーティションを特定してください。例えば /dev/sda3 を使用して、1日に書き込まれたデータ量を測定します。

```
MB_START=`awk '/sda3 / { print $10 / 2 / 1024 }' /proc/diskstats`
```

1日後

```
MB_END=`awk '/sda3 / { print $10 / 2 / 1024 }' /proc/diskstats`
```

1日の変化率(単位:MB)はMB_END - MB_START で得られます。

SIOS DataKeeper が1日にミラーリングできるおおよその量は以下のとおりです。

T1(1.5 Mbps) - 14,000 MB/日 (14 GB)

T3(1.5 Mbps) - 410,000 MB/日 (410 GB)

ギガビット(1 Gbps) - 5,000,000 MB/日 (5 TB)

詳細変化率の測定

変化率を収集する最良の方法は、一定期間(例:1日)ディスクの書き込み動作をログに記録して、ディスクの書き込みのピーク期間を特定することです。

ディスクの書き込み動作を追跡するには、システムのタイムスタンプをログに記録して /proc/diskstats のダンプを行う cron ジョブを作成してください。例えば、2分間隔でディスクの統計値を収集するには、/etc/crontab に以下のリンクを追加します。:

```
*/2 * * * * root ( date ; cat /proc/diskstats ) >> /path_to/filename.txt
```

1日、1週間などの期間が経過した後、cron ジョブを無効にし、得られたデータファイルを安全な場所に保存します。

収集した詳細変化率データの解析

roc-calc-diskstats ユーティリティは、前述の手順で収集したデータを解析します。このユーティリティは、長期間ログに記録された出力を持つ /proc/diskstats 出力ファイルから、データセットに含まれるディスクの変化率を計算します。

roc-calc-diskstats

```
#!/usr/bin/perl
# Copyright (c) 2011, SIOS Technology, Corp.
# Author:Paul Clements
use strict;
sub msg {
```

```

printf STDERR @_;
}
sub dbg {
return if (!$ENV{'ROC_DEBUG'});
msg @_;
}
$0 =~ s@^\.*/@@; # basename
sub usage {
msg "Usage:$0 <interval> <start-time> <iostat-data-file> [dev-list]\n";
msg "\n";
msg "This utility takes a /proc/diskstats output file that contains\n";
msg "output, logged over time, and calculates the rate of change of\n";
msg "the disks in the dataset\n";
msg "OUTPUT_CSV=1 set in env. dumps the full stats to a CSV file on STDERR\n";
msg "\n";
msg "Example:$0 1hour \"jun 23 12pm\" steeleye-iostat.txt sdg,sdh\n";
msg "\n";
msg "interval - interval between samples\n";
msg "start time - the time when the sampling starts\n";
msg "iostat-data-file - collect this with a cron job like:\n";
msg "\t0 * * * * (date ; cat /proc/diskstats) >> /root/diskstats.txt\n";
msg "dev-list - list of disks you want ROC for (leave blank for all)\n";
exit 1;
}
usage if (@ARGV < 3);
my $interval = TimeHuman($ARGV[0]);
my $starttime = epoch($ARGV[1]);
my $file = $ARGV[2];
my $blksize = 512; # /proc/diskstats is in sectors
my %devs = map { $_ => 1 } split /,/, $ARGV[3];
my %stat;
my $firsttime;
my $lasttime;
# datestamp divides output
my %days = ( 'Sun' => 1, 'Mon' => 1, 'Tue' => 1, 'Wed' => 1,
'Thu' => 1, 'Fri' => 1, 'Sat' => 1);
my %fields = ( 'major' => 0,
'minor' => 1,

```

```

'dev' => 2,
'reads' => 3,
'reads_merged' => 4,
'sectors_read' => 5,
'ms_time_reading' => 6,
'writes' => 7,
'writes_merged' => 8,
'sectors_written' => 9,
'ms_time_writing' => 10,
'ios_pending' => 11,
'ms_time_total' => 12,
'weighted_ms_time_total' => 13 );
my $devfield = $fields{'dev'};
my $scalffield = $ENV{'ROC_CALC_FIELD'} || $fields{'sectors_written'};
dbg "using field $scalffield\n";
open(FD, "$file") or die "Cannot open $file:$!\n";
foreach (<FD>) {
chomp;
@_ = split;
if (exists($days{$_[0]})) { # skip datestamp divider
if ($firsttime eq '') {
$firsttime = join ' ', @_[0..5];
}
$lasttime = join ' ', @_[0..5];
next;
}
next if ($_[0] !~ /[0-9]/); # ignore
if (!%devs || exists $devs{$_[$devfield]}) {
push @{$stat{$_[$devfield]}}, $_[$scalffield];
}
}
@{$stat{'total'}} = totals(\%stat);
printf "Sample start time:%s\n", scalar(localtime($starttime));
printf "Sample end time:%s\n", scalar(localtime($starttime + (@{$stat{'total'}} - 1) * $interval));
printf "Sample interval:%ss #Samples:%s Sample length:%ss\n", $interval, (@{$stat{'total'}} - 1), (@{$stat{'total'}} - 1) * $interval;
print "(Raw times from file:$firsttime, $lasttime)\n";
print "Rate of change for devices " .(join ' ', ' ', sort keys %stat) ."\n";

```

```

foreach (sort keys %stat) {
my @vals = @{$stat{$_}};
my ($max, $maxindex, $roc) = roc($_, $blksize, $interval, @vals);
printf "$_ peak:%sB/s (%sb/s) (@ %s) average:%sB/s (%sb/s)\n", HumanSize
($max), HumanSize($max * 8), scalar localtime($starttime + ($maxindex *
$interval)), HumanSize($roc), HumanSize($roc * 8);
}
# functions
sub roc {
my $dev = shift;
my $blksize = shift;
my $interval = shift;
my ($max, $maxindex, $i, $first, $last, $total);
my $prev = -1;
my $first = $_[0];
if ($ENV{'OUTPUT_CSV'}) { print STDERR "$dev," }
foreach (@_) {
if ($prev != -1) {
if ($_ < $prev) {
dbg "wrap detected at $i ($_ < $prev)\n";
$prev = 0;
}
my $this = ($_ - $prev) * $blksize / $interval;
if ($this > $max) {
$max = $this;
$maxindex = $i;
}
if ($ENV{'OUTPUT_CSV'}) { print STDERR "$this," }
}
$prev = $_; # store current val for next time around
$last = $_;
$i++;
}
if ($ENV{'OUTPUT_CSV'}) { print STDERR "\n" }
return ($max, $maxindex, ($last - $first) * $blksize / ($interval * ($i -
1)));
}
sub totals { # params: stat_hash
my $stat = shift;

```

```

my @totalvals;
foreach (keys %$stat) {
next if (!defined($stat{$_}));
my @vals = @{$stat{$_}};
my $i;
foreach (@vals) {
$totalvals[$i++] += $_;
}
}
return @totalvals;
}
# converts to KB, MB, etc. and outputs size in readable form
sub HumanSize { # params: bytes/bits
my $bytes = shift;
my @suffixes = ( '', 'K', 'M', 'G', 'T', 'P' );
my $i = 0;
while ($bytes / 1024.0 >= 1) {
$bytes /= 1024.0;
$i++;
}
return sprintf("%.1f %s", $bytes, $suffixes[$i]);
}
# convert human-readable time interval to number of seconds
sub TimeHuman { # params: human_time
my $time = shift;
my %suffixes = ('s' => 1, 'm' => 60, 'h' => 60 * 60, 'd' => 60 * 60 * 24);
$time =~ /^([0-9]*)(.*)$/;
$time = $1;
my $suffix = (split //, $2)[0]; # first letter from suffix
if (exists $suffixes{$suffix}) {
$time *= $suffixes{$suffix};
}
return $time;
}
sub epoch { # params: date
my $date = shift;
my $seconds = `date +%s' --date "$date" 2>&1`;
if ($?!= 0) {

```

```
die "Failed to recognize time stamp:$date\n";
}
return $seconds;
}
```

使用法:

```
# ./roc-calc-diskstats <interval> <start_time> <diskstats-data-file>
[dev-list]
```

使用例(概要のみ):

```
# ./roc-calc-diskstats 2m "Jul 22 16:04:01" /root/diskstats.txt
sdb1,sdb2,sdc1 > results.txt
```

この例は、概要(およびディスク別のピークI/O情報)を *results.txt* にダンプします。

使用例(概要とグラフデータ):

```
# export OUTPUT_CSV=1
# ./roc-calc-diskstats 2m "Jul 22 16:04:01" /root/diskstats.txt
sdb1,sdb2,sdc1 2> results.csv > results.txt
```

この例は、グラフデータを *results.csv* に、概要(およびディスク別のピークI/O情報)を *results.txt* にダンプします。

結果の例(results.txt)

```
Sample start time:Tue Jul 12 23:44:01 2011
Sample end time:Wed Jul 13 23:58:01 2011
Sample interval:120s #Samples:727 Sample length:87240s
(Raw times from file:Tue Jul 12 23:44:01 EST 2011, Wed Jul 13 23:58:01
EST 2011)
Rate of change for devices dm-31, dm-32, dm-33, dm-4, dm-5, total
dm-31 peak:0.0 B/s (0.0 b/s) (@ Tue Jul 12 23:44:01 2011) average:0.0 B/s
(0.0 b/s)
dm-32 peak:398.7 KB/s (3.1 Mb/s) (@ Wed Jul 13 19:28:01 2011)
average:19.5 KB/s (156.2 Kb/s)
dm-33 peak:814.9 KB/s (6.4 Mb/s) (@ Wed Jul 13 23:58:01 2011)
average:11.6 KB/s (92.9 Kb/s)
dm-4 peak:185.6 KB/s (1.4 Mb/s) (@ Wed Jul 13 15:18:01 2011) average:25.7
KB/s (205.3 Kb/s)
dm-5 peak:2.7 MB/s (21.8 Mb/s) (@ Wed Jul 13 10:18:01 2011) average:293.0
KB/s (2.3 Mb/s)
```

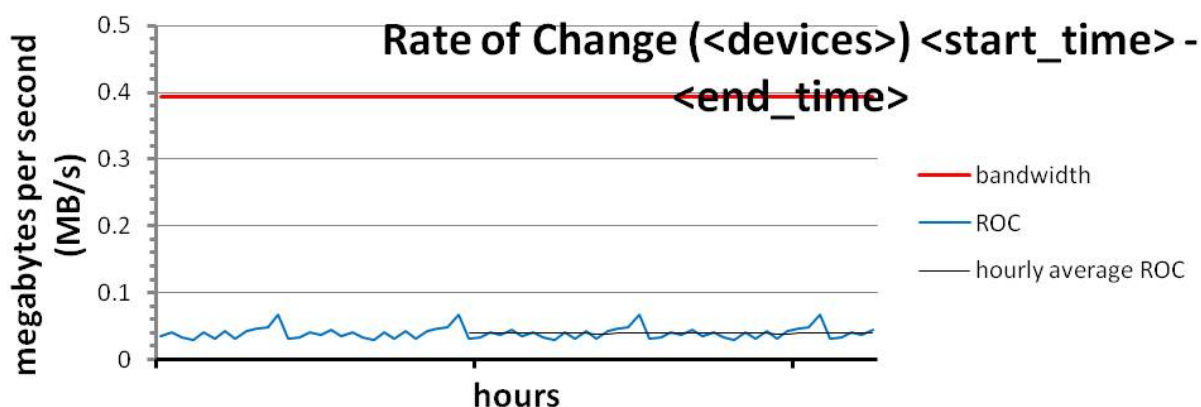
total peak:2.8 MB/s (22.5 Mb/s) (@ Wed Jul 13 10:18:01 2011)
 average:349.8 KB/s (2.7 Mb/s)

詳細変化率データのグラフ作成

お客様に固有の経時的な帯域幅のニーズを分かりやすくするために、テンプレートスプレッドシート diskstats-template.xlsx が用意されています。このスプレッドシートにはサンプルデータがあり、roc-calc-diskstats で収集したデータで上書きできます。

をダウンロードするには

diskstats-template



1. results.csv を開き、total 列を含めてすべての行を選択してください。

	A	B	C	D	E	F	G	H	I	J	K	L
1	dm-31	0	0	0	0	0	0	0	0	0	0	0
2	dm-32	3549.867	6826.667	3549.867	273.0667	7099.733	3549.867	6826.667	3686.4	341.3333	7099.733	3549.8
3	dm-33	3857.067	4505.6	3310.933	1911.467	4846.933	2935.467	4471.467	3310.933	1911.467	4710.4	2935.4
4	dm-4	2218.667	2389.333	2150.4	1570.133	2525.867	2116.267	2389.333	2013.867	4300.8	2833.067	1809.0
5	dm-5	25326.93	26683.73	23449.6	25164.8	26419.2	23048.53	28612.27	21367.47	35140.27	32145.07	40797.
6	total	34952.53	40405.33	32460.8	28919.47	40891.73	31650.13	42299.73	30378.67	41693.87	46788.27	49092.

2. diskstats-template.xlsx を開き、diskstats.csv ワークシートを選択してください。



3. セル 1-A を右クリックし、[Insert Copied Cells] を選択してください。
4. レプリケーション用に割り当てた帯域幅の量を反映するように、ワークシートの左下にあるセルの

詳細変化率データのグラフ作成

bandwidth 値を調整してください。

単位: メガビット/秒 (Mb/sec)

注記: その右側にあるセルの値は、収集した生データに合わせて自動的にバイト / 秒単位に変換されます。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	dm-31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	dm-32	3549.86667	6826.66667	3549.867	273.0667	7099.733	3549.867	6826.667	3686.4	341.3333	7099.733	3549.867	6826.667	3276.8	273.0667	6826.667	3549.867	6826.667
3	dm-33	3857.06667	4505.6	3310.933	1911.467	4846.933	2935.467	4471.467	3310.933	1911.467	4710.4	2935.467	4710.4	2935.467	2798.933	4676.267	3857.067	4710.4
4	dm-4	2218.66667	2389.33333	2150.4	1570.133	2525.867	2116.267	2389.333	2013.867	4300.8	2833.067	1809.067	27955.2	1570.133	2286.933	2525.867	2116.267	2628.267
5	dm-5	25326.9333	26683.73333	23449.6	25164.8	26419.2	23048.53	28612.27	21367.47	35140.27	32145.07	40797.87	28492.8	23338.67	28561.07	27067.73	27784.53	29849.6
6	total	34952.5333	40405.33333	32460.8	28919.47	40891.73	31650.13	42299.73	30378.67	41693.87	46788.27	49092.27	67985.07	31121.07	33920	41096.53	37307.73	44014.93
7																		
8	bandwidth (Mb/s)																	
9		10	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720

5. 以下の行/列番号を記録してください。

- Total(下のスクリーンショットでは行 6)
- Bandwidth(下のスクリーンショットでは行 9)
- 最終データポイント(下のスクリーンショットでは列 R)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	dm-31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	dm-32	3549.86667	6826.66667	3549.867	273.0667	7099.733	3549.867	6826.667	3686.4	341.3333	7099.733	3549.867	6826.667	3276.8	273.0667	6826.667	3549.867	6826.667
3	dm-33	3857.06667	4505.6	3310.933	1911.467	4846.933	2935.467	4471.467	3310.933	1911.467	4710.4	2935.467	4710.4	2935.467	2798.933	4676.267	3857.067	4710.4
4	dm-4	2218.66667	2389.33333	2150.4	1570.133	2525.867	2116.267	2389.333	2013.867	4300.8	2833.067	1809.067	27955.2	1570.133	2286.933	2525.867	2116.267	2628.267
5	dm-5	25326.9333	26683.73333	23449.6	25164.8	26419.2	23048.53	28612.27	21367.47	35140.27	32145.07	40797.87	28492.8	23338.67	28561.07	27067.73	27784.53	29849.6
6	total	34952.5333	40405.33333	32460.8	28919.47	40891.73	31650.13	42299.73	30378.67	41693.87	46788.27	49092.27	67985.07	31121.07	33920	41096.53	37307.73	44014.93
7																		
8	bandwidth (Mb/s)																	
9		10	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720	1310720

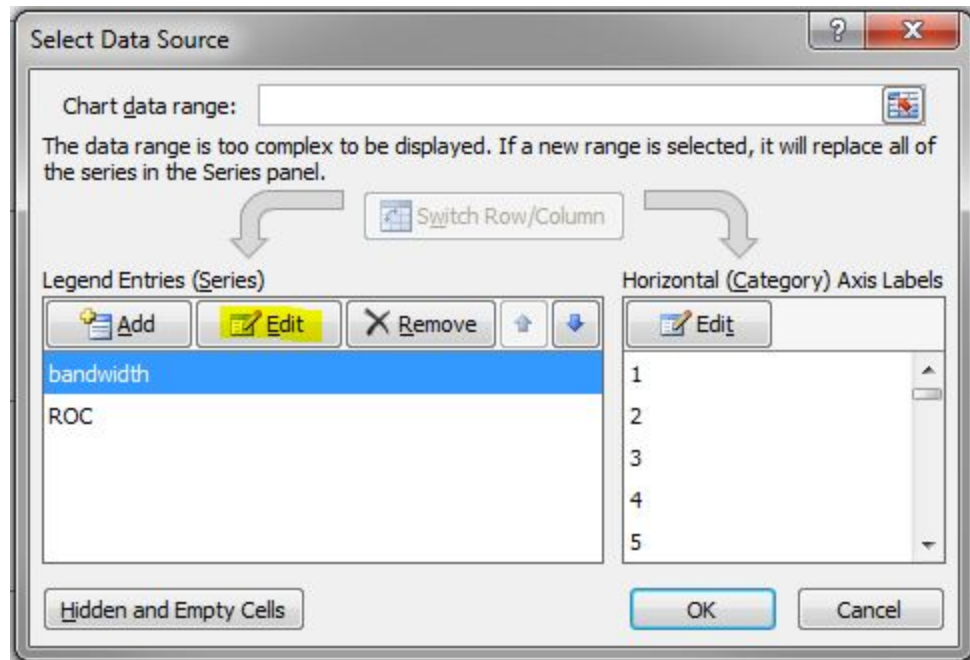
6. bandwidth vs ROC ワークシートを選択してください。



7. グラフを右クリックし、[Select Data...]を選択してください。

- Bandwidth 系列を調整してください。
 - 左の [Series] リストから bandwidth を選択してください。
 - [Edit] をクリックしてください。

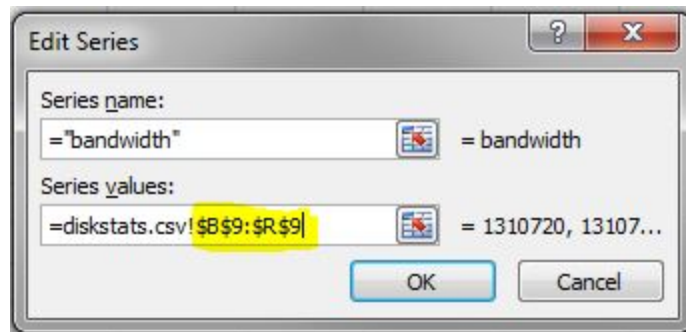
詳細変化率データのグラフ作成



iii. 以下の構文を使用して、**[Series Values]** フィールドを調整してください。

"=diskstats.csv!\$B\$<row>:\$<final_column>\$<row>"

例: "=diskstats.csv!\$B\$9:\$R:\$9"

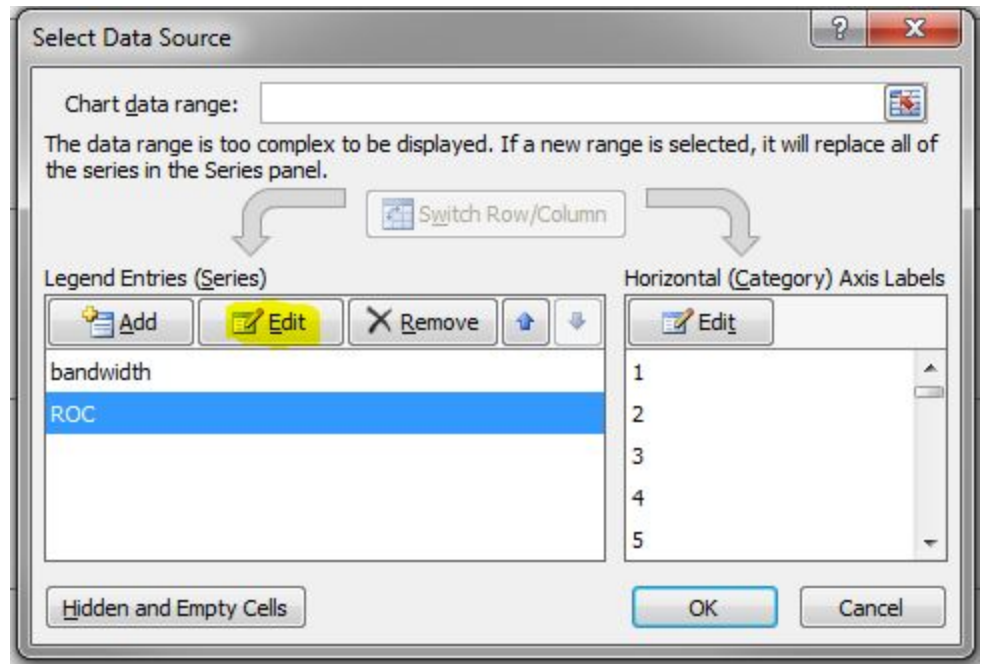


iv. **[OK]** をクリックしてください。

b. **ROC** 系列を調整してください。

i. 左の **[Series]** リストから **ROC** を選択してください。

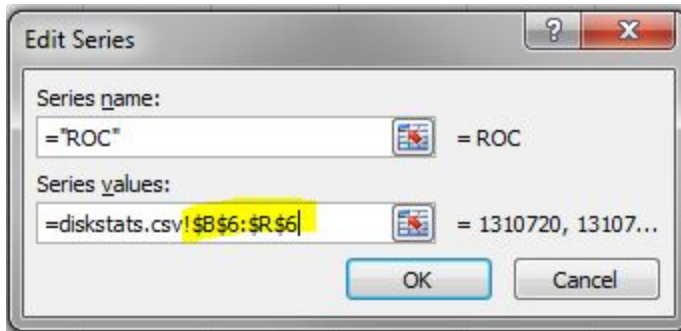
ii. **[Edit]** をクリックしてください。



iii. 以下の構文を使用して、**[Series Values]** フィールドを調整してください。

```
"=diskstats.csv!$B$<row>:$<final_column>$<row>"
```

例: "=diskstats.csv!\$B\$6:\$R:\$6"



iv. **[OK]** をクリックしてください。

c. **[OK]** をクリックしてウィザードを終了してください。

8. Bandwidth vs ROC のグラフが更新されます。結果を解析して、データのレプリケーションをサポートするために十分な帯域幅があるかどうかを判断してください。

SIOS DataKeeper for Linux のリソースタイプ

DataKeeper リソース階層を作成するときに、リソースタイプを選択するように LifeKeeper から要求されます。

DataKeeper リソースには、いくつかのタイプがあります。お使いの環境に最適なタイプを選択するときに、以下の情報が役立ちます。

Replicate New File System

[Replicate New File System](#) を選択すると、NetRAID デバイスが作成/拡張され、NetRAID デバイ스에指定のマウントポイントがマウントされます。また、LifeKeeper がサポートするファイルシステムと NetRAID デバイスの両方が、LifeKeeper で保護されます。ローカルのディスクまたはパーティションがフォーマットされます。**注意:** データがすべて削除されます。

Replicate Existing File System

[Replicate Existing File System](#) を選択すると、現在マウントされているディスクまたはパーティションが使用され、ディスクまたはパーティションのデータが削除されることなく NetRAID デバイスが作成されます。SIOS DataKeeper はローカルのディスクまたはパーティションをアンマウントし、ローカルのディスクまたはパーティションを使用して NetRAID デバイスを作成します。そして、NetRAID デバイ스에マウントポイントをマウントします。次に、NetRAID デバイスと LifeKeeper がサポートするファイルシステムの両方を LifeKeeper で保護します。

重要: SIOS Protection Suite for Linux のマルチサイトクラスタ階層を作成する場合、作成プロセス中にアプリケーションが停止します。階層の作成と拡張が完了した後、アプリケーションを再起動する必要があります。

DataKeeper Resource

[DataKeeper リソース](#) を選択すると、NetRAID デバイスが作成/拡張され、ファイルシステムは含めずに LifeKeeper で保護されます。RAW I/O デバイスを使用できるデータベースを使用している場合は、この複製タイプを選択できます。

ユーザがデータアクセスを続行できるように、SIOS DataKeeper は、現在マウントされている NetRAID デバイスのアンマウントと削除は実行しません。ユーザは、手動スイッチオーバーの前に NetRAID デバイスを手動でアンマウントし、手動スイッチオーバーの後に他のサーバにマウントする必要があります。

注記: DataKeeper リソースの作成後に、手動マウントしたファイルシステムを LifeKeeper で保護する場合は、以下の操作を行います。

1. LifeKeeper がサポートするファイルシステムで、NetRAID デバイスをフォーマットしてください。
2. NetRAID デバイスをマウントしてください。
3. NetRAID デバイスを使用して、共有ストレージのディスクまたはパーティションにあるかのようにファイルシステムの階層を作成し、拡張してください。

これで、LifeKeeper のファイルシステムリカバリキットが、フェイルオーバー時のファイルシステムのマウント / アンマウントを実行します。

リソースの設定作業

SIOS DataKeeper の設定作業はすべて、LifeKeeper のグラフィカルユーザインターフェース (GUI) から実行できます。LifeKeeper の GUI では、SIOS DataKeeper のリソースの設定、管理、監視の作業をガイド付きで行うこと

ができます。

概要

SIOS DataKeeper の設定に関して、以下の作業を行うことができます。

- **Create a Resource Hierarchy** - DataKeeper リソース階層を作成します。
- **Delete a Resource Hierarchy** - DataKeeper リソース階層を削除します。
- **Extend a Resource Hierarchy** - DataKeeper リソース階層をプライマリサーバからバックアップサーバに拡張します。
- **Unextend a Resource Hierarchy** - LifeKeeper クラスタ内にある 1 台のサーバの DataKeeper リソース階層を拡張解除 (削除) します。
- **Create Dependency** - 既存のリソース階層と別のリソースインスタンスとの間に子の依存関係を作成し、クラスタ内のすべての対象サーバに依存関係の変更を伝播します。
- **Delete Dependency** - リソースの依存関係を削除して、クラスタ内にあるすべての対象サーバに依存関係の変更を伝播します。
- **In Service** - リソース階層をアクティブにします。
- **Out of Service** - リソース階層を非アクティブにします。
- **View/Edit Properties** - リソース階層のプロパティの表示または編集を行います。

DataKeeper リソース階層の作成

[マルチサイトクラスタ](#)環境に DataKeeper リソース階層を作成する場合は、**[Hierarchy Type]** を選択した後、このセクションの最後にある手順を参照してください。

プライマリサーバで以下の操作を行ってください。

1. **[Edit] > [Server] > [Create Resource Hierarchy]** を選択してください。
[Create Resource Wizard] ダイアログボックスが表示されます。
2. ドロップダウンリストから **[Data Replication]** オプションを選択し、**[Next]** をクリックして続行してください。
3. 以下の情報を入力するように要求されます。ダイアログボックスで **[Back]** ボタンが有効な場合は、前のダイアログボックスに戻ることができます。これは、エラーが発生して、前に入力した情報を修正する必要がある場合に便利な機能です。いつでも **[Cancel]** をクリックして、作成処理全体を取り消すことができます。

フィールド	ヒント
Switchback Type	<p>[intelligent switchback] を指定する必要があります。これは、バックアップサーバにフェイルオーバーした後、管理者が手動で DataKeeper リソースをプライマリサーバにスイッチバックする必要があることを意味します。</p> <p>注意: このリリースの SIOS DataKeeper は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、DataKeeper リソースの上に存在する他の LifeKeeper リソースにも適用されます。</p>
Server	作成する NetRAID デバイスが存在するサーバ(通常はプライマリサーバ)の名前を選択してください。ドロップダウンリストボックスには、クラスタ内のすべてのサーバが表示されます。
Hierarchy Type	<p>以下のいずれかを選択して、作成するデータレプリケーションのタイプを選択してください。</p> <ul style="list-style-type: none"> • Replicate New File System • Replicate Existing File System • DataKeeper Resource
Bitmap File	<p>インテントログの記録に使用するビットマップファイルの名前を選択するか、入力してください。[None] を選択すると、インテントログは使用されず、すべての再同期が部分的ではなく全体の再同期になります。</p> <p>重要: ビットマップファイルは btrfs ファイルシステム上に置いてはいけません。データレプリケーションのビットマップファイルが btrfs ファイルシステム上に置かれると、LifeKeeper がミラーを構成しようとした時、"invalid argument" エラーの原因になります。ビットマップファイルのデフォルトの置き場所は、<code>/opt/LifeKeeper</code> の下です。このデフォルトの置き場所は、<code>/opt/LifeKeeper</code> が btrfs ファイルシステム上にある場合変更されます。</p>
Enable Asynchronous Replication ?	このレプリケーションリソースによるターゲットシステムへの非同期レプリケーションのサポートを許可するには、 [Yes] を選択してください。すべてのターゲットについて同期レプリケーションを使用する場合は、 [No] を選択してください。後で、レプリケーションリソースが各ターゲットサーバに拡張されるときに、実際のレプリケーションタイプ(同期または非同期)を選択するように要求されます。(両方のレプリケーションタイプの詳細については、 SIOS DataKeeper によるミラーリング を参照してください)。これらのターゲットのいずれかへのレプリケーションを非同期で実行する場合は、他のターゲットへのレプリケーションを同期実行する場合でもここでは [Yes] を選択する必要があります。

以降の一連のダイアログボックスは、**[Hierarchy Type]** で選択した項目によって異なります。一部のダイアログボックスはすべての階層タイプで同じですが、表示される順序と必要な情報が少し異なることがあります。以下の3つのトピックで、階層作成の残りのプロセスについて説明しています。

- [DataKeeper Resource](#)
- [Replicate New File System](#)
- [Replicate Existing File System](#)

リソース階層の拡張

この操作は **[Edit]** メニューから開始できます。または **[Create Resource Hierarchy]** オプションの動作が完了すると自動的に開始されます。その場合は、手順 2 を参照してください。

1. **[Edit]** メニューの **[Resource]** から **[Extend Resource Hierarchy]** を選択します。**Pre-Extend Wizard** が表示されます。拡張操作に慣れていない場合は、**[Next]** をクリックしてください。LifeKeeper の **[Extend Resource Hierarchy]** のデフォルト値が分かっている、入力と確認を省略する場合は **[Accept Defaults]** をクリックしてください。
2. **Pre-Extend Wizard** に以下の情報を入力します。

注記: 最初の 2 つのフィールドは **[Edit]** メニューから拡張を開始した場合にのみ表示されます。

フィールド	ヒント
Template Server	<p>現在 In Service の DataKeeper リソース階層が存在するテンプレートサーバを選択してください。ここで選択するテンプレートサーバと次のダイアログボックスで選択する拡張するタグによって、In Service (アクティブ) のリソース階層が表示されることを理解しておくことが重要です。</p> <p>選択したテンプレートサーバで In Service でないリソースタグを選択した場合、エラーメッセージが表示されます。このダイアログのドロップダウンボックスには、クラスタ内にある全サーバの名前が表示されます。</p>
Tag to Extend	<p>これは、テンプレートサーバからターゲットサーバに拡張する DataKeeper インスタンスの名前です。ドロップダウンボックスには、テンプレートサーバ上に作成したすべてのリソースが表示されます。</p>
Target Server	<p>拡張先のサーバを入力するか、選択してください。</p>
Switchback Type	<p>[intelligent switchback] を指定する必要があります。これは、バックアップサーバにフェイルオーバーした後、管理者が手動で DataKeeper リソースをプライマリサーバにスイッチバックする必要があることを意味します。</p> <p>注意: このリリースの SIOS DataKeeper は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、SIOS DataKeeper リソースの上に存在する他の LifeKeeper リソースにも適用されます。</p>
Template Priority	<p>テンプレートの優先順位を選択するか、入力してください。これはサーバで現在 In Service の DataKeeper 階層の優先順位です。1 ~ 999 の範囲で、まだ優先順位として使用されていない値が有効で、小さい数字ほど優先順位が高くなります(数値 1 が最高の優先順位)。拡張処理時に、別のシステムですでに使用中の優先順位をこの階層に対して指定することはできません。デフォルト値が推奨されます。</p> <p>注記: このフィールドは、階層をはじめて拡張するときにだけ表示されます。</p>

フィールド	ヒント
Target Priority	ターゲットの優先順位を選択するか、入力してください。これは、他のサーバにある同等の階層に対する、新しく拡張する DataKeeper 階層の優先順位です。1 ~ 999 の範囲で、まだ優先順位として使用されていない値が有効で、リソースのカスケードフェイルオーバーシナリオにおけるサーバの優先順位を示します。数値が小さいほど優先順位は高くなります(数値 1 が最高の優先順位)。LifeKeeper のデフォルトでは、階層が作成されたサーバに「1」が割り当てられることに注意してください。優先順位は連続している必要はありませんが、特定のリソースについて 2 つのサーバに同じ優先順位を割り当てることはできません。

拡張前のチェックが正常に終了したというメッセージが表示されたら、[Next] をクリックしてください。

拡張する階層に応じて、拡張されるリソースタグ(一部編集不可)を示す一連の情報ボックスが表示されます。

3. [Next] をクリックして、[Extend Resource Hierarchy] の構成タスクを開始してください。
4. 次のセクションには、別のサーバに DataKeeper リソースを拡張するために必要な手順を示します。

DataKeeper リソース階層の拡張

1. pre-extend スクリプトが正常に実行されたというメッセージが表示されたら、以下の情報を指定するように要求されます。

フィールド	ヒント
Mount Point	ターゲットサーバ上にあるファイルシステムマウントポイントを入力してください(DataKeeper リソースに関連する、LifeKeeper が保護するファイルシステムがない場合は、このダイアログは表示されません)。
Root Tag	ルートタグを選択するか、入力してください。これは、ターゲットサーバ上にあるファイルシステムリソースインスタンスの一意の名前です(DataKeeper リソースに関連する、LifeKeeper が保護するファイルシステムがない場合は、このダイアログは表示されません)。

フィールド	ヒント
Target Disk or Partition	<p>複製するファイルシステムの配置先となる、ターゲットサーバ上のディスクまたはパーティションを選択してください。</p> <p>ドロップダウンボックスのディスクまたはパーティションのリストには、<u>以下のものを除いて</u>、使用できるすべてのディスクが表示されます。</p> <ul style="list-style-type: none"> • すでにマウント済みのもの • スワップディスクまたはスワップパーティション • LifeKeeper が保護するディスクまたはパーティション <p>ドロップダウンリストには、root (/)、boot (/boot)、/proc、floppy、cdrom などの特殊なディスクまたはパーティションも表示されません。</p> <p>注記: ターゲットのディスクまたはパーティションは、ソースのディスクまたはパーティション以上のサイズである必要があります。</p>
DataKeeper Resource Tag	DataKeeper リソースタグの名前を選択するか、入力してください。
Bitmap File	<p>インテントログの記録に使用するビットマップファイルの名前を選択するか入力してください。[None] を選択すると、インテントログは使用されず、すべての再同期が部分的ではなく全体の再同期になります。</p>
Replication Path	<p>ターゲットサーバとクラスタ内の他の指定サーバとの間で複製に使用する、ローカルとリモートの IP アドレスのペアを選択してください。有効なパスおよび対応する IP アドレスは、このサーバのペアに対して指定した LifeKeeper コミュニケーションパスのセットから得られます。DataKeeper の特性により、プライベート (専用) ネットワークを使用することが強く推奨されます。</p> <p>DataKeeper リソースをすでに 1 台以上のターゲットサーバに拡張している場合、追加のサーバに対する拡張を実行すると、新しいターゲットサーバと既存のサーバとの組み合わせのそれぞれについて、繰り返し複製パスを指定するように要求されます。</p>
Replication Type	<p>指定したサーバのペアについて使用する複製タイプとして、[synchronous] または [asynchronous] を選択してください。</p> <p>前述の [Replication Path] フィールドと同様に、DataKeeper リソースをすでに 1 台以上のターゲットサーバに拡張している場合、追加のサーバに対する拡張を実行すると、新しいターゲットサーバと既存のサーバとの組み合わせのそれぞれについて、繰り返し複製タイプを指定するように要求されます。</p>

2. **[Extend]** をクリックして次に進んでください。拡張が実行中であることを確認する情報ボックスが表示されます。
3. **[Finish]** をクリックして、DataKeeper リソースインスタンスが正常に拡張されたことを確認してください。
4. **[Done]** をクリックして、**[Extend Resources Hierarchy]** メニューを終了してください。

注記: 必ずすべてのサーバでスイッチオーバーを手動実行して、新しいインスタンスの機能をテストしてください。詳細については、[リソース階層のテスト](#)を参照してください。この時点で、SIOS DataKeeper がソースからターゲットのディスクまたはパーティションにデータの再同期を開始しています。LifeKeeper の GUI では、ターゲットサーバにある DataKeeper リソースのステータスは「Resyncing」になります。再同期が完了すると、ステータスは「Target」になります。これは通常のスタンバイ状態です。

再同期中、DataKeeper リソース、およびそれに依存するリソースはフェイルオーバーできません。これは、データの破損を防止するためです。

リソース階層の拡張解除

LifeKeeper クラスタ内にある 1 台のサーバからリソース階層を削除するには、次の手順を実行します。

1. **[Edit]** メニューの **[Resource]** から **[Unextend Resource Hierarchy]** を選択してください。
2. DataKeeper リソースを拡張解除する**ターゲットサーバ**を選択してください。DataKeeper リソースが現在 In Service (アクティブ) のサーバは選択できません。

注記: 右側のペインから個々のリソースインスタンスを右クリックして **[Unextend]** 作業を選択した場合、このダイアログボックスは表示されません。

[Next] をクリックしてください。

3. **拡張解除する DataKeeper 階層**を選択し、**[Next]** をクリックしてください(このダイアログは、いずれかのペインでリソースインスタンスを右クリックして、**[Unextend]** を選択した場合には表示されません)。
4. 選択したターゲットサーバと DataKeeper リソース階層の拡張解除を確認する情報ボックスが表示されます。**[Unextend]** をクリックしてください。
5. DataKeeper リソースが正常に拡張解除されたことを確認する別の情報ボックスが表示されます。**[Done]** をクリックして、**[Unextend Resource Hierarchy]** メニューを終了してください。

注記: これで、データがバックアップサーバにレプリケーションされなくなります。

リソース階層の削除

LifeKeeper 構成内のすべてのサーバから DataKeeper リソースを削除するには、次の手順を実行してください。

注記: DataKeeper リソースは、削除する前に Out of Service にすることが推奨されます。Out of Service にしない場合、**md** と **NetRAID** のデバイスが削除されず、ファイルシステムを手動でアンマウントする必要があります。[DataKeeper リソースを Out of Service にする](#)を参照してください。

1. **[Edit]** メニューの **[Resource]** から **[Delete Resource Hierarchy]** を選択してください。
2. 削除する DataKeeper リソース階層が存在する**ターゲットサーバ**の名前を選択してください。

注記: 左側ペインのグローバルリソースまたは右側ペインの個々のリソースインスタンスを右クリックして **[Delete Resource]** 作業を選択した場合、このダイアログボックスは表示されません。
3. **[Hierarchy to Delete]** を選択してください(左右のペインのリソースインスタンスを右クリックして **[Delete Resource]** 作業を選択した場合、このダイアログは表示されません) **[Next]** をクリックしてください。
4. 選択したターゲットサーバと、削除の対象として選択した階層を確認する情報ボックスが表示されます。

[Delete] をクリックしてください。

5. DataKeeper リソースが正常に削除されたことを確認する別の情報ボックスが表示されます。[Done] をクリックして終了してください。

注記: リソースを削除する前にマウントされた状態の NetRAID デバイスは、マウントされたまま残ります。それ以外の NetRAID デバイスは削除されます。

DataKeeper リソースを Out of Service にする

DataKeeper リソースを Out of Service にすると、LifeKeeper によるリソースの保護が解除されます。ミラーが解除され、ファイルシステムがアンマウントされます(該当する場合)。md デバイスが停止し、nbd サーバとクライアントが強制終了されます。

警告: データのミラーリングを停止して LifeKeeper の保護を解除する場合以外は、DataKeeper リソースを Out of Service にしないでください。一時停止の操作を使用して、ミラーリングを一時停止してください。

1. LifeKeeper の GUI の右側ペインにある、In Service の **DataKeeper リソース** を右クリックしてください。
2. リソースのポップアップメニューの **[Out of Service]** をクリックしてください。
3. 選択したリソースが Out of Service になることを示すダイアログボックスが表示されます。この操作に関連するリソースの依存関係がダイアログに表示されます。**[Next]** をクリックしてください。
4. 情報ボックスに、Out of Service にするリソースの結果が表示されます。**[Done]** をクリックしてください。

DataKeeper リソースを In Service にする

DataKeeper リソースを In Service にする操作は、リソースの作成と似ています。LifeKeeper は nbd サーバとクライアントを起動し、ソースとターゲットのデバイス間でデータを同期する md デバイスを起動して、ファイルシステムをマウントします(該当する場合)。

1. 右側のペインにある **DataKeeper リソースインスタンス** を右クリックしてください。
2. ポップアップメニューの **[In Service]** をクリックしてください。選択したサーバとリソースを In Service にすることを確認するダイアログボックスが表示されます。**[In Service]** をクリックしてリソースを In Service にしてください。
3. 情報ボックスに、In Service にするリソースの結果が表示されます。この操作に関連するリソースの依存関係が確認ダイアログに表示されます。**[Done]** をクリックしてください。

リソース階層のテスト

手動スイッチオーバーを開始することによって、DataKeeper リソース階層をテストできます。このテストは、プライマリサーバからバックアップサーバへのリソースインスタンスのフェイルオーバーをシミュレートします。

LifeKeeper の GUI からの手動スイッチオーバーの実行

手動スイッチオーバーを開始するには、LifeKeeper の GUI で **[Edit] > [Resource] > [In Service]** を選択します。例えば、バックアップサーバで In Service リクエストが実行されると、DataKeeper リソース階層がバックアップサーバ

側で In Service になり、プライマリサーバ側では Out of Service になります。この時点で、元のバックアップサーバがプライマリサーバに、元のプライマリサーバがバックアップサーバになります。

スイッチオーバー後、LifeKeeper の GUI では、ターゲットサーバにある DataKeeper リソースのステータスが「**Resyncing**」(再同期中)になります。再同期が完了すると、ステータスは「**Target**」になります。これは通常のスタンバイ状態です。

注記: 再同期中は、DataKeeper リソースの手動フェイルオーバーはできません。

[Out of Service] 要求を実行した場合、リソース階層は他のサーバで In Service にならずに、Out of Service になります。リソースを同じサーバ上で In Service に戻すことができるのは、再同期中にリソースが Out of Service になった場合のみです。

SIOS DataKeeper for Linux の管理

以下のトピックには、リソースを作成した後の SIOS DataKeeper for Linux の動作と問題を理解し、管理するのに役立つ情報があります。

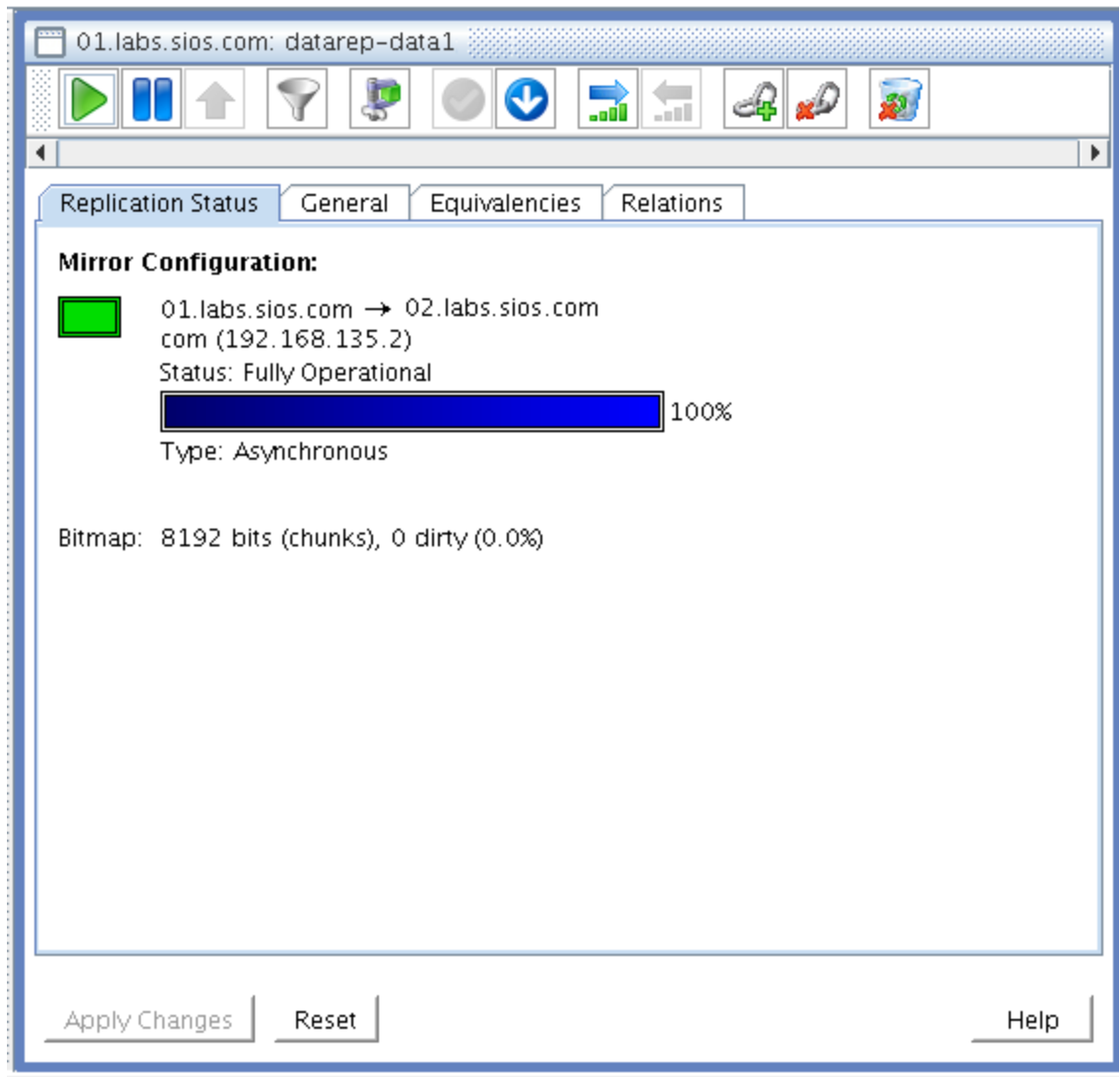
ミラーのステータスの表示

[Replication Status] ダイアログには、ミラーに関する以下の情報が表示されます。

- **Mirror status:** Fully Operational(フルに動作可能)、Paused(一時停止)、Resyncing(再同期中)、または Out Of Sync(同期停止)
- **Synchronization status:** 同期が完了した割合
- **Replication type:** synchronous(同期)または asynchronous(非同期)
- **Replication direction:** ソースサーバからターゲットサーバに
- **Bitmap:** ビットマップ/インテントログの状態
- **Network Compression Level:** 圧縮レベル(有効の場合)

[Replication Status] ダイアログを表示するには、次の手順に従います。

1. **[View]** メニューをクリックし、**[Properties Panel]** を選択します。
2. **[LifeKeeper status]** 表示にある **DataKeeper** リソースをクリックします。
または
 1. **[LifeKeeper status]** 表示にある **DataKeeper** リソースを右クリックします。
 2. ポップアップメニューから **[Properties]** を選択します。

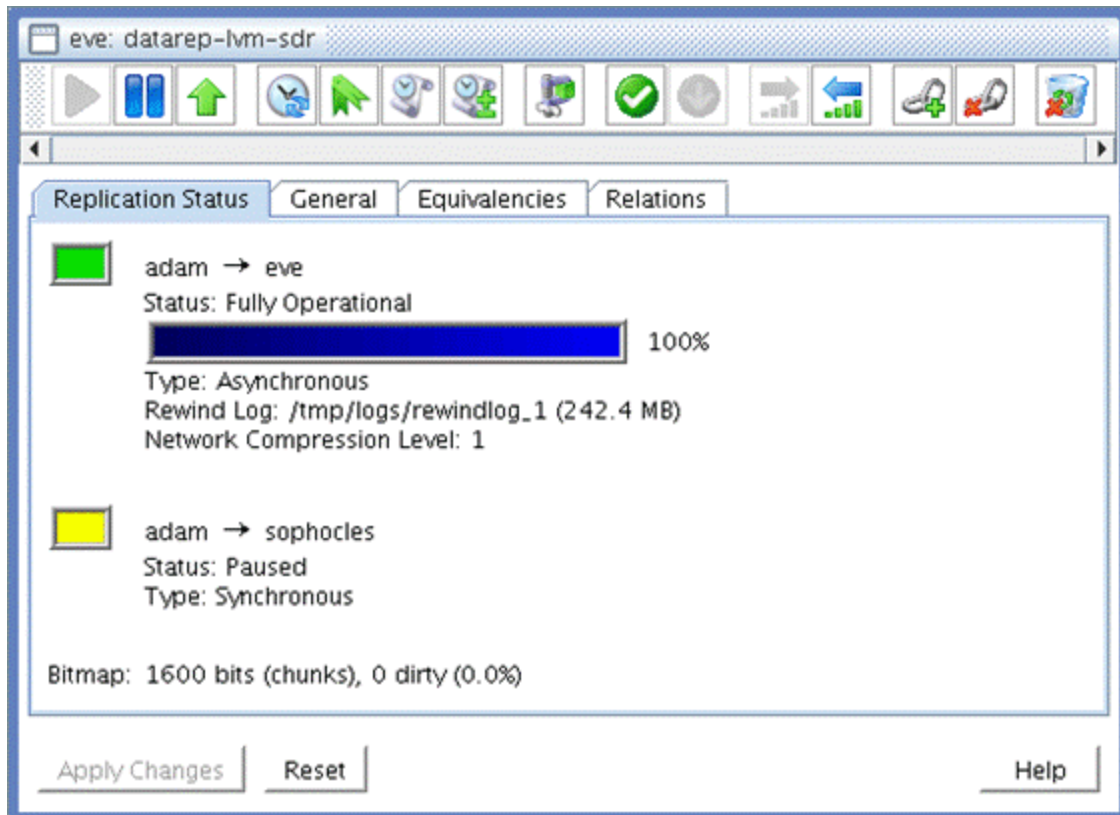


GUI からのミラーの管理

SIOS DataKeeper のミラーは、LifeKeeper の GUI から以下の 2 とおりの方法で実行できます。

1. **[Properties Panel]** を有効にし、ツールバーのアイコン(スクリーンショットを参照)をクリックします。

ミラーを強制的にオンラインにする



説明を表示するには、それぞれのアイコンをクリックしてください。



または

2. **data replication** リソースを右クリックし、ポップアップメニューから動作を選択します。

ミラーを強制的にオンラインにする



[Force Mirror Online] は、両方のサーバが動作不能になり、かつプライマリーサーバの再起動後にリソースを In Service にできない場合にのみ使用してください。**[Force Mirror Online]** を選択すると、`data_corrupt` フラグが削除され、DataKeeper リソースが In Service になります。詳細については、[トラブルシューティング](#)セクションの「プライマリーサーバがリソースを ISP にできない」を参照してください。

注記: `mirror_settings` はターゲットシステム(ミラーのソースになるシステムには無関係に設定を有効にする場合はすべてのシステム)で実行する必要があります。設定の変更内容を有効にするには、ミラーを一時停止してから再起動する必要があります。

一時停止と再開

ミラーの一時停止



ミラーの再開



ミラーを一時停止して、すべての書き込みをターゲットディスクに複製する動作を一時的に停止できます。例えば、ミラーを一時停止して、ターゲットディスクのスナップショットを収集することも、トラフィックのピーク時にソースシステムの I/O パフォーマンスを向上させることもできます。

ミラーを一時停止すると、ミラーは、ターゲットシステムの通常のファイルシステムのマウントポイントに読み取りアクセスするように(カーネル 2.6.19 以降は読み取り/書き込みアクセス)マウントされます。ミラーの一時停止中にターゲットに書き込まれたデータはすべて、ミラーの再開時に上書きされます。

圧縮レベルの設定



ネットワークの圧縮レベルは、0 ~ 9 の値に設定できます。値 0 は、圧縮を無効にします。レベル 1 は最も高速で圧縮率が最も低いレベルです。一方、レベル 9 は最も低速ですが圧縮率が最高です。ネットワーク圧縮は通常、WAN 環境で利用することによって効果を発揮します。

コマンドラインからのミラー管理

ミラーの管理は、LifeKeeper の GUI からの操作だけでなく、コマンドラインからも実行できます。DataKeeper のリソースの管理に使用できるコマンドがいくつかあります(`$LKROOT/bin` ディレクトリを参照)。

ミラーの操作

```
mirror_action <tag> <action> [source] [target(s)]
```

<tag> DataKeeper リソースを表す LifeKeeper リソースタグ

例:

<action> 以下のいずれか: pause、resume、force、fullresync

[source] (オプション) 現在のソースシステム (ソースが指定されない場合、コマンドが実行されたカレントシステムを使用)

[target] (オプション) 操作対象のターゲットシステム (またはシステムのリスト。ターゲットが指定されない場合、該当するすべてのターゲットを使用)

注記: force 操作を使用する場合、ソースノードを指定するために source 引数が必要ですが、target (s) 引数は必要ありません。pause、resume、fullresync のいずれかの操作を使用するときに target (s) 引数を指定する場合は、source 引数も必要です。

例:

datarep-ext3 という名前のミラーを一時停止する。

```
mirror_action datarep-ext3 pause
```

adam から eve と sophocles の両方のシステムへの複製を再開する。

```
mirror_action datarep-ext3 resume adam eve sophocles
```

システム eve へのオンラインミラーリングを強制実行する。

```
mirror_action datarep-ext3 force eve
```

adam から sophocles への複製を再開し、これらのシステム間で全体の再同期を強制実行する。

```
mirror_action datarep-ext3 fullresync adam sophocles
```

ミラーの設定

```
mirror_settings <tag> <setting> <value>
```

<tag> DataKeeper リソースを表す LifeKeeper リソースタグ

<setting> 以下のいずれか: logdir、logmax、compress

<value> 設定する値

注記: mirror_settings はターゲットシステム (ミラーのソースになるシステムには無関係に設定を有効にする場合はすべてのシステム) で実行する必要があります。設定の変更内容を有効にするには、ミラーを一時停止してから再起動する必要があります。

例:

ネットワーク圧縮のレベルを 5 に設定する。

```
mirror_settings datarep-ext3 compress 5
```

ネットワーク圧縮を無効にする。

```
mirror_settings datarep-ext3 compress 0
```

ミラーのサイズ変更

`mirror_resize` コマンドはオフラインのミラーのサイズ変更を実行し、DataKeeper ミラーのサイズを変更します。リソースを削除して再作成する必要はありません。ソースシステム上で実行し、ミラーは Out of Service でなければなりません。ミラーのサイズを変更する前に、ベースとなるディスクのサイズを変更する必要があります。ベースとなるディスクのサイズが自動的に検出され、新しいミラーのサイズとして使用されます。

```
mirror_resize [-f] -s <size> <tag>
```

<tag> ミラーリソースのタグ。

-f ユーザに確認せずにサイズ変更を強制実行 (推奨しません)。

-s <size> 代替ミラーサイズ (KB 単位) を指定。このパラメータは必須です。

ミラーのサイズ変更の推奨手順:

1. ミラーおよびすべての従属リソースを Out of Service にします。
2. ベースとなるミラーディスクでディスクサイズを変更します (例: 論理ボリュームの拡張、LUN の拡張)。ソースとターゲットの両方でこれを実行します (ターゲットのサイズはソースのサイズ以上でなければならないことに注意してください)。
3. ソースシステムで `mirror_resize` を実行します。これによってミラーの内部メタデータとビットマップが更新され、新しく拡張したディスクサイズが反映されます。

例: `mirror_resize -s <size in blocks> <tag>`

4. ミラー (`datarep`) リソースのみを In Service にします。新しく拡張したディスク、またはパーティションの再同期が実行されます。
5. ミラーデバイス上でファイルシステムのサイズ変更を実行します (例: `resize2fs /dev/mdX`。X はサイズを変更するミラーの md デバイス番号で、`/dev/md0` のようになります)。注記: ファイルシステムのサイズを変更する前に、`fsck` が必要になる場合があります。
6. ファイルシステムとアプリケーションリソースをオンラインにします。



注記: `mirror_resize` はマルチターゲット/マルチサイト構成ではサポートされません。

ビットマップの管理

```
bitmap -a <num>|-c|-d|-x <size_kb>|-X <bitmap_file>
```

- a <num>ビットマップファイルに非同期書き込みのパラメータを追加します。同期ミラーのアップグレードにより非同期ターゲットを含むようになった場合、これは必須です。<num> のデフォルト値は 256 です。この制限に最適な値を計算するには、**SIOS DataKeeper for Linux** による [ミラーリングの非同期ミラーリング情報](#) を参照してください。
- c ビットマップファイルをクリーニングします (すべてのビットを 0 に設定)。ソースディスクの余分な複製がターゲットに存在する場合、これを使用することで全体の再同期を回避できます。、このオプションは、特に注意して使用してください。
- d ビットマップファイルをダーティに設定します (すべてのビットを 1 に設定)。例えば制御分離の状況が発生した後などに、このオプションを使用して全体の再同期を強制実行できます。
- m ビットマップを読み取り、マージストリームを作成します。
- X <bitmap file> ビットマップファイルを調べて、ビットマップとミラーに関する有用な情報を表示します。
- x <size_kb> size_kb. のディスクが有効になるようにビットマップファイルを拡張します。
(注記: このオプションはミラーのサイズ変更のために内部的にのみ使用されます。)

さらに、mdadm コマンドを使用して、DataKeeper のリソースを管理できます。これは、DataKeeper のリソースが実際には md デバイスに存在するからです。詳細については、mdadm (8) のマニュアルページを参照してください。注記: mdadm を使用するときには、オペレーティングシステムに含まれているバージョンよりも新しい、`$LKROOT/bin` 内のバージョンを必ず使用してください。

コマンドラインからのミラーステータスの監視

通常、ミラーステータスは、LifeKeeper の GUI から、**[Resource Properties]** ダイアログの **[Replication Status]** を使用して確認できます。ただし、以下の操作でもミラーのステータスを監視できます。

```
$LKROOT/bin/mirror_status <tag>
```

例:

```
# mirror_status datarep-ext3-sdr

[-] eve -> adam

    Status: Paused

    Type: Asynchronous

[-] eve -> sophocles

    Status: Resynchronizing

    [=> ] 11%

    Resync Speed: 1573K/sec

    Type: Synchronous
```

サーバの障害

```
Bitmap: 4895 bits (chunks), 4895 dirty (100.0%)
```

以下のコマンドも役に立つことがあります。

```
cat /proc/mdstat
```

サンプルの *mdstat* ファイルを示します。

```
eve:~ # cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 nbd10[1] nbd8[3](F) sdb1[0]
      313236 blocks super non-persistent [3/2] [UU_]
      bitmap: 3/3 pages [12KB], 64KB chunk, file:
/opt/LifeKeeper/bitmap_ext3-sdr
unused devices: <none/></tag>
```

サーバの障害

プライマリサーバとバックアップサーバの両方が動作不能になった場合、DataKeeper リソースは、両方のサーバが再び動作可能になった場合にのみ In Service / アクティブになります。これは、間違った方向への再同期に起因するデータの破損を防ぐためです。動作可能なサーバが、リソースが「**In Service Protected**」(ISP) である最後のサーバであることが確実に分かっている場合は、DataKeeper リソースを右クリックし、**[Force Mirror Online]** を選択してそのリソースを強制的にオンラインにすることができます。

再同期

DataKeeper リソースの再同期中、ターゲットサーバにあるこのリソースインスタンスのステータスは「**Resyncing**」(再同期中)になります。ただし、リソースインスタンスは、プライマリサーバの「ソース」(ISP) です。LifeKeeper の GUI は、ターゲットサーバにある DataKeeper のリソースのステータスを以下のアイコンで表示します。



プライマリサーバにある DataKeeper のリソースは、以下のアイコンで表示されます。



再同期が完了するとすぐに、ターゲットのリソースの状態が「**ターゲット**」になり、アイコンが以下のように変化します。



再同期プロセスについて、以下の点に注意してください。

- SIOS DataKeeper のリソースとその親リソースは、プライマリでの障害発生時に再同期プロセス中のターゲットにはフェイルオーバーできません。
- ターゲットサーバの同期中に DataKeeper リソースが out of service / 非アクティブになった場合、そのリソースは、同じシステム、またはすでに同期済みの別のターゲット(複数のターゲットが存在する場合)でのみ In Service / アクティブにすることができ、再同期が継続されます。
- 再同期プロセス中にプライマリサーバが動作不能になった場合、同期プロセス中のターゲットサーバはすべて、DataKeeper リソースを In Service にすることができません。プライマリサーバが再び動作可能になった後に、ミラーの再同期が継続されます。

全同期の回避

大量のデータを WAN リンク経由で複製する場合、膨大なネットワーク帯域幅と時間を消費する可能性がある全同期は避けることが望ましいです。新しいカーネルと共に使用する場合、SIOS DataKeeper はビットマップテクノロジーを使用して全同期をほぼ防ぐことができます。ただし、既存のデータを複製する場合、ミラーの初期設定時に発生する最初の全同期を回避することはできません(新規データの場合には SIOS DataKeeper は全同期を実行しないので以降の手順は不要です)。

既存のデータを複製するときに、全同期を回避する方法がいくつかあります。ここでは推奨する2とおりの方法を説明します。

方法 1

1 番目の方法では、RAW ディスクイメージを取得してターゲットサイトに輸送します。データがターゲットシステムに到着するまで、ソースシステムのミラーをアクティブにしておくことができるので、この方法ではダウンタイムが最小になります。

手順

1. ミラーを作成してください([Replicate Existing Filesystem] を選択)。ただし、ターゲットシステムにミラーを拡張しないでください。
2. ミラーを out of service にしてください。
3. ソースディスクまたはパーティションのイメージを取得してください。この例では、選択したディスクまたはパーティションは /dev/sda1 です

```
root@source# dd if=/dev/sda1 of=/tmp/sdr_disk.img bs=65536
```

(ブロックサイズの引数 65536 は単に効率的にするためです)。

ディスクまたはパーティションの RAW ディスクイメージを持つファイルが作成されます。

方法 2

ファイルの代わりに、ハードドライブやその他の記憶デバイスも使用できます。

4. オプション手順 - ソースディスクまたはパーティションのチェックサムを取得してください。

```
root@source# md5sum /dev/sda1
```

5. オプション手順 - ディスクイメージファイルを圧縮してください。

```
root@source# gzip /tmp/sdr_disk.img
```

6. ビットマップファイルをクリアしてください。(以下の例の"_dr"の部分はタグ名により異なります。)

```
root@source# /opt/LifeKeeper/bin/bitmap -c /opt/LifeKeeper/bitmap__  
dr
```

7. ミラーと依存ファイルシステム、およびアプリケーション(存在する場合)のサービスを開始してください。ビットマップファイルにより、データがターゲットシステムに転送される間に発生した変更内容が追跡されます。

8. 好みの転送方法を使用して、ターゲットシステムにディスクイメージを転送してください。

9. オプション手順 - ターゲットシステムでディスクイメージファイルを圧縮解除してください。

```
root@target# gunzip /tmp/sdr_disk.img.gz
```

10. オプション手順 - イメージファイルのチェックサムが、手順 4 で取得した元のチェックサムと一致することを確認してください。

```
root@target# md5sum /tmp/sdr_disk.img
```

11. イメージをターゲットシステム(例: /dev/sda2)に転送してください。

```
root@target# dd if=/tmp/sdr_disk.img of=/dev/sda2 bs=65536
```

12. 両方のシステムで、etc/default/LifeKeeper に LKDR_NOFULL_SYNC=1 を設定してください。

```
root@source# echo 'LKDR_NO_FULL_SYNC=1' >>
```

```
/etc/default/LifeKeeper
```

```
root@target# echo 'LKDR_NO_FULL_SYNC=1' >> /etc/default/LifeKeeper
```

13. ミラーをターゲットに拡張してください。部分的な再同期が実行されます。

方法 2

ターゲットシステムを簡単に輸送できる場合、またはシステムの設定時にターゲットシステムがソースと同じ場所にある場合に、この方法を使用できます。この方法では、最初の全同期を高速なローカルネットワークで実行できるように、最終的な WAN ミラーを作成するネットワークルートを LAN ミラーに一時的に変更します。以下の例では、ソースサイトはサブネット 10.10.10.0/24 にあり、ターゲットサイトがサブネット 10.10.20.0/24 にあると仮定しています。ソースとターゲットのシステムの間で一時的に静的ルートを設定することにより、ローカルのイーサネット接続またはループバックケーブルを使用して「WAN」トラフィックをあるサーバから別のサーバに直接送信できます。

手順

1. ソースサイトでシステムをインストールし、設定してください。
2. 静的ルートを追加してください。

```
root@source# route add -net 10.10.20.0/24 dev eth0
```

```
root@target# route add -net 10.10.10.0/24 dev eth0
```

この時点で、両方のシステムがLAN上で相互に通信できる必要があります。

3. LifeKeeperでコミュニケーションパスを設定してください。
4. ミラーを作成し、ターゲットに拡張してください。全同期が実行されます。
5. ミラーをPauseにしてください。ミラーが再開されるまで、変更内容はビットマップファイルで追跡されます。
6. 静的ルートを削除してください。

```
root@source# route del -net 10.10.20.0/24
```

```
root@target# route del -net 10.10.10.0/24
```

7. ターゲットシステムをシャットダウンし、恒久的に配置する場所に輸送してください。
8. ターゲットシステムを起動し、ソースとのネットワーク接続を確立してください。
9. Resume the mirrorを実行してください。部分的な再同期が実行されます。

DataKeeperでLVMを使用する

SPS for Linuxは現在LVM上でのDataKeeperの使用とDataKeeper上でのLVMの使用の両方をサポートしています。標準的なDataKeeperの設定では、LVM上でのDataKeeper使用がサポートされ、SPS LVM Recovery Kitは不要です。必要なリカバリキットは、DataKeeperだけです。ただし、DataKeeper設定上でLVMを使用する場合は、LVM Recovery Kitが必要になります。

SIOSではLVM上でのDataKeeper使用を推奨していますが、DataKeeper設定上でLVMが使用される場合は、2段階の階層作成プロセスが必要になります。DataKeeperデバイス(階層)では、プライマリサーバ上でLVMボリュームグループと論理ボリュームを作成する前に、DataKeeperの「Data Replication Resource」オプションを使用した設定を行う必要があります。必要なボリュームグループと論理ボリュームが作成できたら、残りの階層を保護されるアプリケーションに関連したリカバリキットの設定手順に従って作成します。完成した階層は以下に示す図3のようになります。**注記:**データの整合性を保つため、DataKeeper上にLVMを構成する場合には、DataKeeperミラー1つだけ、または複数の同期ミラー、いずれかの構成でなければなりません。

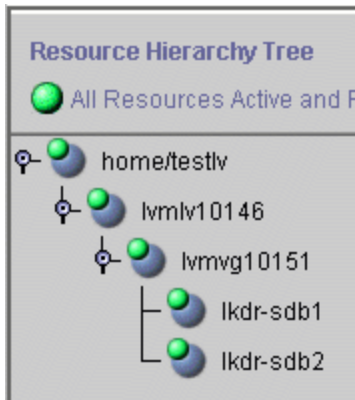
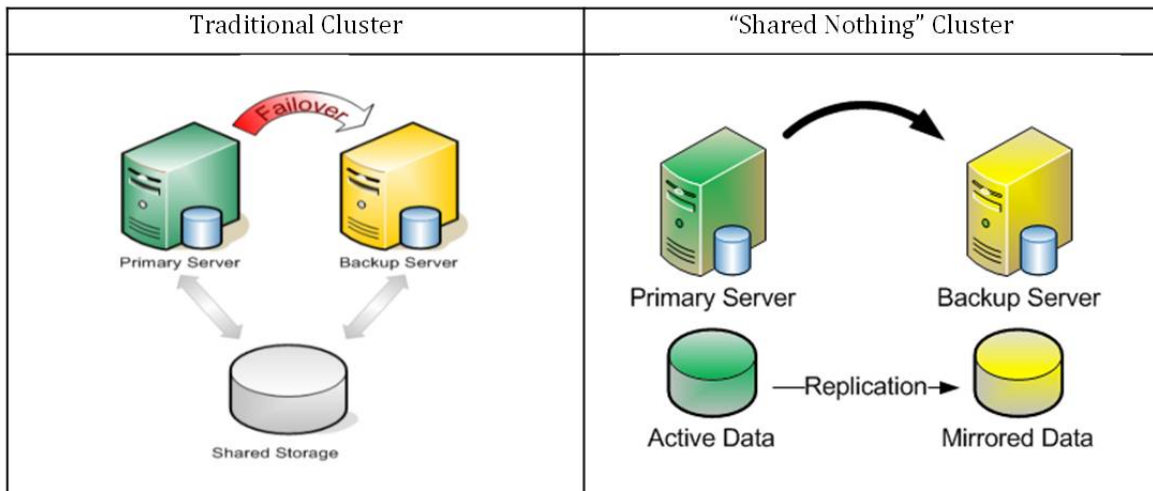


図 3: DataKeeper 上の LVM を含む階層

Fusion-io を使用するクラスタ化

DataKeeper のパフォーマンスを最大に発揮させるための Fusion-io のベストプラクティス

SPS for Linux はブロックレベルのデータレプリケーション機能を備えており、共有ストレージを使用しないHAクラスタを非常に簡単に設定できます。Fusion-io を使用すると、SPS for Linux ではフェイルオーバー保護用として「shared nothing」クラスタを構成できます。



クラスタ設定の一部としてデータレプリケーションを活用する場合、ディスクへの書き込みと同等な速度でデータがネットワーク上で複製可能なだけの十分な帯域幅があることが重要です。以下のベストプラクティスを使用すると、高速ストレージを使用する場合に「shared nothing」SPS クラスタ設定を最大に活用できます。

ネットワーク

- **10 Gbps の NIC を使用する。** Fusion-io のフラッシュベースのストレージデバイス (または OCZ、LSI、などの同様な製品) は、750 MB/秒を超えるデータ速度でデータを書き込むことができます。1 Gbps の NIC は理論的な最大値である約 125 MB/秒のみを送出できるので、ioDrive の性能を活用すると、1 Gbps のネットワーク接続での複製よりもはるかに高速で簡単にデータを書き込むことができます。リアルタイムのデータレプリケーションを実行するのに十分な帯域幅をサーバ間に確保するには、レプリケーショントラフィックの転送に必ず 10 Gbps の NIC を使用してください。
- **ジャンボフレームを有効にする。** 使用しているネットワークカードとスイッチでサポートされている場合、ジャンボフレームを有効にするとネットワークのスループットが増加するだけでなく、CPU サイクルが減少します。ジャンボフレームを有効にするには、以下の設定を実行してください (RedHat/CentOS/OEL Linux ディストリビューションの例)。

- 次のコマンドを実行してください。

```
ifconfig <interface_name> mtu 9000
```

- 再起動後も変更内容を保持するには、以下のファイルに「MTU=9000」を追加してください。

```
/etc/sysconfig/network-scripts/ifcfg-<interface_name>
```

- エンドツーエンドのフレーム動作を検証するには、次のコマンドを実行してください。

```
ping -s 8900 -M do <IP-of-other-server>
```

- **NIC の転送キュー長を変更する。**

- 次のコマンドを実行してください。

```
/sbin/ifconfig <interface_name> txqueuelen 10000
```

- 再起動後も変更内容を保持するには、`/etc/rc.local` に追加してください。

- **NIC の netdev_max_backlog を変更する。**

- 次の設定を `/etc/sysctl.conf` に追加してください。

```
net.core.netdev_max_backlog = 100000
```

TCP/IP の調整

- レプリケーションのパフォーマンスを向上することが確認された **TCP/IP の調整** を以下に示します。
 - `/etc/sysctl.conf` を編集して、以下のパラメータを追加してください (注記: これらは例であり、使用している環境で異なる場合がある)。

```
net.core.rmem_default = 16777216
```

```
net.core.wmem_default = 16777216
```

```
net.core.rmem_max = 16777216
```

```
net.core.wmem_max = 16777216
```

設定上の推奨項目

```
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.core.optmem_max = 16777216
net.ipv4.tcp_congestion_control=htcp
```

設定上の推奨項目

- Fusion-ioドライブ上に、ビットマップファイルを入れる小型の(約 100 MB)のディスクパーティションを割り当ててください。このパーティションにファイルシステムを作成し、マウントしてください(例: */bitmap*)。

```
# mount | grep /bitmap
/dev/fioa1 on /bitmap type ext3 (rw)
```

- ミラーを作成する前に、*/etc/default/LifeKeeper*内の以下のパラメータを調整してください。

- LKDR_CHUNK_SIZE=4096

- デフォルト値は64です。

- LKDR_SPEED_LIMIT=1500000

- デフォルト値は50000です。

- LKDR_SPEED_LIMITは、再同期に使用する最大帯域幅を指定します。可能な最大速度で再同期が実行されるように、この値を十分高い値に設定する必要があります。

- LKDR_SPEED_LIMIT_MIN=200000

- デフォルト値は20000です。

- LKDR_SPEED_LIMIT_MINは、同時に他のI/Oが実行されているときに許可する再同期の速度を指定してください。再同期の実行時に通常のI/O動作が妨げられないようにするには、経験則として、この値をドライブの最大書き込みスループットの半分以下に設定する必要があります。

- 通常と同様に、ミラーを作成してクラスタを設定してください。

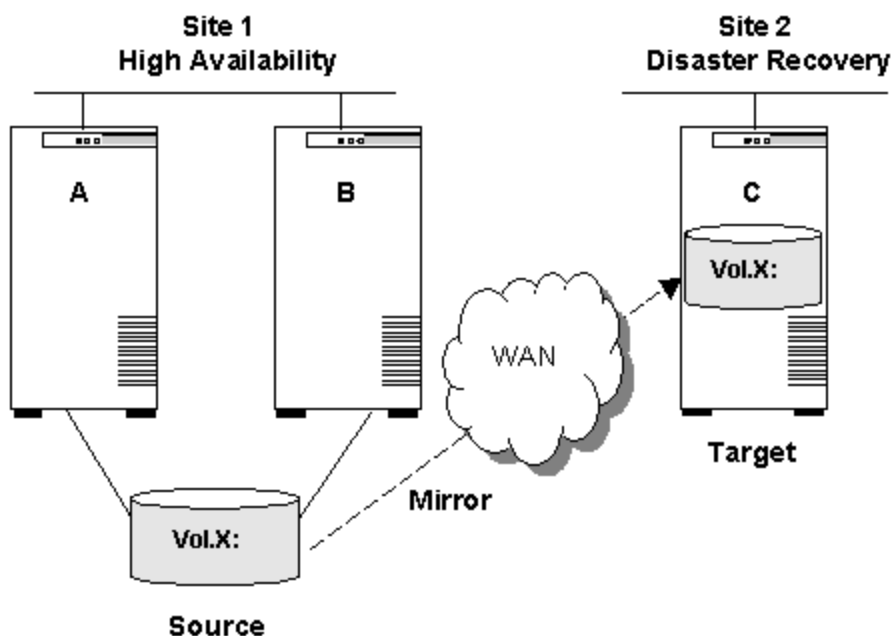
Multi-Site Cluster

SIOS Protection Suite for Linux Multi-Site Cluster

SIOS Protection Suite for Linux Multi-Site Cluster は別のライセンス製品であり、2 台以上のサーバ間で LifeKeeper の共有ストレージ構成を使用し、さらに SIOS DataKeeper for Linux を使用して共有ディスクを 1 台以上のターゲットサーバに複製する機能を持ちます。

SIOS Protection Suite for Linux Multi-Site Cluster

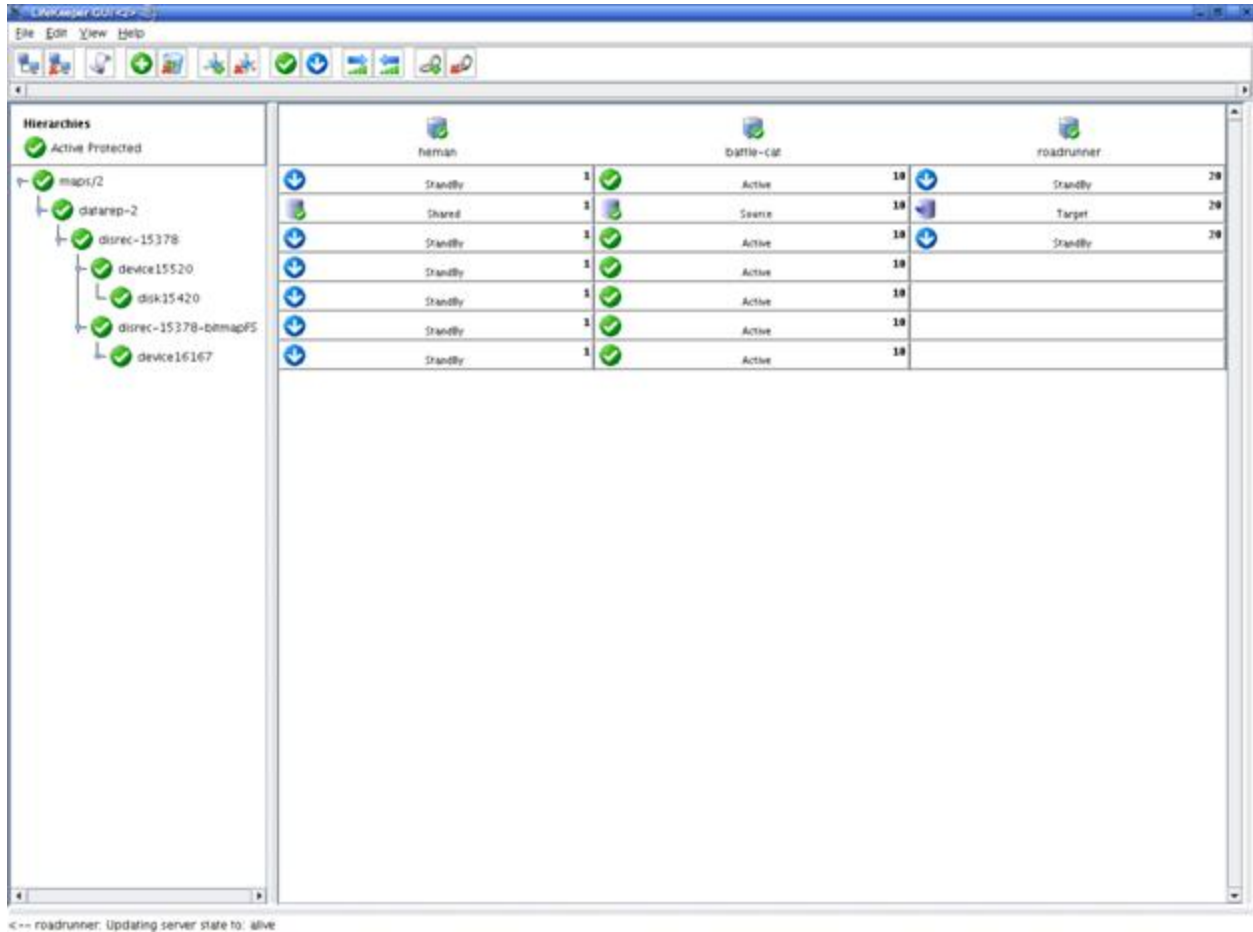
SIOS Protection Suite for Linux Multi-Site Cluster は、LifeKeeper を使用して 2 台以上のサーバ間で共有ストレージを構成し、その共有ディスクを SIOS DataKeeper を使用して、1 台以上のサーバへミラーリングを構成する付加機能のための個別ライセンス製品です。



SIOS Protection Suite for Linux Multi-Site Cluster は、異なるサブネットに存在する複数のネットワークセグメントにわたって IP アドレスのフェイルオーバーを提供するように構成されたワイドエリアネットワークに組み込むことができます。この構成には、仮想ネットワーク(仮想 LAN(VLAN))と仮想プライベートネットワーク(VPN)が含まれます。

以下の画像は、SIOS Protection Suite for Linux Multi-Site Cluster 製品を構成した後の SIOS LifeKeeper の GUI です。階層の釣り合いが取れていないように見えますが、階層は適切に構成されており、正しく機能しま

すでに SIOS DataKeeper を使用していて SIOS LifeKeeper のグラフィカルユーザインターフェースに慣れている場合、LifeKeeper の GUI での SIOS Protection Suite Multi-Site Cluster リソース階層の表示は、旧リリースの SIOS DataKeeper とは異なります。



Multi-Site Cluster を設定する際の考慮事項

ローカルサイトのビットマップの保存先

ローカルサイトのノード間にはミラーリング対象の共有ディスクの他に、ビットマップファイル保存専用の共有ファイルシステムリソースが必要です。

マルチサイトクラスタ構成では、ローカルノード間で共有されるビットマップ専用のファイルシステムをビットマップファイルの保存用に割り当てる必要があります。そのビットマップファイルは、保護されているレプリケーションされたファイルシステムの変更履歴に使用されます。これによって、ローカルノード間のシームレスなスイッチオーバーとフェイルオーバーが可能になります。レプリケーションされたファイルシステムへのすべての変更は、ローカルノード間の共有

ファイルシステム上のビットマップファイルに履歴が残ります。LUN設定の際に考慮すべき補足事項は以下の通りです。

- ファイルシステムは、ローカルノード間でのみ、ビットマップ保存専用である必要があります。
- 設定において、追加のレプリケーションファイルシステムごとにビットマップ保存のための1 LANを追加してください。**他のデータレプリケーションリソースとLUNの共有はできません。**
- ファイルシステムは、レプリケーションされたファイルシステムのビットマップに対応できる十分なサイズである必要があります。ビットマップファイルを保存する為に必要なサイズは、レプリケーションされたファイルシステムのサイズに基づいて計算されます。初期設定を使用する際、レプリケーションされたファイルシステムにおいて64KBのスペースごとに1ビットがビットマップで割り当てられます。例えば、64MBのレプリケーションされたファイルシステムサイズは、1000ビットのビットマップファイルを必要とします。

マルチサイトクラスタ設定で避けるべきリソース構成

システムの構成を始める前に、Linux のマルチサイトクラスタ階層の環境では避けるべき階層構成を理解しておくことが重要です。

以下に、Linux Multi-Site Cluster 環境で避ける必要のある階層構成の例を3つ示します。これらすべての例で、Linux Multi-Site Cluster 階層は、下にあるデバイスを別の階層と共有しています。いずれかの階層で障害またはスイッチオーバーが起こると、関連する階層が影響を受けます。これにより、アプリケーションの障害やミラーの破損など、予期しない結果が起こる可能性があります。この場合、後で全同期プロセスを実行する必要があります。さらに、ミラーソースから DR サイトに切り替えて DR サイトからプライマリサイトへのミラーバックを許可すると、事態が複雑になることがあります。これは、ミラーターゲットシステムが下位レベルのディスクリソースを In Service にしているからです。すべての共有リソースも、ミラーターゲットと同じノードで動作可能 (ISP) にする必要があります。

例:	説明
1	Multi-Site Cluster 階層のミラーディスクリソースを、複数回、同じ階層または別の階層で使用する。
2	ミラービットマップ用に、同じ Multi-Site Cluster のファイルシステムまたはディスクリソースを、複数の Multi-Site Cluster 階層で使用する(各ミラーのビットマップファイルは、一意の LUN に存在する必要があり、共有できない)。
3	ビットマップファイルシステム、デバイス、またはディスクリソースを、別の階層(マルチサイトまたは非マルチサイト)で使用する。

Multi-Site Cluster の設定上の注意点

その他のマルチサイトクラスタを設定する際の注意点は次の通りです。

- Linux Multi-Site Cluster を使用する場合、SIOS Logical Volume Manager Recovery Kitをディザスタリカバリノードにインストールしないでください。

SIOS Protection Suite for Linux Multi-Site Cluster リソース階層の作成

プライマリサーバで以下の操作を行ってください。

1. **[Edit] > [Server] > [Create Resource Hierarchy]** を選択してください。
[Create Resource Wizard] ダイアログボックスが表示されます。
2. ドロップダウンリストから **[Data Replication]** オプションを選択し、**[Next]** をクリックして続行してください。
3. 以下の情報を入力するように要求されます。ダイアログボックスで **[Back]** ボタンが有効な場合は、前のダイアログボックスに戻ることができます。これは、エラーが発生して、前に入力した情報を修正する必要がある場合に便利な機能です。いつでも **[Cancel]** をクリックして、作成処理全体を取り消すことができます。

フィールド	ヒント
Switchback Type	[intelligent switchback] を指定する必要があります。これは、バックアップサーバにフェイルオーバーした後、管理者が手動で Multi-Site Cluster リソースをプライマリサーバにスイッチバックする必要があることを意味します。 注意: このリリースの SIOS DataKeeper は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、Multi-Site Cluster 階層を構成する LifeKeeper リソースにも適用されます。この制限の対象として、階層の上に存在するもの、または階層内の子が含まれます。
Server	NetRAID デバイスを作成するサーバ(通常はプライマリサーバ)の名前を選択してください。ドロップダウンリストボックスには、クラスタ内のすべてのサーバが表示されます。
Hierarchy Type	以下のいずれかを選択して、作成するデータ複製のタイプを選択してください。 <ul style="list-style-type: none"> • Replicate New File System • Replicate Existing File System • DataKeeper Resource

以降の一連のダイアログボックスは、**[Hierarchy Type]** で選択した項目によって異なります。一部のダイアログボックスはすべての階層タイプで同じですが、表示される順序と必要な情報が少し異なることがあります。以下の3つのトピックで、階層作成の残りのプロセスについて説明しています。

- [Replicate New File System](#)
- [Replicate Existing File System](#)
- [DataKeeper Resource](#)

Replicate New File System

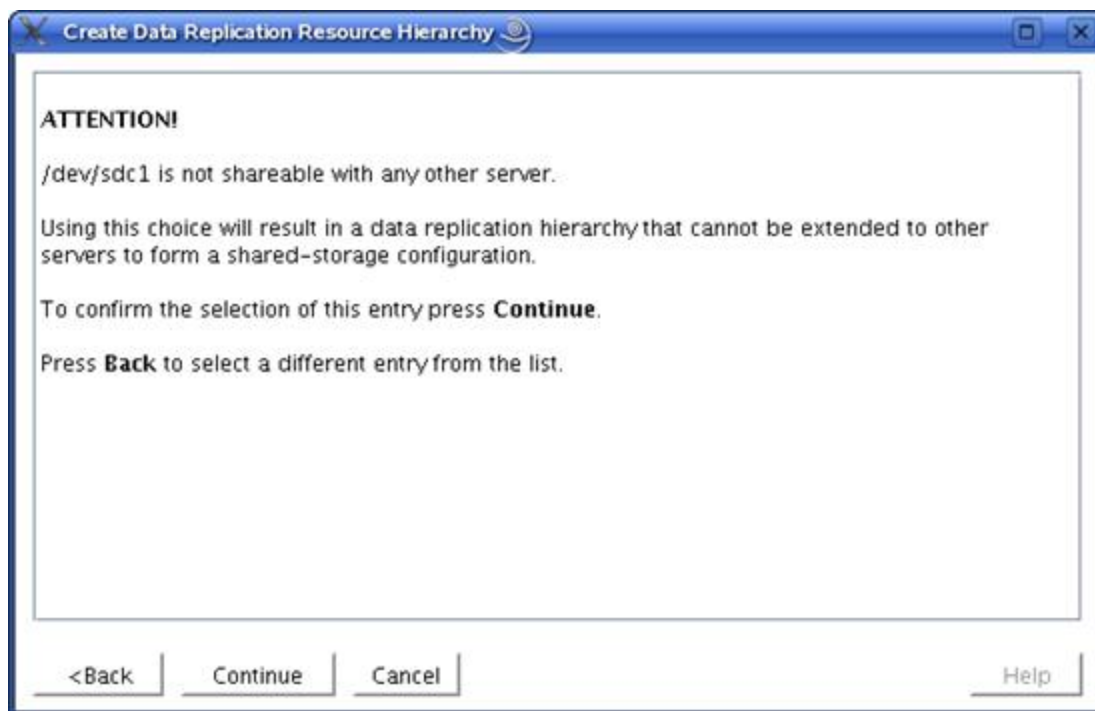
このオプションは、NetRAID デバイスを作成し、LifeKeeper がサポートするファイルシステムタイプでフォーマットします。ファイルシステムを NetRAID デバイスにマウントし、マウントしたファイルシステムと NetRAID デバイスの両方を LifeKeeper で保護します。NetRAID デバイスとローカルのディスクまたはパーティションがフォーマットされ、既存のデータが削除されます。新しいファイルシステムにミラーを作成し、LifeKeeper で保護する場合にこのオプションを選択してください。このリソースタイプには、1 つの空いているディスクまたはパーティションが必要です。

注意: このオプションを選択すると、ローカルのディスクまたはパーティションがフォーマットされ、既存のデータがすべて削除されます。

1. 要求されたら、以下の情報を入力してください。

フィールド	ヒント
Source Disk or Partition	<p>ドロップダウンリストには、<u>以下のものを除いて</u>、使用できるすべてのディスクが表示されます。</p> <ul style="list-style-type: none"> • 現在マウントされているもの • スワップディスクまたはスワップパーティション • LifeKeeper が保護するディスクまたはパーティション <p>ドロップダウンリストには、root (/)、boot (/boot)、/proc、floppy、cdrom などの特殊なディスクまたはパーティションも表示されません。</p>

2. 非共有のソースのディスクまたはパーティションを選択した場合、以下の画面が表示されます。



3. 共有のソースのディスクまたはパーティションを選択するには、**[Back]**を選択してください。残りの情報を指定して、SIOS Protection Suite for Linux Multi-Site Cluster リソースの構成を完了してください。

フィールド	ヒント
New Mount Point	新しいファイルシステムの 新しいマウントポイント を入力してください。これは、複製したディスクまたはパーティションが配置されるマウントポイントです。
New File System Type	ファイルシステムタイプ を選択します。LifeKeeper がサポートするファイルシステムタイプのみを選択できます。
DataKeeper Resource Tag	DataKeeper リソースインスタンスの一意の DataKeeper リソースタグ名を選択するか、入力してください。
File System Resource Tag	ファイルシステムリソースインスタンスのファイルシステムリソースタグを選択するか、入力してください。

フィールド	ヒント
Bitmap File	<p>プルダウンリストからビットマップファイルの項目を選択してください。</p> <p>表示されたリストには、ビットマップファイルの保持に使用できる共有ファイルシステムがあります。\$LKROOT/bin ディレクトリを参照)。ビットマップファイルは、階層内のローカルノード間で切り替え可能な共有デバイスに配置する必要があります。</p> <p>重要: ビットマップファイルは btrfs ファイルシステム上に置いてはいけません。データレプリケーションのビットマップファイルが btrfs ファイルシステム上に置かれると、LifeKeeper がミラーを構成しようとした時、"invalid argument" エラーの原因になります。ビットマップファイルのデフォルトの置き場所は、/opt/LifeKeeper の下です。このデフォルトの置き場所は、/opt/LifeKeeper が btrfs ファイルシステム上にある場合変更されます。</p>

4. **[Next]** をクリックして、**確認** 画面に進んでください。
5. 確認画面に、新しいファイルシステムの作成場所、およびローカルのディスクまたはパーティションについて保留中の再フォーマットに関する警告が表示されます。**[Create]** をクリックして、**リソースの作成**を開始します。
6. リソースを新しいファイルシステムに作成するために有効なデータを指定したかどうか、LifeKeeper により検証されます。LifeKeeper が問題を検知した場合は、情報ボックスにエラーが表示されます。検証が正常に完了すると、リソースが作成されます。ディスクまたはパーティションのサイズにより、ファイルシステムの作成には数分かかることがあります。

[Next] をクリックして次に進んでください。

7. 新しい複製ファイルシステムのリソース階層が正常に作成されたことを示す情報ボックスが表示されます。複製を開始してリソース階層を LifeKeeper で保護するには、クラスタ内の別のサーバにリソース階層を**拡張**する必要があります。

リソースを拡張する場合は **[Next]**、後でリソースを拡張する場合は **[Cancel]** をクリックしてください。

[Continue] をクリックすると、**Pre-extend Wizard** が起動します。リソース階層を別のサーバに拡張する方法の詳細については、[リソース階層の拡張](#)の手順 2 を参照してください。

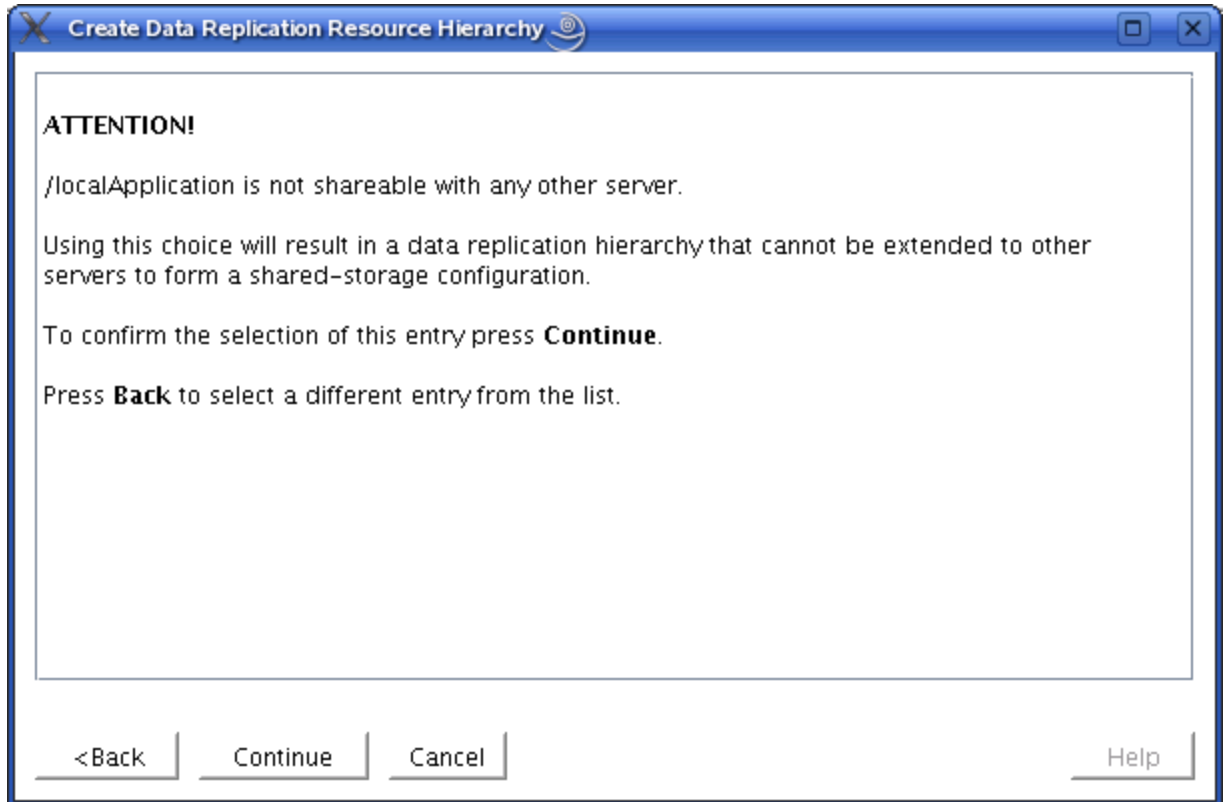
Replicate Existing File System

このオプションは、ローカルのディスクまたはパーティションに現在マウントされているファイルシステムをアンマウントし、NetRAID デバイスを作成して、ファイルシステムを NetRAID デバイスに再マウントします。NetRAID デバイスとマウントされたファイルシステムの両方が、LifeKeeper で保護されます。既存のファイルシステムにミラーを作成し、LifeKeeper で保護する場合にこのオプションを選択してください。

1. 要求されたら、以下の情報を入力してください。

フィールド	ヒント
Existing Mount Point	これは、プライマリサーバの NetRAID デバイスにマウントするマウントポイントです。ローカルのディスクまたはパーティションがすでに、このマウントポイントにマウントされている必要があります。

2. 非共有のソースのマウントポイントを選択した場合、以下の画面が表示されます。



3. 共有のソースのディスクまたはパーティションを選択するには、**[Back]**を選択してください。残りの情報を指定して、SIOS Protection Suite for Linux Multi-Site Cluster リソースの構成を完了してください。

フィールド	ヒント
DataKeeper Resource Tag	DataKeeper リソースインスタンスの一意の DataKeeper リソースタグ名を選択するか、入力してください。
File System Resource Tag	ファイルシステム リソースタグの名前を選択するか、入力してください。

フィールド	ヒント
Bitmap File	<p>プルダウンリストからビットマップファイルの項目を選択してください。</p> <p>表示されたリストには、ビットマップファイルの保持に使用できる共有ファイルシステムがあります。\$LKROOT/bin ディレクトリを参照)。ビットマップファイルは、階層内のローカルノード間で切り替え可能な共有デバイスに配置する必要があります。</p> <p>重要: ビットマップファイルは btrfs ファイルシステム上に置いてはいけません。データレプリケーションのビットマップファイルが btrfs ファイルシステム上に置かれると、LifeKeeper がミラーを構成しようとした時、"invalid argument" エラーの原因になります。ビットマップファイルのデフォルトの置き場所は、/opt/LifeKeeper の下です。このデフォルトの置き場所は、/opt/LifeKeeper が btrfs ファイルシステム上にある場合変更されます。</p> <p>重要: プルダウンリストには、レプリケーション対象となる共有ディスク領域も表示されますが、これをビットマップファイルの保存領域として選択しないでください。ビットマップファイルの保存先に、レプリケーション対象となる共有ファイルシステムを使用することはできません。必ずレプリケーション領域とは別の、ビットマップ専用を追加した共有ファイルシステムを選択するようにしてください。</p>

4. **[Next]** をクリックして、DataKeeper リソースをプライマリサーバに作成してください。
5. DataKeeper リソースの作成するために有効なデータを指定したかどうか、LifeKeeper により検証されます。LifeKeeper が問題を検知した場合は、情報ボックスにエラーが表示されます。検証が正常に完了すると、リソースが作成されます。

[Next] をクリックしてください。

6. 既存のレプリケーションファイルシステムのリソース階層が正常に作成されたことを示す情報ボックスが表示されます。レプリケーションを開始してリソース階層を LifeKeeper で保護するには、クラスタ内の別のサーバにリソース階層を拡張する必要があります。

リソースを拡張する場合は **[Next]**、後でリソースを拡張する場合は **[Cancel]** をクリックしてください。

[Continue] をクリックすると、*Pre-extend Wizard* が起動します。リソース階層を別のサーバに拡張する方法の詳細については、リソース階層の拡張の手順 2 を参照してください。

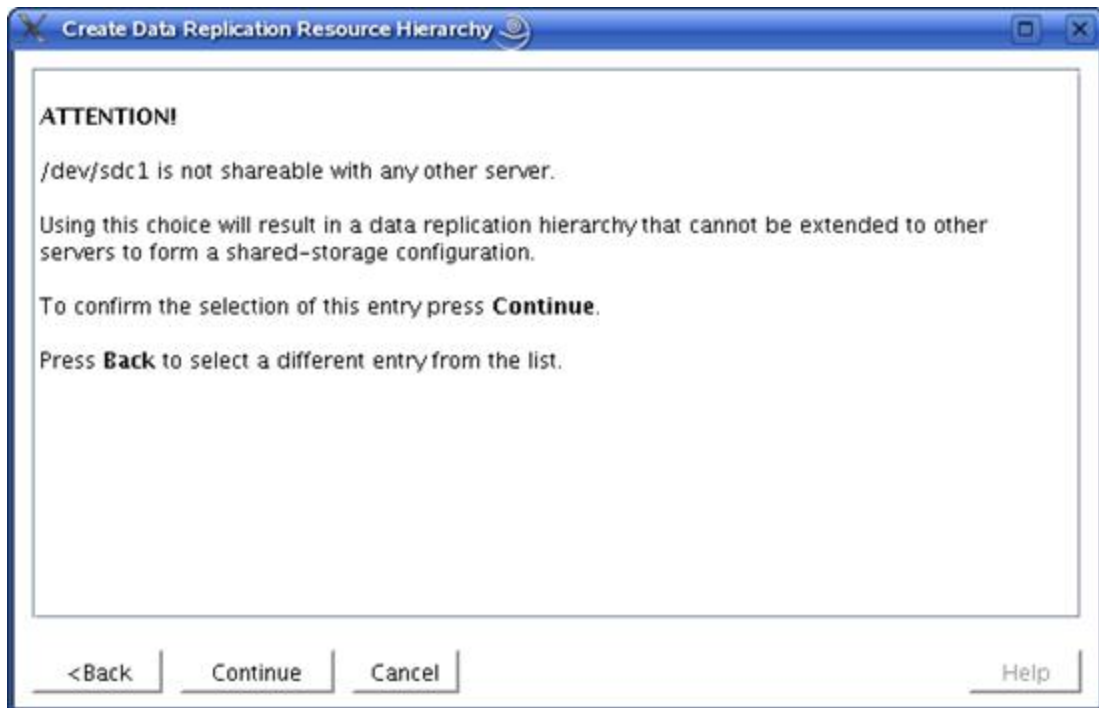
DataKeeper Resource

このオプションは、NetRAID デバイスのみを作成し(ファイルシステムは作成しない)、NetRAID デバイスを LifeKeeper で保護します。ディスクまたはパーティション上に DataKeeper デバイスのみを作成し、LifeKeeper で保護する場合にこのオプションを選択してください。読み取り可能なミラーを作成するには、このデバイス上にファイルシステムを作成し、マウントする操作を手動で行う必要があります。このリソースタイプには、1つの空いているディスクまたはパーティションが必要です。

1. 要求されたら、以下の情報を入力してください。

フィールド	ヒント
Source Disk or Partition	<p>ドロップダウンボックスのソースディスクまたはパーティションのリストには、<u>以下のものを除いて</u>、使用できるすべてのディスクが表示されます。</p> <ul style="list-style-type: none"> • 現在マウントされているもの • スワップディスクまたはスワップパーティション • LifeKeeper が保護するディスクまたはパーティション <p>ドロップダウンリストには、root (/)、boot (/boot)、/proc、floppy、cdrom などの特殊なディスクまたはパーティションも表示されません。</p> <p>注記: VMware を使用する場合は、「VMware の既知の問題」を参照してください。</p>

2. 非共有のソースのディスクまたはパーティションを選択した場合、以下の画面が表示されます。



3. 共有のソースのディスクまたはパーティションを選択するには、**[Back]** を選択してください。残りの情報を指定して、SIOS Protection Suite for Linux Multi-Site Cluster リソースの構成を完了してください。

フィールド	ヒント
DataKeeper Resource Tag	DataKeeper リソースインスタンスの一意の DataKeeper リソースタグ名 を選択するか、入力してください。
Bitmap File	<p>プルダウンリストからビットマップファイルの項目を選択してください。</p> <p>表示されたリストには、ビットマップファイルの保持に使用できる共有ファイルシステムがあります。\$LKROOT/bin ディレクトリを参照)。ビットマップファイルは、階層内のローカルノード間で切り替え可能な共有デバイスに配置する必要があります。</p> <p>重要: ビットマップファイルは btrfs ファイルシステム上に置いてはいけません。データレプリケーションのビットマップファイルが btrfs ファイルシステム上に置かれると、LifeKeeper がミラーを構成しようとした時、"invalid argument" エラーの原因になります。ビットマップファイルのデフォルトの置き場所は、/opt/LifeKeeper の下です。このデフォルトの置き場所は、/opt/LifeKeeper が btrfs ファイルシステム上にある場合変更されます。</p>

4. **[Next]** をクリックします。
5. 使用する前に、ファイルシステムを手動で作成し、NetRAID デバイス(/dev/mdX)にマウントする必要があることを示す情報ウィンドウが表示されます。

[Create] をクリックして、DataKeeper デバイスをローカルのディスクまたはパーティションに作成してください。

6. 情報ボックスが表示され、DataKeeper リソースの作成のために有効なデータを指定したかどうか、LifeKeeper により検証されます。LifeKeeper が問題を検知した場合は、情報ボックスにエラーが表示されます。検証が正常に完了すると、リソースが作成されます。

[Next] をクリックして次に進んでください。

7. DataKeeper リソースデバイスが正常に作成されたことを示す情報ボックスが表示されます。データの複製を開始し、バックアップ/ターゲットサーバを LifeKeeper で保護するには、クラスタ内の別のサーバに階層を拡張する必要があります。

リソースを拡張する場合は **[Continue]**、後でリソースを拡張する場合は **[Cancel]** をクリックしてください。

[Continue] をクリックすると、**Pre-extend Wizard** が起動します。リソース階層を別のサーバに拡張する方法の詳細については、[リソース階層の拡張](#)の手順 2 を参照してください。

リソース階層の拡張

この操作は **[Edit]** メニューから、プライマリサーバからセカンダリサーバに開始する必要があります。または **[Create Resource Hierarchy]** オプションの動作が完了すると自動的に開始されます。その場合は、手順 2 を参照してください。

1. **[Edit]** メニューの **[Resource]** から **[Extend Resource Hierarchy]** を選択してください。 **Pre-Extend Wizard** が表示されます。拡張操作に慣れていない場合は、**[Next]** をクリックしてください。LifeKeeper の **[Extend Resource Hierarchy]** のデフォルト値が分かっている、入力と確認を省略する場合は **[Accept Defaults]** をクリックしてください。
2. **Pre-Extend Wizard** に以下の情報を入力します。

フィールド	ヒント
Template Server	<p>DataKeeper リソースが現在 in service のテンプレートサーバを選択してください。ここで選択するテンプレートサーバと次のダイアログボックスで選択する拡張するタグによって、サービス中 (アクティブ) のリソース階層が表示されることを理解しておくことが重要です。</p> <p>選択したテンプレートサーバで in service でないリソースタグを選択した場合、エラーメッセージが表示されます。このダイアログのドロップダウンボックスに、クラスタ内の全サーバの名前が表示されます。</p>
Tag to Extend	<p>これは、テンプレートサーバからターゲットサーバに拡張する DataKeeper インスタンスの名前です。ドロップダウンボックスには、テンプレートサーバ上に作成したすべてのリソースが表示されます。</p>
Target Server	<p>拡張先のサーバを入力するか、選択してください。</p>
Switchback Type	<p>[intelligent switchback] を指定する必要があります。これは、バックアップサーバにフェイルオーバーした後、管理者が手動で Multi-Site Cluster 階層のリソースをプライマリサーバにスイッチバックする必要があることを意味します。</p> <p>注意: このリリースの DataKeeper for Linux は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、マルチサイトクラスタ階層を構成する LifeKeeper リソースにも適用されます。この制限の対象として、階層の上位あるいは下位にあるリソースも含まれます。</p>
テンプレートの優先順位	<p>[Template Priority] を選択または入力します。これはサーバで現在 in service の DataKeeper 階層の優先順位です。優先順位は、1 ~ 999 の範囲で未使用の値が有効で、小さい数字ほど優先順位が高くなります (数字 1 が最高の優先順位に相当)。拡張処理時に、別のシステムですでに使用中の優先順位をこの階層に対して指定することはできません。デフォルト値を推奨します。</p> <p>注記: このフィールドは階層を最初に拡張するときだけ表示されます。</p>
Target Priority	<p>Target の優先順位 を選択するか、入力してください。これは、他のサーバにある同等の階層に対する、新しく拡張する DataKeeper 階層の優先順位です。1 ~ 999 の範囲で、まだ優先順位として使用されていない値が有効で、リソースのカスケディングフェイルオーバーシナリオにおけるサーバの優先順位を示します。数値が小さいほど優先順位は高くなります (1 は最高の優先順位を表す)。LifeKeeper のデフォルトでは、階層が作成されたサーバに「1」が割り当てられることに注意してください。優先順位は連続している必要はありませんが、特定のリソースについて 2 つのサーバに同じ優先順位を割り当てることはできません。</p>

3. 拡張前のチェックが正常に終了したというメッセージが表示された後、**[Next]** をクリックしてください。
4. 拡張する階層に応じて、拡張されるリソースタグ (一部編集不可) を示す一連の情報ボックスが表示されます。

リソース階層の拡張を実行する場合は、**[Next]** をクリックしてください。

次のセクションには、別のサーバに DataKeeper リソースを拡張するために必要な手順を示します。

DataKeeper リソース階層の拡張

1. pre-extend スクリプトが正常に実行されたというメッセージが表示されたら、以下の情報を指定するように要求されます。

フィールド	ヒント
Mount Point	ターゲットサーバ上にあるファイルシステムのマウントポイント名を入力してください (DataKeeper リソースに関連する、LifeKeeper が保護するファイルシステムがない場合は、このダイアログは表示されません)。
Root Tag	ルートタグを選択するか、入力してください。これは、ターゲットサーバ上にあるファイルシステムリソースインスタンスの一意の名前です (DataKeeper リソースに関連する、LifeKeeper が保護するファイルシステムがない場合は、このダイアログは表示されません)。
DataKeeper Resource Tag	DataKeeper リソースタグ の名前を選択するか、入力してください。
Bitmap File	<p>インテントログの記録に使用するビットマップファイルの名前を選択してください。[None]を選択すると、インテントログは使用されず、すべての再同期が部分的ではなく全体の再同期になります。</p> <p>重要: ビットマップファイルは btrfs ファイルシステム上に置いてはいけません。データレプリケーションのビットマップファイルが btrfs ファイルシステム上に置かれると、LifeKeeper がミラーを構成しようとした時、"invalid argument" エラーの原因になります。ビットマップファイルのデフォルトの置き場所は、<code>/opt/LifeKeeper</code> の下です。このデフォルトの置き場所は、<code>/opt/LifeKeeper</code> が btrfs ファイルシステム上にある場合変更されます。</p>

2. **[Next]** をクリックして次に進んでください。拡張を実行中であることを確認する情報ボックスが表示されません。
3. **[Finish]** をクリックして、DataKeeper リソースインスタンスが正常に拡張されたことを確認してください。
4. **[Done]** をクリックして、**[Extend Resources Hierarchy]** メニューを終了してください。

注記: 必ずすべてのサーバで手動スイッチオーバーを実行して、新しいインスタンスの機能をテストしてください。詳細については、[リソース階層のテスト](#)を参照してください。この時点で、DataKeeper がソースからターゲットのディスクまたはパーティションにデータの再同期を開始しています。LifeKeeper の GUI では、ターゲットサーバにある DataKeeper リソースのステータスは「**Resyncing**」になります。再同期が完了すると、ステータスは「**Target**」になります。これは通常の**スタンバイ**状態です。

再同期中、DataKeeper リソース、およびそれに依存するリソースはフェイルオーバーできません。これは、データの破損を防止するためです。

ディザスタリカバリシステムへの階層の拡張

この操作は、ISP ノードから、または複数ノードの作成プロセスの一環として **[Edit]** メニューからのみ実行できま

す。または **[Create Resource Hierarchy]** オプションの動作が完了すると自動的に開始されます。その場合は、手順 2 を参照してください。

1. **[Edit]** メニューの **[Resource]** から **[Extend Resource Hierarchy]** を選択してください。 **Pre-Extend Wizard** が表示されます。拡張操作に慣れていない場合は、 **[Next]** をクリックしてください。LifeKeeper の **[Extend Resource Hierarchy]** のデフォルト値が分かっている、入力と確認を省略する場合は **[Accept Defaults]** をクリックしてください。
2. **Pre-Extend Wizard** に以下の情報を入力します。

注記: 最初の 2 つのフィールドは **[Edit]** メニューから拡張を開始した場合にのみ表示されます。

フィールド	ヒント
Target Server	拡張先のサーバを入力するか、選択してください。
Switchback Type	[intelligent switchback] を指定する必要があります。これは、バックアップサーバにフェイルオーバーした後、管理者が手動で Multi-Site Cluster 階層のリソースをプライマリサーバにスイッチバックする必要があることを意味します。 注意: このリリースの SIOS DataKeeper for Linux は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、Multi-Site Cluster 階層を構成する LifeKeeper リソースにも適用されます。この制限の対象として、階層の上位あるいは下位にあるリソースも含まれます。
Target Priority	ターゲットの優先順位 を選択するか、入力してください。これは、他のサーバにある同等の階層に対する、新しく拡張する DataKeeper 階層の優先順位です。1 ~ 999 の範囲で、まだ優先順位として使用されていない値が有効で、リソースのカスケディングフェイルオーバーシーケンスにおけるサーバの優先順位を示します。数値が小さいほど優先順位は高くなります(数値 1 が最高の優先順位)。LifeKeeper のデフォルトでは、階層が作成されたサーバに「1」が割り当てられることに注意してください。優先順位は連続している必要はありませんが、特定のリソースについて 2 つのサーバに同じ優先順位を割り当てることはできません。
Template Priority	テンプレートの優先順位 を選択するか、入力してください。これはサーバで現在サービス中の DataKeeper 階層の優先順位です。1 ~ 999 の範囲で、まだ優先順位として使用されていない値が有効で、小さい数字ほど優先順位が高くなります(数値 1 が最高の優先順位)。拡張処理時に、別のシステムですでに使用中の優先順位をこの階層に対して指定することはできません。デフォルト値を推奨します。 注記: このフィールドは、階層を最初に拡張するときに表示されます。
テンプレートの優先順位	[Template Priority] を選択または入力します。これはサーバで現在 in service の DataKeeper 階層の優先順位です。優先順位は、1 ~ 999 の範囲で未使用の値が有効で、小さい数字ほど優先順位が高くなります(数字 1 が最高の優先順位に相当)。拡張処理時に、別のシステムですでに使用中の優先順位をこの階層に対して指定することはできません。デフォルト値を推奨します。 注記: このフィールドは階層を最初に拡張するときだけ表示されます。

フィールド	ヒント
Target Priority	Target の優先順位 を選択するか、入力してください。これは、他のサーバにある同等の階層に対する、新しく拡張する DataKeeper 階層の優先順位です。1 ~ 999 の範囲で、まだ優先順位として使用されていない値が有効で、リソースのカスケードリングフェイルオーバーシナリオにおけるサーバの優先順位を示します。数値が小さいほど優先順位は高くなります (1 は最高の優先順位を表す)。LifeKeeper のデフォルトでは、階層が作成されたサーバに「1」が割り当てられることに注意してください。優先順位は連続している必要はありませんが、特定のリソースについて 2 つのサーバに同じ優先順位を割り当てることはできません。

- Pre-Extend のチェックが正常に終了したというメッセージが表示されたら、**[Next]** をクリックしてください。

注記: 拡張する階層に応じて、拡張されるリソースタグ (一部編集不可) を示す一連の情報ボックスが表示されます。

- [Next]** をクリックして、**[Extend Resource Hierarchy]** の構成タスクを開始してください。

次のセクションには、別のサーバに DataKeeper リソースを拡張するために必要な手順を示します。

- pre-extend スクリプトが正常に実行されたというメッセージが表示されたら、以下の情報を指定するように要求されます。

フィールド	ヒント
Target Server	拡張先のサーバを入力するか、選択してください。
Switchback Type	[intelligent switchback] を指定する必要があります。これは、バックアップサーバにフェイルオーバーした後、管理者が手動で Multi-Site Cluster 階層のリソースをプライマリサーバにスイッチバックする必要があることを意味します。 注意: このリリースの SIOS DataKeeper for Linux は、DataKeeper リソースの自動スイッチバックをサポートしていません。さらに、自動スイッチバックの制限は、Multi-Site Cluster 階層を構成する LifeKeeper リソースにも適用されます。この制限の対象として、階層の上位あるいは下位にあるリソースも含まれます。
Mount Point	ターゲットサーバ上にあるファイルシステムのマウントポイント名を入力してください (DataKeeper リソースに関連する、LifeKeeper が保護するファイルシステムがない場合は、このダイアログは表示されません)。
Root Tag	ルートタグ を選択するか、入力してください。これは、ターゲットサーバ上にあるファイルシステムリソースインスタンスの一意の名前です (DataKeeper リソースに関連する、LifeKeeper が保護するファイルシステムがない場合は、このダイアログは表示されません)。

フィールド	ヒント
Target Disk or Partition	<p>複製するファイルシステムの配置先となる、ターゲットサーバ上のディスクまたはパーティションを選択してください。</p> <p>ドロップダウンボックスのディスクまたはパーティションのリストには、<u>以下のものを除いて</u>、使用できるすべてのディスクが表示されます。</p> <ul style="list-style-type: none"> • すでにマウント済みのもの • スワップディスクまたはスワップパーティション • LifeKeeper が保護するディスクまたはパーティション <p>ドロップダウンリストには、root (/)、boot (/boot)、/proc, floppy, cdrom などの特殊なディスクまたはパーティションも表示されません。</p> <p>注記: ターゲットのディスクまたはパーティションは、ソースのディスクまたはパーティション以上のサイズである必要があります。</p>
DataKeeper Resource Tag	<p>DataKeeper リソースタグの名前を選択するか、入力してください。</p>
Bitmap File	<p>インテントログの記録に使用するビットマップファイルの名前を選択するか入力してください。[None]を選択すると、インテントログは使用されず、すべての再同期が部分的ではなく全体の再同期になります。</p> <p>重要: ビットマップファイルは btrfs ファイルシステム上に置いてはいけません。データレプリケーションのビットマップファイルが btrfs ファイルシステム上に置かれると、LifeKeeper がミラーを構成しようとした時、"invalid argument" エラーの原因になります。ビットマップファイルのデフォルトの置き場所は、/opt/LifeKeeper の下です。このデフォルトの置き場所は、/opt/LifeKeeper が btrfs ファイルシステム上にある場合変更されます。</p>
Replication Path	<p>ターゲットサーバとクラスタ内の他の指定サーバとの間で複製に使用する、ローカルとリモートの IP アドレスのペアを選択してください。有効なパスおよび対応する IP アドレスは、このサーバのペアに対して指定した LifeKeeper コミュニケーションパスのセットから得られます。</p> <p>DataKeeper の特性により、プライベート(専用)ネットワークを使用することが強く推奨されます。DataKeeper リソースをすでに 1 台以上のターゲットサーバに拡張している場合、追加のサーバに対する拡張を実行すると、新しいターゲットサーバと既存のサーバとの組み合わせのそれぞれについて、繰り返し複製パスを指定するように要求されます。</p>
Replication Type	<p>指定したサーバのペアについて使用する複製タイプとして、[synchronous] または [asynchronous] を選択してください。</p> <p>前述の [Replication Path] フィールドと同様に、DataKeeper リソースをすでに 1 台以上のターゲットサーバに拡張している場合、追加のサーバに対する拡張を実行すると、新しいターゲットサーバと既存のサーバとの組み合わせのそれぞれについて、繰り返し複製タイプを指定するように要求されます。</p>

2. **[Next]** をクリックして次に進んでください。拡張が実行中であることを確認する情報ボックスが表示されません。
3. **[Finish]** をクリックして、DataKeeper リソースインスタンスが正常に拡張されたことを確認してください。
4. **[Done]** をクリックして、**[Extend Resources Hierarchy]** メニューを終了してください。

リストアおよびリカバリの設定

異なるサブネットで構成されたマルチサイト環境においてIPリソースを使用する場合、IPリソースの **[Modify Restore and Recover]** で**[Disable]**を選択する必要があります。詳細は[\[IP 構成の確認および編集\]](#)を参照してください。

必ずすべてのサーバで手動スイッチオーバーを実行して、新しいインスタンスの機能をテストしてください。詳細については、[リソース階層のテスト](#)を参照してください。ディザスタリカバリノードへの拡張が完了している場合、この時点で、SIOS DataKeeper がソースからターゲットのディスクまたはパーティションにデータの再同期を開始していません。LifeKeeper の GUI では、ターゲットサーバにある DataKeeper リソースのステータスは「Resyncing」(再同期中)になります。再同期が完了すると、ステータスは「Target」になります。これは通常のスタンバイ状態です。

再同期中、DataKeeper リソースおよびそれに依存するリソースはフェイルオーバーできません。これは、データの破損を防止するためです。

まだ実行していない場合は、必ず confirm failover フラグをセットしてください。この手順の詳細については、[\[Confirm Failover\]](#) と [\[Block Resource Failover\]](#) の設定のセクションを参照してください。

Multi-Site Cluster 環境へのマイグレーション

SIOS Multi-Site Migrate 機能が、SIOS Protection Suite for Linux Multi-Site Cluster 製品に装備されています。この追加機能を使用すると、管理者は既存の SIOS Linux LifeKeeper 環境を Multi-Site Cluster 環境に移行できます。移行手順により、階層のダウンタイムを最小に抑えて、選択した共有ファイルシステムのリソースを安全に移行して複製できます。

既存のファイルシステムから Multi-Site リソースを作成するときの重要な考慮事項をいくつか示します。

- Multi-Site の移行手順では、作成プロセスでファイルシステムをアンマウントし、NetRAID デバイスに再マウントします。
- リソースの作成手順中は、このファイルシステムに依存するアプリケーションをすべて、停止する必要があります。この操作は移行手順が処理するので、管理者からの操作は不要です。
- **NAS**(scsi/netstorage)、**DRBD**(scsi/drbd)、**SDR**(scsi/netraid)、および **Multi-Site Cluster** リソース (scsi/disrec) のリソースタイプを含む階層は、Multi-Site の移行機能を使用して移行することはできません。

要件

マイグレーションを実行する前に、お使いのシステムが本書の[インストールと設定](#)セクションに記載されている要件を満たすことを確認してください。

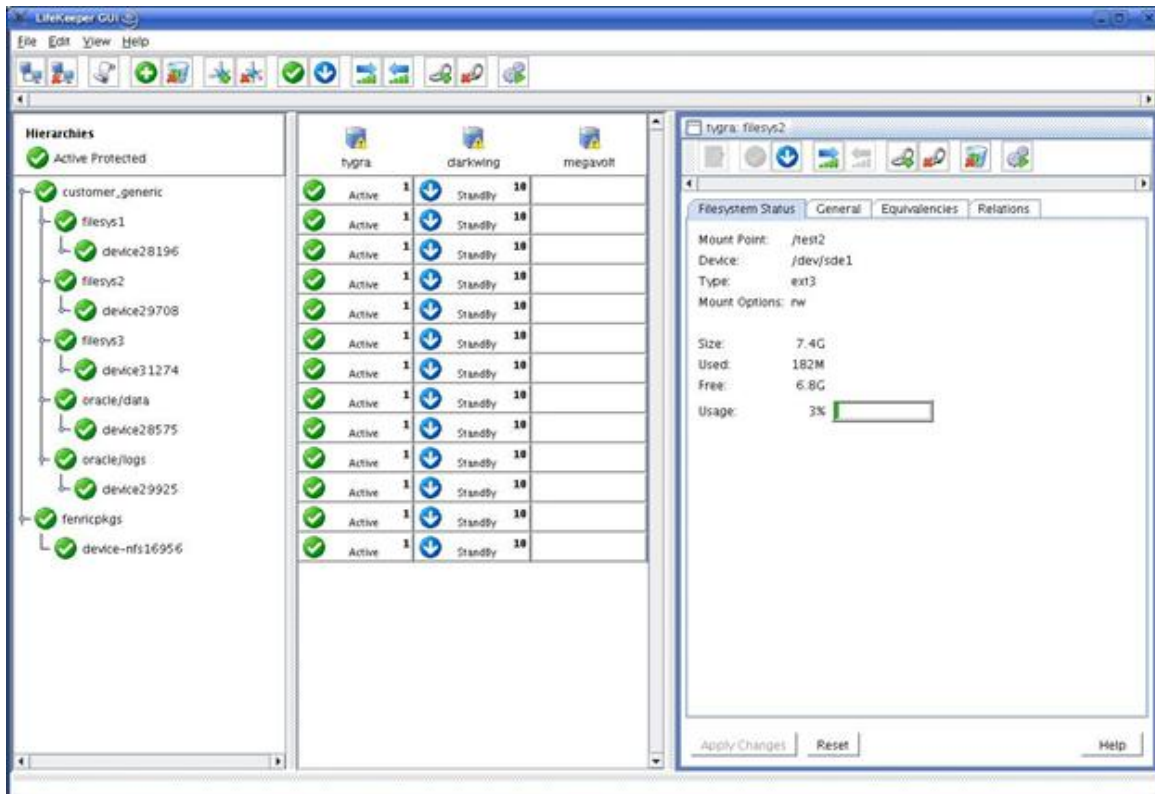
始める前に

この機能はストレージデバイスを共有する2つのサーバを有する構成のためにあります。1台のサーバはプライマリノードと想定され、プライマリサイトにあります。3台目のサーバはリモートのディザスタリカバリサイトにあります。

SIOS Protection Suite for Linux Multi-Site Cluster をプライマリとその他の共有ストレージノードにインストールした後は、**マイグレーション**機能を活用するために必要な追加のインストールや設定は不要です。

始める前に

以下の画像に、移行を開始する前のファイルシステムのリソース階層を示します。



マイグレーションの実行

Multi-Site Migrate を構成して実行するには3とおりの方法があります。以下の操作ができます。

- LifeKeeper GUI のツールバーから **[Migrate]** アイコン



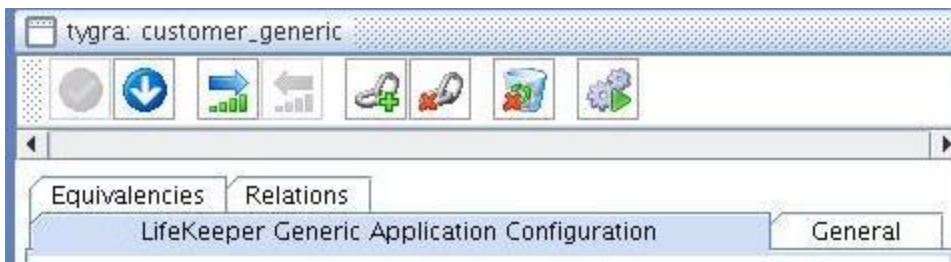
を選択し、移行するリソースを選択しま

す。

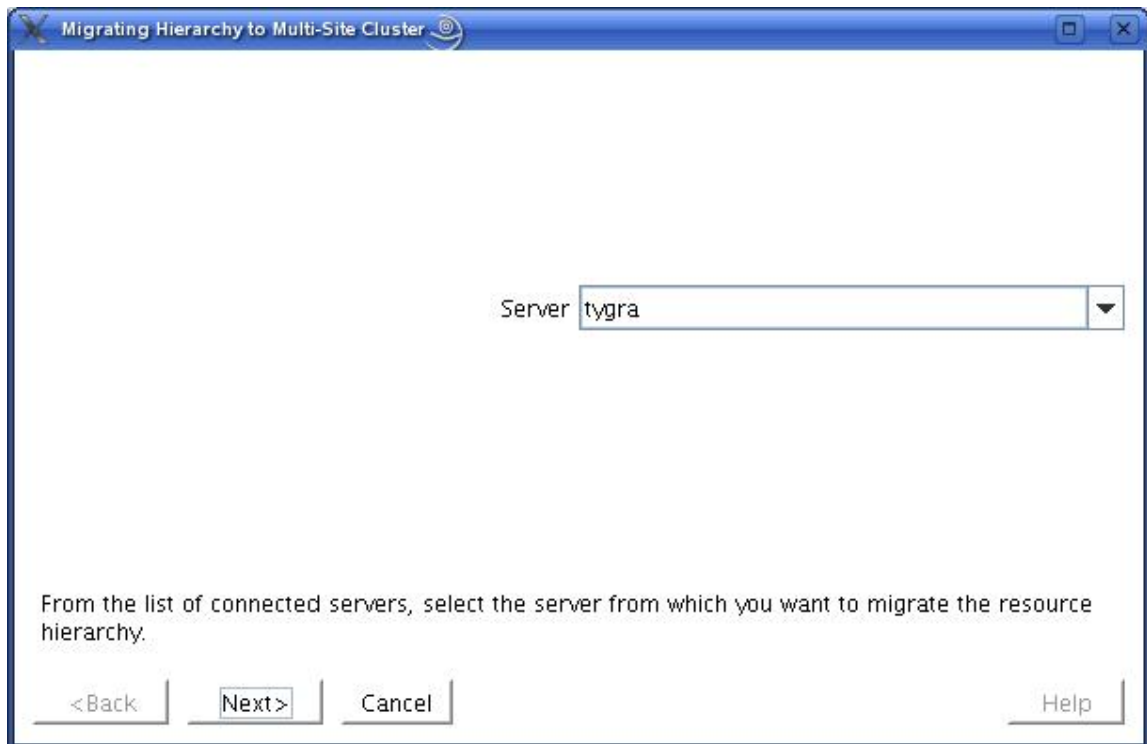
- ファイルシステムリソースを右クリックして、**[Migrate Hierarchy to Multi-Site Cluster]** メニューオプションを選択します。



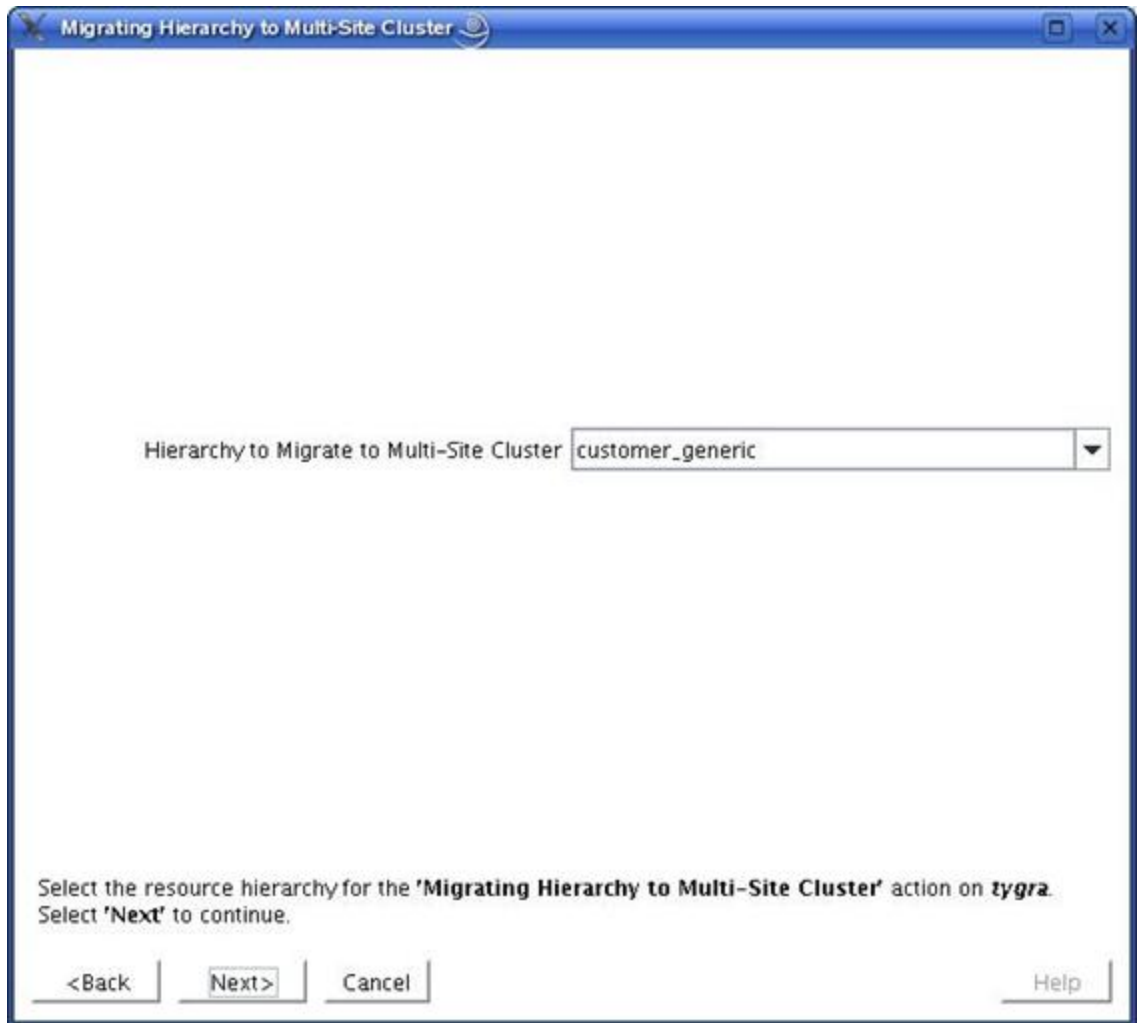
- ファイルシステムリソースを選択し、**[Properties Panel]** ツールバーの **[Migration]** アイコンを選択します。



グローバルツールバーのアイコンから移行を開始した場合、以下のダイアログボックスが表示されます。

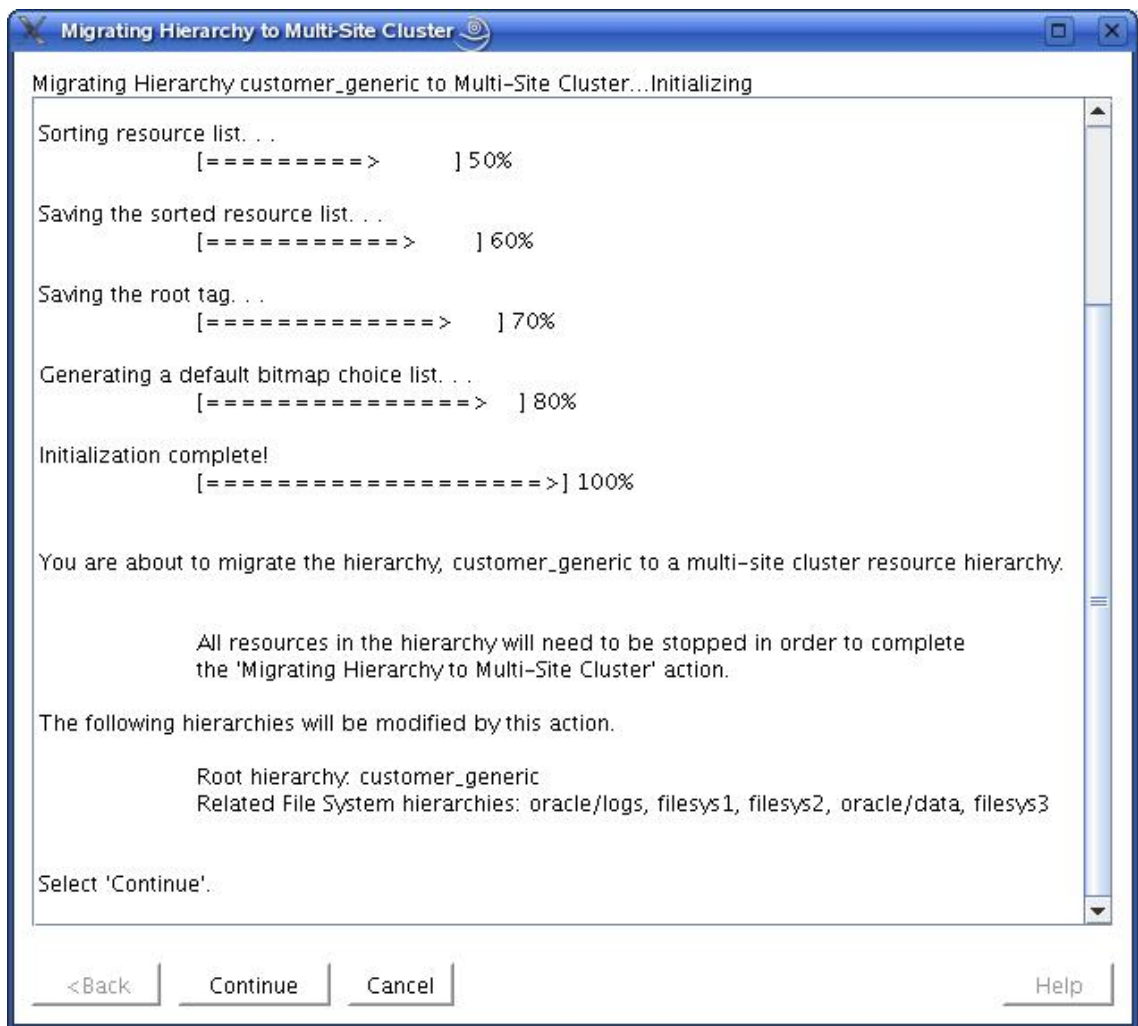


1. 移行する階層が存在する、サービス中のサーバを選択してください。[Next]をクリックしてください。

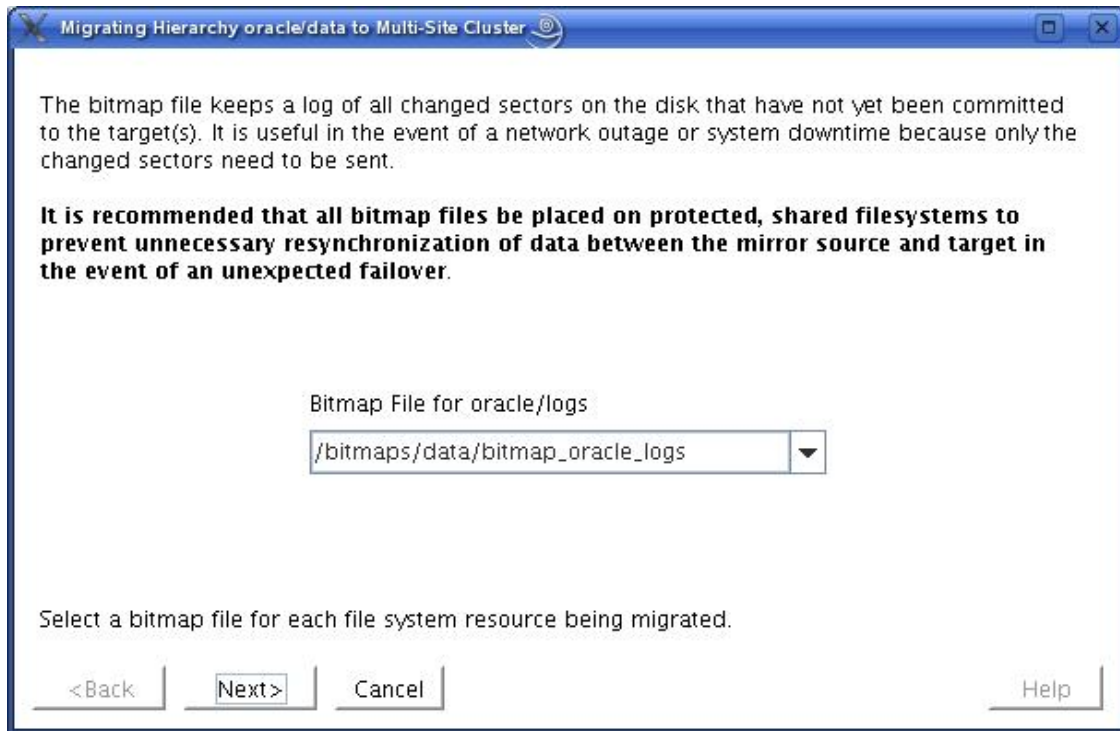


2. 移行する **root 階層タグ** を選択し、**[Next]** をクリックしてください。root タグは、ファイルシステムにすることも、他のアプリケーションリソースにすることもできます。選択したタグ(ファイルシステム以外のリソースの場合)には、ファイルシステムに依存するリソースが含まれている必要があります。

LifeKeeper の GUI のウィンドウでファイルシステムを選択し、ポップアップウィンドウから **[Migrate Hierarchy to Multi-Site Cluster]** を選択するか、**[Properties Panel Migrate]** アイコンの **[Migrate]** アイコンを選択した場合、以下の初期化画面が表示されます。

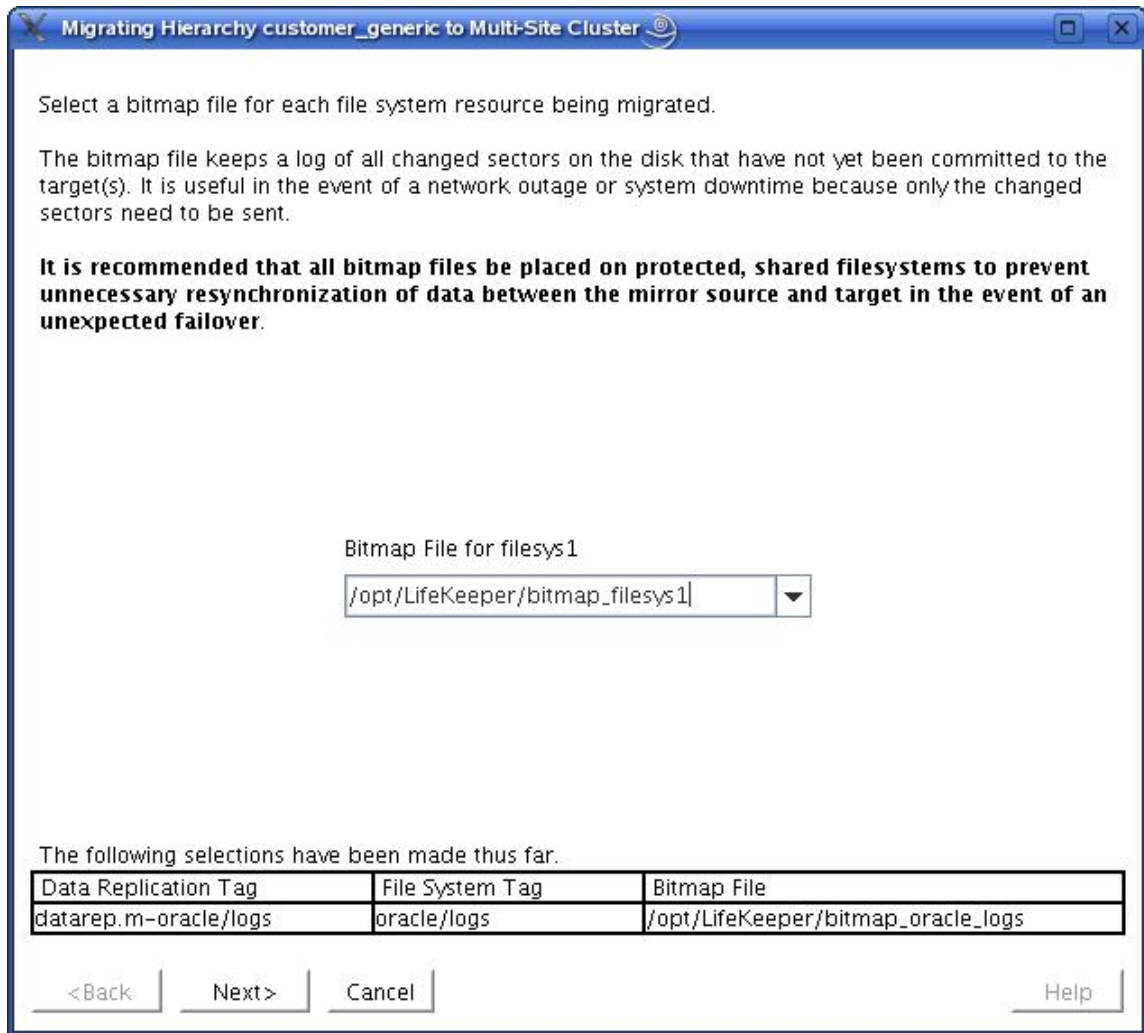


3. **[Continue]** ボタンが有効になったらクリックしてください。以下のビットマップダイアログが表示されます。



4. 移行するファイルシステムのビットマップファイルを選択してください。[Next] をクリックしてください。

重要:[Next] をクリックした後は、このファイルシステムのビットマップファイルの選択を変更できなくなります。

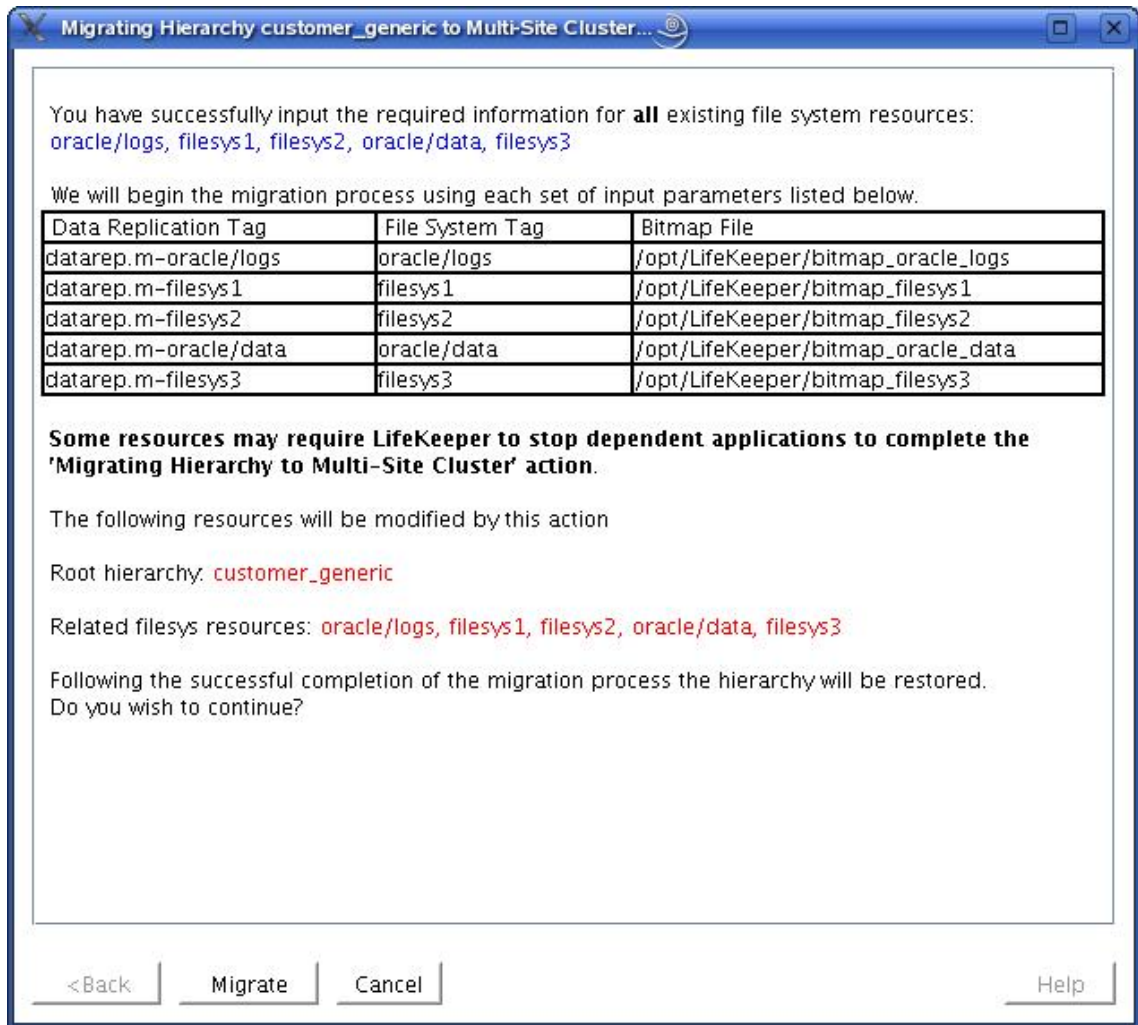


5. 階層内で移行する2番目のファイルシステムのビットマップファイルを選択してください。前のダイアログボックスで1番目のビットマップファイルを選択した後、追加のファイルシステムタグが表示されるので、それらの各タグについて一意のビットマップファイルを入力できます。

注記: 移行するファイルシステムが1つのみの場合は、この画面は表示されません。また、移行するファイルシステムが2つ以上の場合、この画面に似た複数の画面が表示されます。

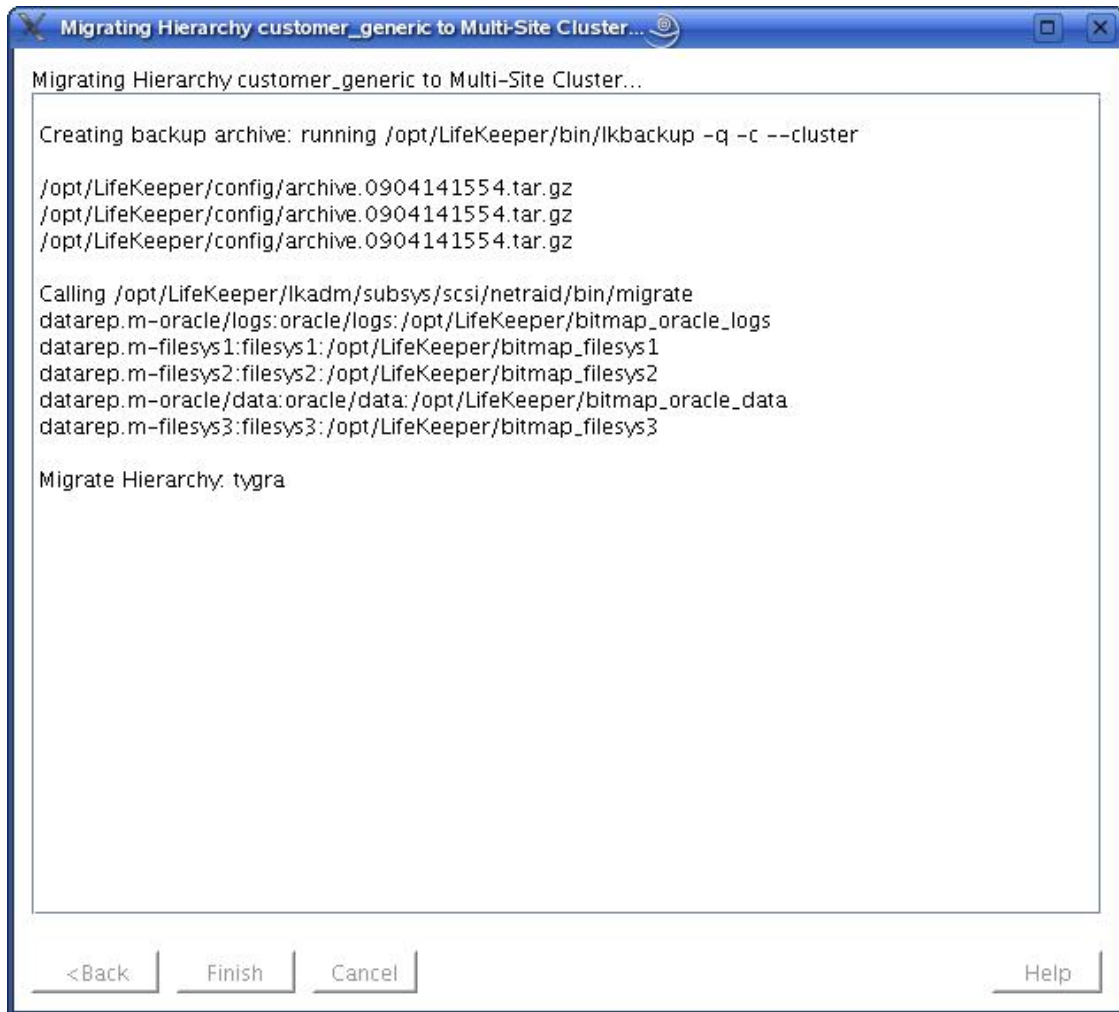
6. **[Next]** をクリックしてください。以下のような概要画面が表示されます。

マイグレーションの実行



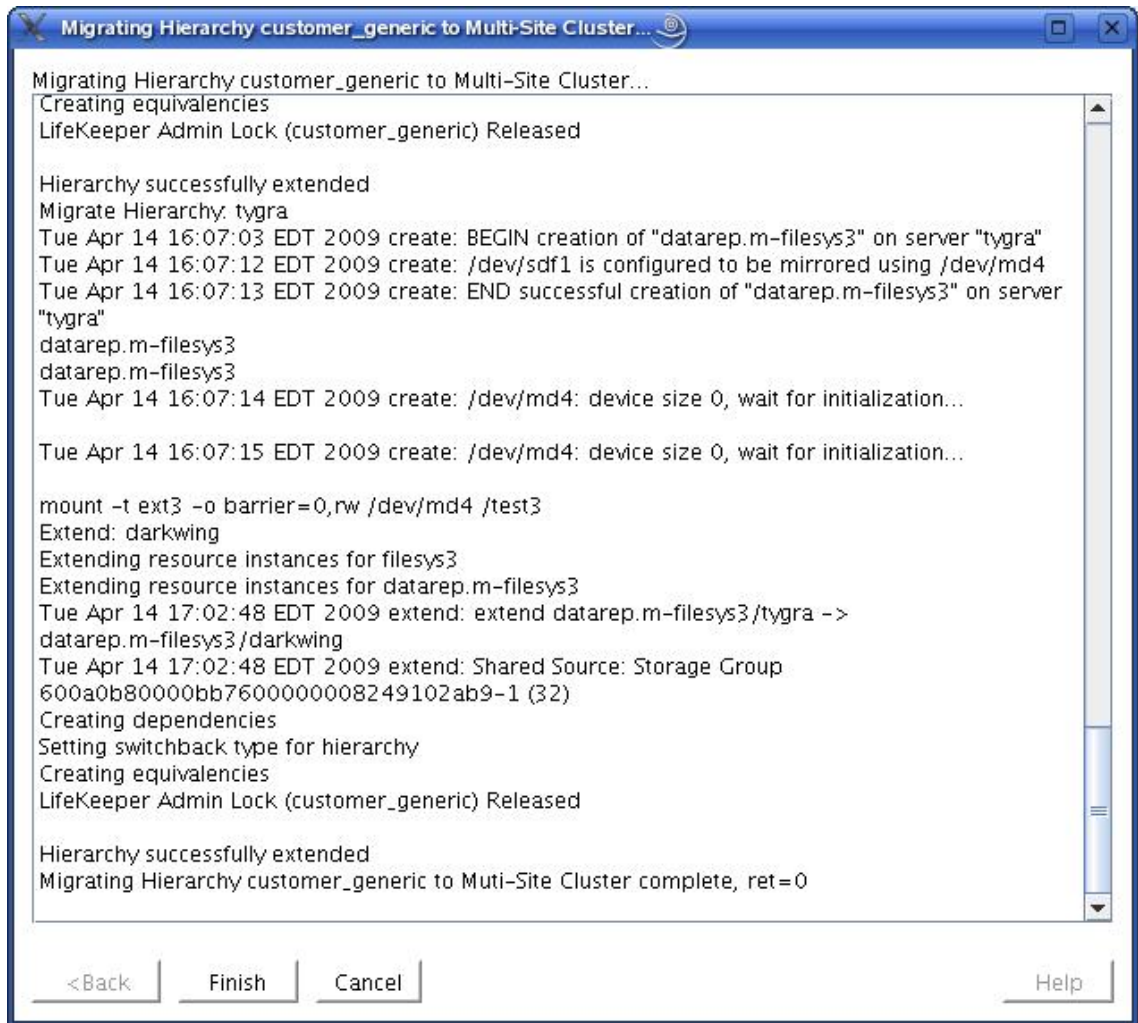
7. この概要画面には、移行手順で送信したすべての構成情報が表示されます。**[Migrate]** をクリックすると、以下の画面が表示されます。

マイグレーションの実行



8. 移行ステータスがこのウィンドウに表示されます。**[Finish]** ボタンが有効になったらクリックしてください。

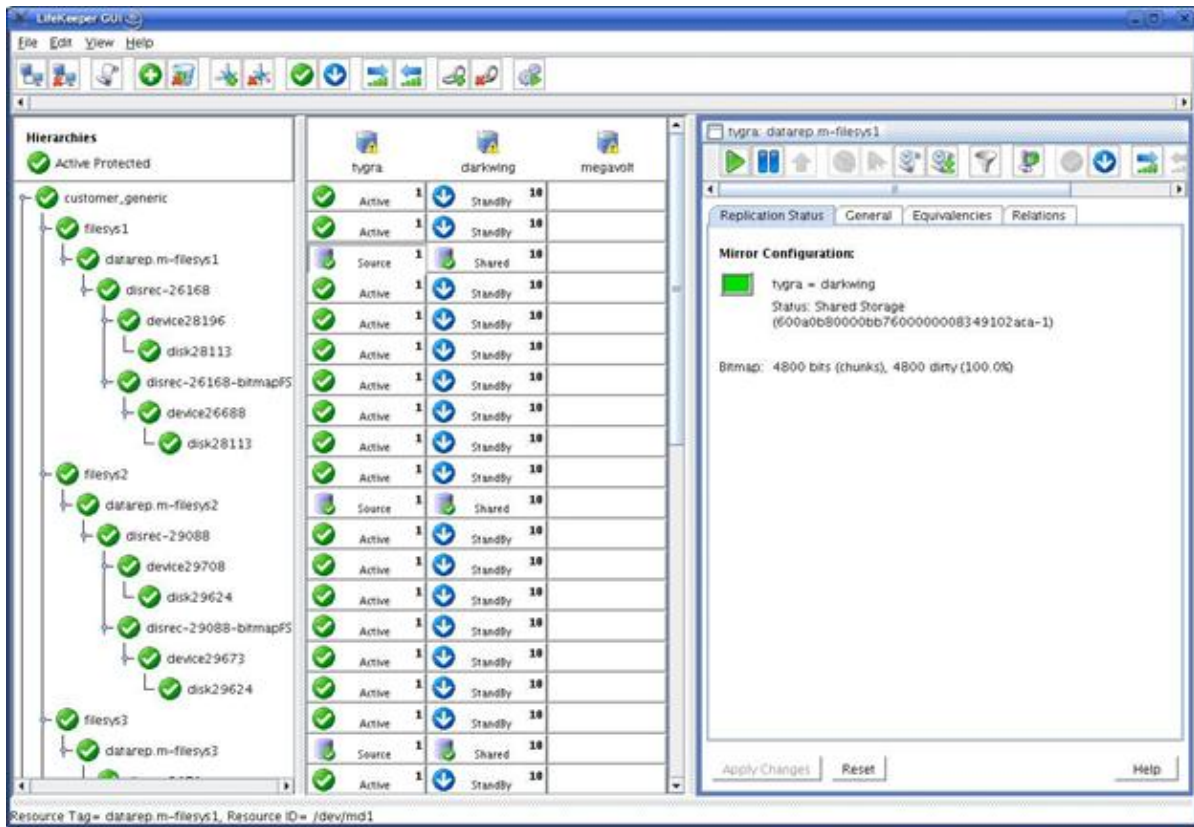
マイグレーションの正常な完了



マイグレーションの正常な完了

以下の画像に、Multi-Site のマイグレーションが完了した後のファイルシステムリソース階層の例を示します。これで、階層を非共有ノード (megavolt) に拡張できます。

マイグレーションの正常な完了



トラブルシューティング

このセクションでは、DataKeeper for Linux の使用時に遭遇する可能性がある問題に関する情報を示します。必要に応じて、エラーの原因およびエラー状態を解消するために必要な処置についても説明しています。

DataKeeper for Linux に固有のメッセージについては、DataKeeper のメッセージカタログを参照してください。他の SPS コンポーネントからメッセージが送出されることもあります。その場合は、総合メッセージカタログを参照してください。これらのメッセージカタログは両方とも、弊社のテクニカルドキュメンテーションサイトの「エラーコードの検索」から見つけることができます。これらのメッセージカタログには、SIOS Protection Suite for Linux の使用中に遭遇される可能性のあるすべてのエラーコード（操作、管理、および GUI に関するものを含む）の一覧があります。また、エラーコードの原因に関する追加の説明や、問題解決のために必要な処置についても、必要に応じて記載されています。この一覧から、受信したエラーコードを検索できます。また、該当する SPS コンポーネントの個別のメッセージカタログに直接アクセスすることもできます。

以下の表に、予測される問題と推奨される処置を示します。

症状	推奨される処置
DataKeeper リソースを削除した後に NetRAID デバイスが削除されません。	NetRAID デバイスがマウントされている場合、DataKeeper リソースを削除しても NetRAID デバイスは削除されません。以下のコマンドを使用して、手動でデバイスをアンマウントして削除することができます。 <code>mdadm -S <md_device> (<md_device> を調べるには <code>cat /proc/mdstat</code>)</code>
インストール/HADR rpm の失敗	これらのファイルを手動でインストールするための詳細手順については、 インストール セクションを参照してください。
フェイルオーバー中のエラー	デバイスのステータスを確認してください。再同期が進行中の場合、フェイルオーバーは実行できません。

症状	推奨される処置
<p>プライマリサーバに障害が発生すると、セカンダリサーバの DataKeeper リソースが ISP になります。ただし、プライマリサーバが再起動すると、両方のサーバで DataKeeper リソースが OSF になります。</p>	<p>DataKeeper リソース階層の作成時に選択した「スイッチバックタイプ」を確認してください。このリリースでは、DataKeeper リソースの自動スイッチバックはサポートされていません。リソースプロパティのウィンドウで、スイッチバックタイプを [Intelligent] に変更できます。</p>
<p>両方のサーバが動作不能になってからプライマリサーバが再起動したときに、リソースを ISP にすることができない。</p>	<p>セカンダリサーバよりも前にプライマリサーバが動作可能になった場合、DataKeeper リソースを強制的にオンラインにすることができます。このためには、リソースプロパティのダイアログを開き、[Replication Status] タブ、[Actions] ボタンを順にクリックし、次に [Force Mirror Online] を選択してください。[Continue] をクリックして確認してから、[Finish] をクリックしてください。</p>
<p>現在マウントしている NFS ファイルシステムに DataKeeper 階層を作成するときのエラー</p>	<p>現在 NFS がエクスポートしたファイルシステムに、DataKeeper 階層を作成しようとしています。エクスポートする前に、このファイルシステムを複製する必要があります。</p>
<p>DataKeeper の GUI のウィザードに、新しく作成したパーティションがリストされない。</p>	<p>Linux OS は、システムを次回再起動するまで、新しく作成したパーティションを認識しないことがあります。新しく作成したパーティションのエントリを調べるには、<code>/proc/partitions</code> ファイルを表示してください。新しく作成したパーティションがこのファイルに表示されない場合、システムを再起動する必要があります。</p>

症状	推奨される処置
<p>プライマリとバックアップの両方のサーバで、リソースが緑 (ISP) で表示される。</p>	<p>これは、「制御分離」のシナリオで、一時的な通信障害により発生することがあります。通信の再開後、両方のシステムがそれぞれ、それ自体をプライマリと見なします。</p> <p>いずれのシステムが最終のプライマリシステムであったかが不明なので、DataKeeper はデータを再同期しません。手動操作が必要です。</p> <p>ビットマップを使用しない場合：</p> <p>最終のバックアップであったサーバを特定し、そのサーバのリソースをサービス休止にする必要があります。その後、DataKeeper が全体の再同期を実行します。</p> <p>ビットマップを使用している場合 (2.6.18 以前のカーネル)：</p> <p>元のバックアップノードから始めて、両方のリソースをサービス休止にする必要があります。次に、以下のコマンドを実行して、プライマリノードのビットマップをダーティに設定する必要があります。\$LKROOT/lkadm/subsys/scsi/netraid/bin/bitmap -d /opt/LifeKeeper/bitmap_filesys</p> <p>(<code>/opt/LifeKeeper/bitmap_filesys</code> / ビットマップファイルの名前)。これにより、リソースがサービス中になると、全体の再同期が強制実行されます。次に、プライマリノードでリソースを in service にします。全体の再同期が開始されます。</p> <p>ビットマップを使用する場合 (2.6.19 以降のカーネル、Red Hat Enterprise Linux 5.4 の 2.6.18-164 以降のカーネル、または Red Hat 5.4 以降のサポートする派生カーネル)：</p> <p>最終のバックアップであったサーバを特定し、そのサーバのリソースをサービス休止にする必要があります。その後、DataKeeper が部分的な再同期を実行します。</p>
<p>Core - 言語環境の影響</p>	<p>LifeKeeper の一部のスクリプトは Linux のシステムユーティリティの出力を解析し、情報を抽出するために特定のパターンに依存します。英語圏以外のロケールで一部のコマンドが実行されている場合、予測されたパターンは変更され、LifeKeeper スクリプトは必要な情報の取得に失敗します。このため、<code>/etc/default/LifeKeeper</code> では、言語環境変数 <code>LC_MESSAGES</code> が POSIX「C」のロケール (<code>LC_MESSAGES=C</code>) に設定されています。言語を英語にして Linux をインストールする必要はありません (インストールメディアで使用できる任意の言語を選択可能)。<code>/etc/default/LifeKeeper</code> の <code>LC_MESSAGES</code> の設定は LifeKeeper にのみ影響します。<code>/etc/default/LifeKeeper</code> の <code>LC_MESSAGES</code> の値を変更する場合は、LifeKeeper の動作に悪影響を及ぼす可能性があることに注意してください。悪影響は、メッセージカタログがさまざまな言語とユーティリティに対応してインストールされているかどうか、および LifeKeeper が予期していないテキスト出力をそれらが生成するかどうかによって左右されます。</p>
<p>GUI - GUI の終了後に Web ブラウザから再接続したときに、GUI のログインプロンプトが再表示されないことがある。</p>	<p>GUI アプレットを終了するか切断してから、同じ Web ブラウザのセッションから再接続しようとすると、ログインプロンプトが表示されないことがあります。</p> <p>回避策： Web ブラウザを閉じ、Web ブラウザを開き直してからサーバに接続します。Firefox ブラウザを使用している場合は、Firefox のウィンドウをすべて閉じてから、開き直します。</p>

症状	推奨される処置
GUI - RHEL5 の kGUiapp が未サポートのテーマエラーをレポートする。	<p>GUI アプリケーションクライアントの開始時に、以下のコンソールメッセージが表示されることがあります。</p> <pre data-bbox="406 373 1347 436">/usr/share/themes/Clearlooks/gtk-2.0/gtkrc:60:Engine "clearlooks" is unsupported, ignoring</pre> <p>このメッセージは、RHEL 5 および FC6 Java プラットフォームの表示方式からのもので、GUI クライアントの動作に悪影響は及ぼしません。</p>
DataKeeper の Create Resource が失敗します	<p>特定の環境 (例: IDE ディスクエミュレーションを使用した仮想環境、HP CCISS ストレージを使用したサーバ、またはSSD) で DataKeeper を使用する場合、ミラーを作成すると以下のエラーが発生することがあります。</p> <pre data-bbox="406 674 1347 737">ERROR 104052: Cannot get the hardware ID of the device "dev/hda3"</pre> <p>これは、LifeKeeper が問題のディスクを認識せず、一意の ID を取得してそのデバイスに関連付けることができないためです。</p> <p>回避策: DEVNAME device_pattern ファイルに、ディスクのパターンを追加します。次に例を示します。</p> <pre data-bbox="406 940 1331 1203"># cat /opt/LifeKeeper/subsys/scsi/resources/DEVNAME/device_pattern /dev/hda* /dev/fio* (Fusion IO SSD) /dev/hio* (PCI SSD)</pre>
マルチサイトクラスタ構成で、一時停止、再開を行った場合、全同期が発生することがある。	<p>ビットマップファイルの場所が間違っている可能性があります。ビットマップファイルは、マルチサイトクラスタでローカルノード間の共有ファイルシステム上にある必要があります。この共有ファイルシステムは、レプリケーションに使用される共有ファイルシステムと別でなければなりません。一度 DataKeeper リソースを削除して、正しいビットマップファイルの場所を使用してリソースを再作成してください。</p>

Recovery Kit

SIOS Protection Suite の全リカバリキットのリストとその管理ガイドについては、SIOS テクニカルドキュメンテーション (<http://jpdocs.us.sios.com>) の Linux Recovery Kit 用ドキュメンテーションを参照してください。

A**API 134****B****Bitmap File 292****Block Resource Failover 40****btrfs Filesystem 292****C****Confirm Failover 37****CONFIRM_SO**

リザベーションの無効化 114

Core 4**F****File Systems 5****G****Generic Applications 5****GUI**

LifeKeeper サーバでの実行 173

ソフトウェアパッケージ 153

リモートシステムでの実行 170

開始 166

概要 163

終了 175

設定 164

停止 166

GUI からのミラーの管理 300

I

In Service 195

INTERFACELIST 235

IP Addresses 5

J

Java

セキュリティポリシー 168

プラグイン 170

L

LifeKeeper Communications Manager (LCM) 204

ステータスの情報 205

警報とリカバリ

LifeKeeper の警報 インターフェース 205

LifeKeeper イベントメール通知

設定 35

LifeKeeper のローカルリカバリ動作と制御のインターフェース (LRACI) 5

LifeKeeper の起動 175

LifeKeeper の削除 211

LifeKeeper の自動再起動の無効化 177

LifeKeeper の自動再起動の有効化 176

LifeKeeper の停止 176

LifeKeeper 設定データベース (LCD) 196

/opt/LifeKeeper のLCD のディレクトリ構造 203

コマンド

LCD インターフェース (LCDI) 197

ディレクトリ構造 201

フラグ 201

リソースタイプ 201

リソースのサブディレクトリ 202

設定データ 200

lkbbackup

SDR による 241

破損したイクイバレンシ 234

lkpolicy ツール 131

M

Multi-Site Cluster 313

ファイルシステム

既存の複製 319

新規の複製 317

リストアおよびリカバリの設定 329

リソース階層

ディザスタリカバリシステムへの拡張 325

作成 316

移行

実行 330

正常な完了 339

概要 313

制限 315

設定する際の考慮事項 314

要件 329

N

N-Way リカバリ 135

Nested File System 235

O

Out of Service 196

Q

Quorum/Witness 116

- quorum モード 118
- Quorum を喪失した (多数派ではなくなった) ときのアクション 119
- witness モード 119
- インストールと設定 117
- リザベーションの無効化 113
- 共有 Witness 120
- 設定可能なコンポーネント 117

R

RAW I/O 5

S

SNMP によるイベント転送 29

- SNMP のトラブルシューティング 32
- 概要 29
- 設定 31

STONITH

- リザベーションの無効化 113

T

TTY 接続 28

ア

アクティブ / スタンバイ 9

アクティブ / アクティブ 8

イ

イクイバレンス情報 13

イベントメール通知 33

- トラブルシューティング 36

概要 29

インテリジェントスイッチバック 10

ウ

ウォッチドッグ

リザベーションの無効化 114

エ

エラーの検出 135

カ

カスタム証明書 48

コ

コマンドライン

ミラーステータスの監視 305

ミラー管理 302

コミュニケーションパス

ハートビート 6

ファイアウォール 212

作成 136

削除 138

サ

サーバグループ 6

サーバのプロパティ

フェイルオーバー 138

表示 182

サーバの障害 306

サーバプロパティ

編集 136

ス

ステータスの表 174

ステータス表示

簡略 20

詳細 15

ダ

ダイアログ

Cluster Connect 188

Cluster Disconnect 189

Resource Properties 189

Server Properties 191

タ

タグ名

リソース 264

有効な文字 264

ツ

ツールバー 159

GUI 159

サーバのコンテキスト 162

リソースのコンテキスト 161

デ

データ複製パス 278

テ

テクニカルサポート 2

テクニカルノート 216

ト

トラブルシューティング 223, 341

GUI

 トラブルシューティング 256

コミュニケーションパス 260

既知の問題 227

制限 227

不完全なリソースの作成 261

不完全なリソースの優先順位の変更 261

ネ

ネストされたファイルシステム 235

ネットワーク帯域幅

 変化率の測定 279

 要件の特定 279

ハ

ハードウェア 6

は

はじめに

 ミラーリング 267

 仕組み 269

フ

ファイアウォール

 ファイアウォールを使用した状態での LifeKeeper の実行 212

 ファイアウォール経由での LifeKeeper GUI の実行 214

フェイルオーバーのシナリオ 273

フェンシング

 I/O フェンシング表 114

 概要 111

代替方式 113

ブ

ブラウザのセキュリティパラメータ 173

プ

プロパティパネル 174

マ

マルチサイトクラスタ

リソース階層

拡張 323

始める前に 330

ミ

ミラーステータス

コマンドラインからの監視 305

ミラーのステータス

表示 299

ミラーを強制的にオンラインにする 301

ミラー管理

コマンドライン 302

メ

メッセージバー 175

メニュー 154

[Edit] メニュー - [Resource] 157

[Edit] メニュー - [Server] 157

File 156

Help 159

View 158

サーバのコンテキスト 155

リソースのコンテキスト 154

リ

リカバリ

Out-of-Service 階層 264

サーバ障害 263

フェイルオーバー後 210

手動リカバリ時のパニック 264

停止できないプロセス 264

リザベーション

SCSI 112

無効化 113

リソースタイプ 11

リソースのプロパティ 144

リソースの状態 12

リソースの優先順位 144

リソースポリシー管理 129

リソース依存関係

作成 149

削除 150

リソース階層 11

In Service 297

Out of Service 297

ツリーの折り畳み 187

ツリーの展開 187

テスト 297

メンテナンス 210

階層の関係 13

拡張 146, 293

Generic Application 147

Raw デバイス 148

ファイルシステム 147
拡張解除 148, 296
作成 140, 291
 Generic Application 142
 Raw デバイス 143
 ファイルシステム 141
削除 151, 296
情報 14
転送 215
例 15

圧

圧縮レベル 302

一

一時停止と再開 302

管

管理 135

共

共有データリソース 6

共有通信 7

健

健全性の監視 208

再

再同期 306

 全体の回避 307

自

自動スイッチバック 10

手

手動フェイルオーバー確認 37

出

出力パネル 175

障

障害検出とリカバリ 21

IP ローカルリカバリ 21

サーバの障害リカバリのシナリオ 25

リソースのエラーリカバリのシナリオ 23

切

切断 181

接

接続

サーバをクラスタに 180

設

設定 27

アプリケーション 52

ストレージとアダプタ 53

データレプリケーション 51

ネットワーク 51

ネットワークとLifeKeeper 278

概念 6

手順 27

全般 278

値 206

任意の作業 36

同**同期ミラーリング 268****認****認証情報 133****非****非同期ミラーリング 268****表****表示**

サーバのステータス 181

サーバのプロパティ 182

サーバのログファイル 182

メッセージ履歴 186

リソースのステータス 183

リソースのタグとID 183

リソースのプロパティ 185

接続サーバ 181

表示オプション 186**変****変化率 279****保****保護対象のリソース 3****要****要件**

DataKeeper 51

Quorum/Witness パッケージ 116

STONITH 124

ソフトウェア 277

ハードウェア 277

ファイアウォール 212

