



HP Serviceguardによる Micro Focus Enterprise Server クラスタ化の検証報告

日本ヒューレット・パッカード株式会社 MFA ビジネス推進部 マイクロフォーカス株式会社 技術部





このドキュメントの内容は予告なしに変更される場合があります。このドキュメントの 内容の保証や、商品性又は特定目的への適合性の保証や条件を含め明示的又は暗黙的な 保証や条件は一切無いものとします。マイクロフォーカス株式会社および日本ヒューレ ット・パッカード株式会社は、このドキュメントについていかなる責任も負いません。 また、このドキュメントによって直接又は間接にいかなる契約上の義務も負うものでは ありません。

このドキュメントを形式、手段(電子的又は機械的)、目的に関係なく、マイクロフォ ーカス株式会社および日本ヒューレット・パッカード株式会社の書面による事前の承諾 なく、複製又は転載することはできません。

Copyright© 2006 Micro Focus. All Rights Reserved.

Micro Focus は Micro Focus 社の登録商標、Enterprise Server, Server Express は同 社の商標です。 Serviceguardは、米国Hewlett-Packard Companyの商標です。 その他記載の会社名、製品名は、各社の商標または登録商標です。





目次

はじめに・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 1
注意事項・・・・・・1
使用環境・・・・・・・・・・・・2
使用したハードウェア・・・・・ 2
使用したソフトウェア・・・・・ 2
概要・・・・・・・・・・・・・・
Enterprise Server とテストシステム ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
クラスタリング・・・・・ 4
環境設定・・・・・・・・・・・5
手順・・・・・・5
環境設定モデル・・・・・・5
1. 事前準備・・・・・・ 6
2. 製品のインストール・・・・・ 6
3. Serviceguardの設定・・・・・ 6
4. COBOL コンポーネントの準備・・・・・・・・・・・・・・・・・・・・・・・・ 10
5.Enterprise Server の設定・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 10
6. クライアントの作成・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・12
検証・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
テストパターン・・・・・ 13
パターン① 手動切り替え・・・・・ 14
パターン② データ LAN 両系(二重)障害・・・・・・・・・・・・・・・・ 16
パターン③ 業務障害・・・・・ 18
パターン④ 0S のみで待機・・・・・・・・・・・・・・・・・・・・・・・・20
パターン(5) MFDS 起動で待機・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・24
パターン⑥ ES サーバ起動で待機・・・・・・・・・・・・・・・・・・・・・・・28
パターン⑦ 別の業務を提供しながら待機・・・・・・・・・・・・・・・・・ 30
パターン(8) ディプロイパッケージの共有・・・・・・・・・・・・・・・・35
パターン(9) 構成リポジトリの共有・・・・・・・・・・・・・・・・・・・・・・・38
パターン⑪ ログファイルの共有・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
まとめ・・・・・・ 44
付録
サーバアブリケーションソースサンブル・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 45
クライアントアブリケーションソースサンブル・・・・・・・・・・・・46





はじめに

本書では、Micro Focus Enterprise Server (以下、Enterprise Server)で提供するアプ リケーションサービスを HP Serviceguard (以下、Serviceguard)により共有ディスクを 利用したクラスタ上で動作させた検証を報告します。

Enterprise Server は Windows、Unix、Linux 環境で COBOL サービスをトランザクショ ナルに管理・実行するアプリケーション サーバです。Enterprise Server は Micro Focus Net Express および Server Express COBOL コンパイラ ツール群の運用部分です。 Enterprise Server はメインフレームのトランザクション環境と同等の OLTP 機能を、 J2EE アプリケーションサーバと協調実行できるよう設計されています。ユーザーにとっ ては、ハイエンドのトランザクション システムで、機能およびパフォーマンスと同様に 重要なことは、その環境がきわめて可用性、信頼性、保守性に優れていることです。

Serviceguard は、豊富な実績をもつハイアベイラビリティ・ミドルウェアです。さまざ まなハードウェア/ソフトウェアの障害から基幹系アプリケーションを保護する高度な機 能を HP-UX 上で提供します。Serviceguard は、最大 16 個のノードを疎結合することで エンタープライズ・クラスタを構成し、LAN で接続されたクライアントに対して可用性の 高いアプリケーション・サービスを提供できます。

ServiceguardとEnterprise Serverを組み合わせ、如何に信頼性・可用性の高いCOBOLサービスシステムを提供していくか、その方法について検証します。

注意事項

本書に記載されている内容は、検証環境での評価システムでの情報であり、実際のシス テム環境では、それぞれ個別に評価をし、固有の設定が必要になります。本書に記載さ れている内容は、参考値であり、実際のシステムでの動作を保証するものではありませ ん。





使用環境

使用したハードウェア

(サーバ) HP9000 rp3440 2台 PA8900 1GHz/64MB dualcore CPU x 2 搭載 (OSとしては4CPUとして認識) 8GB DDR メモリ (4x26B) x 3 搭載 300GB-10K U320 SCSI Disk x 3 搭載

(ストレージ)
 HP EVA8000-A 2C1D Array
 1 台
 1466B 10Krpm 2G FC-AL HDD x 32搭載

使用したソフトウェア

(0S)

HP-UX 11i v2 (11.23)

(アプリケーションサーバ) Micro Focus Enterprise Server 4.0J Service Pack 2 for PA-RISC HP-UX

(クラスタリング) HP Serviceguard A. 11. 17.00

(言語)

Micro Focus Server Express 4.0J Service Pack 2 for PA-RISC HP-UX Java(TM) 2 SDK, Standard Edition 1.4.2_8





概要

Enterprise Server とテストシステム

今回のテストで Enterprise Server に実装するサンプルモデルを用意しました。



クライアント

サーバ(クラスタ)

Java クライアントから要求されたリクエストは Enterprise Server 上にディプロイされ たサービスの COBOL アプリケーションに引き渡されます。この COBOL アプリケーション は索引ファイルであるアプリケーションデータから連番を取得し、この連番をカウント アップして書き込みします。そして日時、プログラム名、実行ホスト名、連番を返しま す。Java クライアントはこれを表示して終了します。

以下に関連する Enterprise Server の構成要素について説明します。

(プロセス関連)

• MFDS	Enterprise Server の基本サービスの一つ。1ノードに1つだけ
	起動が必要
・ES サーバ	Enterprise Server の管理単位。1 つのノードに複数定義でき、
	複数のサービスを含むことができる。
・サービス	Enterprise Server で提供する COBOL サービスであり、ディプロ
	イの単位。

・COBOL アプリケーション

COBOL プログラムが提供する業務プロセス。

- ・MF 提供クラス クライアントから Enterprise Server に接続するために利用
- Java クライアント コマンドライン起動で Enterprise Server にリクエストを出し
 受け取った結果を表示する。

(ファイル関連)

・ES システム Enterprise Server のシステムコマンド、ライブラリなどのファ イルを含む。\$COBDIR 環境変数がポイントするディレクトリパ ス。共有可。

・ライセンスシステム

Enterprise Server のライセンス管理の製品のライセンスが投入 されるデータベースを保管する。このディレクトリをディスク コピーなどで移動・復元することはでない。共有不可。





- ・ログファイル Enterprise Server のログ。ES サーバ単位で管理され、再起動
 によって自動的に新しいファイルに書き換えられる。共有可。
- ・構成リポジトリ Enterprise Server の設定情報を格納したファイル群。共有可。
- ・ディプロイパッケージ car ファイル・ディ

car ファイル。ディプロイに必要なファイル群。実装する際のデ ィプロイパッケージは Interfase Mapping Tool Kit(IMTK)によ り COBOL ソースから容易に生成することが出来る。Server Express によってコンパイルされた実行形式のファイルを含む。 (ソースサンプルは巻末に記載)共有可。

・アプリケーションデータ

COBOL プログラムが利用するユーザファイル。共有可。

- ・Java クライアント Java プログラム。(ソースサンプルは巻末に記載)
- ・MF 提供クラス クライアントプログラムが利用する Enterprise Server 接続用の Java クラス。

クラスタリング



Serviceguard ではフェイルオーバーする一連の操作をパッケージの単位で管理します。

パッケージには、再配置可能 IP アドレス (仮想 IP アドレス) や共有ディスクパーティションの有効化、アプリケーションの起動・停止・監視の処理を設定します。 再配置可能 IP アドレスや共有ディスクパーティションはパッケージが立ち上がると自動的に有効化され停止すると無効化します。

MFDS や ES サーバなどの起動・停止はそれぞれのシェルにコマンドで処理を記述します。 本運用ではこの起動・停止のシェルには障害時の状態回復・状態保存を行なう処理など も組み込む必要があります。

またアプリケーションの監視は、障害とする事象が発生するまで監視を繰り返す業務監 視用のシェルをユーザーが準備しこれを組み込む事で行ないます。





環境設定

下図のモデルを想定した場合の環境の Serviceguard、MF 製品の設定例を記述します。

手順

- 1. 事前準備
- 2. 製品インストールと動作環境の設定
- 3. Serviceguard の設定
- 4. COBOL コンポーネントの準備
- 5. Enterprise Server の設定
- 6. クライアントの作成

環境設定モデル







1. 事前準備

各ノードに搭載されたNICの設定、共有ファイルの接続に関する事前の設定、その他の OS レベルの設定を済ませておきます。

2. 製品のインストール

(クラスタ環境)

各ノード個別に下記2つの製品をインストールします。

- Micro Focus Enterprise Server 4.0J Service Pack 2 for PA-RISC HP-UX
- HP Serviceguard A. 11. 17.00

(開発環境)

実装する COBOL コンポーネントの準備に COBOL・Java 開発環境が必要になります。このノード上もしくは別の同一 OS のマシン上に下記の製品をインストールします。

- Micro Focus Server Express 4.0J Service Pack 2 for PA-RISC HP-UX
- Java(TM) 2 SDK, Standard Edition 1.4.2_8
- j2ee. jar

(クライアント環境)

本検証ではクライアントは Java プログラムからリクエストしますので、クライアント に Java 環境が必要になります。本検証では

- Java(TM) 2 SDK, Standard Edition 1.4.2_8
- j2ee. jar

またあらかじめ各製品の動作に必要な環境変数等の設定をしておきます。

3. Serviceguard の設定

本検証では Serviceguard の設定は各種設定ファイルを作成し、コマンドにより反映しま した。ここで設定するのはクラスタ構成ファイル、パッケージ構成ファイル、パッケー ジ制御スクリプトです。

クラスタ構成ファイル

クラスタ各ノードとネットワーク構成を登録します。今回は2ノードの設定をしました。 (/etc/cmcluster/cmclconf.ascii)

パッケージ構成ファイル

- パッケージ定義を必要数準備します。
 - (/etc/cmcluster/mfpkg1/ESSRV01.conf)

(/etc/cmcluster/mfpkg2/ESSRV02.conf)

パッケージ制御スクリプト

このパッケージに再配置可能 IP アドレス、共有ディスク、フェイルオーバーするサービス、監視プロセスなどを登録します。

- (/etc/cmcluster/mfpkg1/ESSRV01.cntl)
- (/etc/cmcluster/mfpkg2/ESSRV02.cntl)

本検証ではフェイルオーバーサービスの起動停止、監視プロセスの立ち上げは後述の mf.sh で記述する事にします。





cntl ファイルは OS コマンドでコピー、conf ファイルは Serviceguard のコマンド (cmcheckconf、cmapplyconf)にてチェック/配布しました。

次に ES サーバを管理するコマンドを設定します。以下は代表的なコマンド群です。 mfds (Micro Focus Directory Server)は Enterprise Server の基本サービスであり、1 ノードに1つのみ起動できる必須サービスです。mfds は ES サーバの立ち上げを行なう 前に起動しておく必要があります。

処理	コマンド例
mfds の初期化処理	ps -ef grep mfds > \$\$tmp.txt 2> /dev/null
(必要に応じて)	tmp=`cat \$\$tmp.txt grep -v "grep -E " awk '{print \$2}'`
	if [".\$tmp" != "."] ; then
	kill −9 \$tmp
	fi
	rm \$\$tmp.txt
mfds の起動	mfds &
	Sleep 5
mfds の停止	ps -ef grep mfds > \$\$tmp.txt 2> /dev/null
	tmp=`cat \$\$tmp.txt grep -v "grep -E " awk '{print \$2}'`
	if [".\$tmp" != "."] ; then
	kill −9 \$tmp
	fi
	rm \$\$tmp.txt
ES サーバの初期化処理	para= <es サーバ名=""></es>
(必要に応じて)	ps -ef grep \$para grep -E
	"cassi cascd casjcp casmgr mfcs " > \$\$tmp.txt 2> /dev/null
	tmp=`cat \$\$tmp.txt grep -v "grep -E " awk '{print \$2}'`
	if [".\$tmp" != "."] ; then
	kill -9 \$tmp
	fi
	rm \$\$tmp.txt
ES サーバの起動	casstart /r <es サーバ名=""></es>
ES サーバの停止	casstop /r <es サーバ名=""></es>





```
(Enterprise Server 管理シェル例1:mf.sh)
   #!/usr/bin/sh
   mffunc=$1
   host_name=$2
   es_name=$3
   APPFILE01=/u01/APPFILE01
   export APPFILE01
   APPFILE02=/u02/APPFILE02
   COBMODE=32
   export COBMODE
   export APPFILE02
   COBDIR=/opt/mf/ES40SP2
   export COBDIR
   SHLIB_PATH=$COBDIR/lib
   export SHLIB_PATH
   PATH=$COBDIR/bin:$PATH
   export PATH
   case $mffunc in
   start)
   /\texttt{etc/cmcluster/mfpkg1/killmfds.sh} >> /\texttt{etc/cmcluster/mfpkg1/ESSRV01.cntl.log}
   /\texttt{etc/cmcluster/mfpkg1/killallcas.sh} ESSRV01 >> /\texttt{etc/cmcluster/mfpkg1/ESSRV01.cntl.log}
   mfds &
   /wait 5
   casstart /rESSRV01
   exit
   monitor)
   sleep 10
   mdump_func()
   mdump $host_name $es_name > $$tmp.txt 2> /dev/null
   web_srv_flg=0
   cat $$tmp.txt | while read line
   do
               # if mfds not started "SERVER_DOWN" found
               # actual text is 'm_ldap_bind: 0x51 SERVER_DOWN'
            tmp=`echo $line | fgrep "SERVER_DOWN"`
if [ ".$tmp" != "." ]; then
                          # echo $line
                     return 3
            fi
            tmp=`echo $line | fgrep "CN:"`
if [ ".$tmp" != "." ]; then
                     tmp=`echo $line | fgrep "Web Services"`
if [ ".$tmp" != "." ]; then
                              # echo $line
                              web_srv_flg=1
                     else
                              # echo "no " $line
                              web_srv_flg=0
                     fi
            fi
             if [ $web_srv_flg -eq 1 ] ; then
                     lisnr=`echo $line | fgrep "mfListenerStatus:"`
if [ ".$lisnr" != "." ] ; then
                               lisnr=`echo $line | fgrep "Started"`
if [ ".$lisnr" != "." ] ; then
                                        return O
                              else
                                        # echo $line
                                        return 1
                               fi
                     fi
            fi
   done
```

パッケージで管理する ES サーバの関連処理をシェル mf. sh として記述します。





```
if [ ".$host_name" = "." ] || [ ".$es_name" = "." ] ; then
          host_name="localhost"
          es_name=""
fi
while : ; do
        # echo "monitoring"
       mdump_func
                         res=$?
        rm $$tmp.txt
        if [ "$res" = "1" ]; then
               echo "**** MF Service Failed ****" >> /etc/cmcluster/mfpkg1/ESSRV01.cntl.log
                exit 0
        fi
        if [ "$res" = "3" ]; then
               echo "**** MF mfds Down ****" >> /etc/cmcluster/mfpkg1/ESSRV01.cntl.log
                exit O
        fi
        sleep 5
done
halt)
casstop /rESSRV01
sleep 5
/etc/cmcluster/mfpkg1/killmfds.sh >> /etc/cmcluster/mfpkg1/ESSRV01.cntl.log
exit
esac
```

start)や halt)で記述の部分はパッケージの起動時・停止時に必要なサービスの管理を 行ないます。本検証ではテストパターンによって処理の書き換えを行いました。

Monitor) で記述の部分はマイクロフォーカスが提供する mdump ユーティリティにて、 Enterprise Server のステータスを取得し、これを解析してサービスが正常に提供され ていることを監視する処理です。対象のサービスが起動状態でなくなるまで監視し続け ます。Serviceguard はこの監視処理が終了したことでアプリケーション障害と見なし、 フェイルオーバーを開始することになります。

また、万がーシステムが中途半端に停止した状態から再起動する場合に備え、各サービス起動の前に不要プロセスを kill する処理を組み込みます。

(Enterprise Server 管理シェル例 2: killmfds.sh)

(Enterprise Server 管理シェル例3: killallcas.sh)

```
para=$1
ps -ef | grep $para | grep -E "cassi|cascd|casjcp|casmgr|mfcs" > $$tmp.txt 2> /dev/null
cat $$tmp.txt
tmp=`cat $$tmp.txt | grep -v "grep -E "| awk ' [print $2]'`
if [ ".$tmp" != "." ] ; then
            echo "kill -9 " $tmp
            kill -9 $tmp
fi
rm $$tmp.txt
```





4. COBOL コンポーネントの準備

Enterprise Server に実装するコンポーネントの準備を行ないます。 作業は開発環境にて行ないます。本検証では Server Express が提供する Interface Mapping Tool Kit を使って一連の作業を簡略化しています。 初めに COBOL ソースの元にサービスに対するデフォルトのマッピングを作成します。サ ービス名は任意に命名します。 imtkmake -defmap src=cblapp01.cbl service=srv01 type=ejb

これで"srv01.xml"の名称でデフォルトのマッピング情報が生成されます。 デフォルトではこのサービスのパラメータはプログラム中のパラメータ名で全

て"io"(入出力)として生成されています。本サンプルでは出力パラメータのみである ため実態に合わせ全て"out"に手修正しました。

そして次のコマンドでディプロイパッケージを生成します。

imtkmake -generate service=srvO1 type=ejb

これで" srv01. car" ファイルが生成されます。

更にここで同じ COBOL ソースを cob コマンドでコンパイルし、実行形式ファイル(本検証 では gnt 形式を選択)を生成しておきます。

5. Enterprise Server の設定

まず/etc/services に次の 2 行を追加します。 mfcobol 86/tcp mfcobol 86/udp 次に mfds コマンドで Enterprise Server を起動します。 Web ブラウザから http://(IP アドレス):86 でアクセスすると Enterprise Server Administration 画面が開きます。ここで多くの操作を実行できる様になります。

初めにデフォルトで生成されている ES サーバ"ESDEMO"を削除します。 次に新規に必要な ES サーバを作成します。名称以外はデフォルトの設定としました。

次にディプロイパッケージを使ってサービスの登録を行ないます。 <Enterprise Server インストールディレクトリ>/deployの下に任意のディプロイ用ディ レクトリを作成し、その下に4で作成した car ファイルと実行形式ファイルをコピーし ます。

<Enterprise Server インストールディレクトリ>/deploy/.mfdeploy を以下の様に書き換えます。

MFDS=localhost MFUS_SERVER=ESSRV01 MFUS_LISTENER=Web Services

<Enterprise Server インストールディレクトリ>/deploy の下のディプロイ用ディレクト リに移動して下記のコマンドを実行します。 mfdepinst srv01.car

すると.mfdeployに記載した ES サーバ上に car ファイル作成時に指定した名称でサービ スが作成されます。登録された ES サーバは開始(Start)ボタンで手動で開始できます。





£	1 - 1 c	of 1 Serv	ers			Show 10 servers a	ata time 🔤	< 前へ] [次へ >>]
	タイプ	名前	現 ステータ フ	通信 プロセス	ラセンス	ステータス ログ	オブジェクト	說明
編集	MFES	ESSRVI	開始 運行 停止…	1 top:172.10.1,183*60272 ² (rp3440-3) � 2 リスナー 詳細	* / 10	MD S3800E Server started successfully 12:06:21 1 minute 33 seconds in '테하섬 '' state since 12:06:21	サ 3 ビ ス ハ 2 ンド デ オ ク ゴ ビ フ バッ 1 ク 詳細 ジ	Communications server for Web Services
追加								

サーバー	·) עגד-	(2)] サー	ビス (3)	וכתן	*ラ (2) <mark>パッケージ</mark>	(1)		画面更新
1 - 1 of 1	l packages				Show 10	packag	es at a	a time 次へ >>
	名前	現 ステータ ス	ステー タス ログ	パッケ ージ モジュ ール	П	パッケージ パス	カスタ ム 構成	設印
編集(STV01.CBLAPP01	Available	ок		/u01/deploy/srv01/srv01.idt	/u01/deploy/srv01		created 15??30??22?b ?? 2006-03-27 fror srv01/srv01.car

A Ser	ver ESSR	V01 [開始]							
サーバ	ען	スナー (2)	サービス	. (3)	ハンドラ (2) バッケージ	2 (1)				
サービス表示フィルタ キームスペース: オペレーション: クラス: All ノ							ラス: All 💽 ハンドラ			
1 - 3 o	f 3 display	able name:	spaces froi	m a	total of 3			Sh	ow 1	0 service namespa
	サービス ネームス ペース	オペレーショ ン	サービス クラス	探索順序	リスナー	要求 ハンドラ	実装 パッケージ	現 ステータ ス	ステ - タ ス ログ	カスタム 構成
	Deployer	Deployer 編集…	MF deployment	1	1 CP 1 Web top:172.16.0.104*:47055* (necp2 +)			Available	ОК	[MF client] scheme=http URL=/cq accept=application/x-zip-compres listener=Web Services
	ES	ES 編集…	MF ES	1	1 CP 1 Web Services top:172.16.0.104*:47055* (necp2 +)			Available	ОК	
削除	srv01	1 of 1 oper	ations sho	wn			,			
		.CBLAPP01 編集		1	1 CP 1 Web Services top:172.16.0.104*:47055* (necp2 +)	MFRHBINP	srv01.CBLAPP01	Available	ок	
追加					·					

次にリスナーの設定をします。

ES サーバを停止後、リスナーの画面の編集(Edit)ボタンで編集画面を開 き、"WebServices"の中に"*:*"で設定されている IP アドレス:ポート番号を固定で 設定します。

ここで設定する IP アドレスはパッケージとしてフェイルオーバーする場合には再配置可能 IP (172.16.104.1)を、パッケージ化せずに単体でテストする場合には定常 IP (172.16.1.183)を設定すると便利です。





サーバー	UZ+-	- (2)	サービス (3) ハン	f∋(2)].	バッケージ (0		画面更新	自動更新開稿 (8) [
ノスナー表	ホフィルター	カセス	All 💽 숫話성	AI AI] ステータ			
\$19. 通	信ブロセス 1		自動起動				統計	Chang	e ⊐Ľ ∽	Ō
Sector ed	リスナー	プロセス	10 コントロールチャネ!	レアドレス		ステータ	ス 封回のステ	一身ス実更	ステータスログ	
編集	2 追加	process 7035	tep:172.16.1.183*;470 (1j:3440-3:+)	625	a a la sola de la	89%	03/17/06-17:	01:13	Received admin reques	n 6 re:
		名前	アドレス	27-92	前回のステー	タス変更	27-9202	송분947	カスタム様成	50.0
	編集	Web Services	5cp:172.16.104.1.9003	B8 36	03/17/06-16:37	15	ок	Web Services and J2EE		Web Serv HTT
	編集	Web	top:172.16.1.183*:47056* (1p3448-3 +)	MAS	03/17/06-16:37	£5	ok:	unes.	(virtual paths) ogin <es>/bin uploads=<es>/deploy decsn<es>/decs/tern/</es></es></es>	8as neb

6. クライアントの作成

本検証で Enterprise Server 上にディプロイしたサービスにリクエストを出すのは Java のクライアントプログラムです。(ソースサンプルは巻末に記載)

このプログラムを利用するには\$CLASSPATHに下記の3ファイルを指定する必要があります。

j2ee. jar

5 5	
mfconnector.jar	(〈Enterprise Server インストールディレクトリ〉/lib に存在)
mfcobolpure.jar	(<enterprise server="" インストールディレクトリ="">/lib に存在)</enterprise>





検証

テストパターン

本検証では以下の 10 パターンを確認しました。

パターン	タイトル	説明
1	手動切り替え	Serviceguard への操作による切り替えを確認
2	データ LAN 両系(二重)	稼動側サーバに接続されたデータ LAN のケーブルを 2
	障害	本とも引き抜いて切り替えを確認
3	業務障害	業務監視用シェルの異常検出による切り替えを確認
4	OS のみで待機	待機状態のノードでは MFDS 起動せず切り替えを確認
	(シングルスタンバイ)	
5	MFDS 起動で待機	待機状態のノードでは MFDS 起動して切り替えを確認
	(高速スタンバイ 1)	
6	DS サーバ起動で待機	待機状態のノードでは MFDS、DS サーバを起動状態で切
	(高速スタンバイ 2)	り替えを確認
$\overline{\mathcal{O}}$	別の業務を提供しなが	待機状態のノードでは MFDS 起動、切り替える DS サー
	ら待機	バとは別の DS サーバを動作させせながら切り替えを確
	(相互スタンバイ)	認。ノード1⇔ノード2の双方向で確認
8	ディプロイパッケージ	ディプロイするパッケージを共有化し、切り替え後に
	を共有化	同じパッケージを引き継ぐことを確認
9	リポジトリを共有化	MFDS のリポジトリファイルを共有化し、切り替え後に
		Enterprise Server の管理情報を引き継ぐことを確認
10	ログ出力先を共有化	ログファイルの出力先を共有化し、切り替え後のログ
		照会ができる事を確認

①~③はフェイルオーバーのトリガーのテストです。代表的なトリガー3種類のパターンで確認します。

フェイルオーバーをステータスで確認します。

 ④~⑦は待機側のノードの状態を変えるテストです。本検証では Enterprise Server の プロセスの実行状態を変えて確認します。
 Java クライアントから繰り返しリクエストを出し続け、フェイルオーバー後もサービス が提供され続ける事を確認します。

⑧~⑪は Enterprise Server システムが利用する情報を共有化するテストです。

以降、パターン毎の設定と検証結果を記述します。





パターン① 手動切り替え

<目的>

通常の運用でも行なわれる動作の確認。

Serviceguardのコマンドを実行し、パッケージの移動が行なわれるか検証する。

<構成>



<設定のポイント> ES 管理シェル mf. sh の start) に組み込む起動コマンド

killmfds.sh
killallcas.sh ESSRV01
mfds &
/wait 5
casstart /rESSRV01

ES 管理シェル mf. sh の halt) に組み込む停止コマンド

casstop	/rESSRV01
sleep 5	
killmfd	s. sh





<検証>

このパターンでは Serviceguard コマンドにてフェイルオーバーを行なう。 rp3440-3 でパッケージ ESSRV01 を開始しておく。

MICRO Focus	Enterprise Server Administration パージョン 10400 rp3440.3 (172.16.1.183)
Home	Status MDS0000I OK
アクション 保存 後旧 インボート すべて削除	画面更新 自動更新間隔 (秒) 10
シャットダウン	271
構成 オブション	¹¹ ステータ 通信 ン タイプ 名前 ス
ユー ザ 追加 更新	■集 MFE5 ● この ● この ● ロッコン 100 1100-1103-00272* (7) 100 1 0 リスナー 単通 ● ロッコン 100 1 1 ロッコン 101 1103-00272* 2 リスナー 単通
表示 ディレクトリ 統計 セッション	
ジャーナル ヘ ルブ	<u>غاما</u>
このページ	

次に Serviceguard コマンドにてパッケージ停止する。

> cmhaltpkg ESSRV01

このコマンドによりパッケージの移動処理が開始する。

すると、パッケージ ESSRV01 の停止処理により MFDS が止まるため rp3440-3 の管理画面 が表示できなくなる

i ページを	を表示できません	
検索中のページは 生しているか、ブラ	は現在、利用できません。Web サイトに技術的な問題が発 ラウザの設定を調整する必要があります。	
次のことを試してく	(tau	
 ① (ご 町約) アドレスノ 正く入入) 25(水の) 25(ネン) 25(ネ	ボタンをクリックするか、後でやり直してください。 ドーにページ アドレスを入力した場合は、ページ アドレスを リンたかどうかを確認してください。 定を確認するはえ (シール) メニューの 「インターネット オ をクリックします。日報点 タブで 「ダイヤルアップの設定」グ 国定1 ボタン または LAN の 別を立 ブリル・プの [LAN の シンをクリックしてください。設定情報は、LAN ローカル エ ワーク)の管理者か、ISP (インターネット サービス プロパ 経球する情報と一致する必要があります。 小管理者がネットワークの接続の設定を使用可能してい rosoft Windows を使用して、シットワークの接続調整を行 動きりにネットワークの接続の設定を見つけることができま	
そして rp	3440-4 でパッケージ ESSRV01 が開始	される。
MICRO FOCUS	Enterprise Server Administration 17-2/32/10400 rp3440.4 (172.16.1.184)	
Home	Status MDS0000I OK	
アクション 保存 復旧 インポート	画面更新 自動更新間隔(秒) 10	
すべて削除 シャットダウン	1 - 1 of 1 Servers	
構成 オブション	マチータ タイプ 名前 ステータ 通信 ン フテータ フロセス ス i	
ユーザ 追加 更新	編集 Mr Es ESSRV0 開始 (1 top:172.16.1.184*.30847* (7)2440-4) ✓ 10 s 2 リスナー 単語	
表示 ディレクトリ 統計 セッション ジャーナル	(ietp)	
	75 / U	

<結果>

ヘルブ

Serviceguard のコマンド cmhaltpkg によるパッケージの移動を確認。





パターン② データ LAN 両系 (二重) 障害

<目的>

Serviceguardの機能のみで検知する障害パターンの確認 稼動側サーバに接続されたデータ LAN の両系 (二重) 障害を起こし、パッケージの移動が 行なわれるか検証する。

<構成>



<設定のポイント> ES 管理シェル mf. sh の start) に組み込む起動コマンド

killmfds.sh
killallcas.sh ESSRV01
mfds &
/wait 5
casstart /rESSRV01

ES 管理シェル mf. sh の halt) に組み込む停止コマンド

casstop /rESSRV01	
sleep 5	
killmfdo ob	
KIIIIIIUS. SII	

クラスタ構成ファイル cmclconf.ascii にてデータ LAN 両系(二重)の定義をしておく





<検証> rp3440-3	<mark>でパッケージESSRV01 を開始して</mark> おく。
MICRO Focus	Enterprise Server Administration ^{N-932/10400} ^{rp3440.3} (172.16.1.183)
Home	Status MDS00001 OK
アクション 保存 復旧 インポート すべて削除	画面更新 自動更新間隔(秒) 10 1 - 1 of 1 Servers
シャットス・リン 構成 オブション	ライ 現 ライ ステータ タイプ 名前 2 フロセス ス
ユーザ 追加 更新	編集 MFES ESSR00 ⁴ 開始 単型 合 「作业 「社 top-172.16.1.183*66596 (pp3440.3) ⁴ 2 リスナー 詳細
表示 ディレクトリ 統計 セッション	
シャーナル ヘルブ	1870

稼動側サーバに接続されたデータ LAN のケーブルを2本とも引き抜く。

すると、パッケージ ESSRV01 の停止処理により mfds が止まるため rp3440-3 の管理画面 が表示できなくなる。

i ページを表示できません	
検索中のページオ現在、利用できません。Web サイトに技術的な問題が発 生しているか、ブラウザの設定を調整する必要があります。	
 次のことを誘してください: (ご) (更新) ボタンをクリックするか、後でやり直してください。 アドレス バーにページ アドレスを入力した場合は、ページ アドレスを 正しく入力したかどうかを確認してください。 (お高の設定を確認とするは、レール メニューの [インターネットオ ブション]をクリックします。(服務) タブで (ヴィヤルアップの設定) ヴ ルーブの 国家定1 ボタン、または [LAN の設定] グルーブの [Bな2] ボタンをクリックしてください。 設定情報通よ LAN ローカル エ リアネット ワーク) の管理者か、SP (インターネット サービス ブロバ イダ) が指供する情報をついまする必要があります。 ネットワークの接触でもなったのます。 	

そして rp3440-4 でパッケージ ESSRV01 が開始される。

MICRO Focus	Setterprise Server Administration アトージョン 10400 rp3440.4 (172.16.1.184)	
Home	Status MDS0000I OK	
アクション 保存 復旧	画面更新 自動更新間隔(秒) 10	
インホート すべて削除 シャットダウン	1 - 1 of 1 Servers	
構成 オブション	現. ステータ 通信 ン・ タイプ 名前 ス. ブロセス ス. 1	
ユーザ 追加 更新	▲ MFES ESSRV0(開始) 日 10p:172.10.1184*04182* (p>440-0)√ 2 リスナー 詳細	
表示 ディレクトリ 統計		
ジャーナル	注意加	

<結果>

データ LAN 両系(二重)障害によるパッケージの移動を確認。





パターン③ 業務障害

<目的>

ユーザー作成の監視シェルにより検知する障害パターンの確認 本検証では Enterprise Server のプロセス異常を想定し、主要サービスである MFDS を停 止することでパッケージの移動が行なわれるか検証する。

<構成>



データLAN (lan0/1)

<設定のポイント>

ES 管理シェル mf. sh の start) に組み込む起動コマンド

killmfds.sh
killallcas.sh ESSRV01
mfds &
/wait 5
casstart /rESSRV01

ES 管理シェル mf. sh の halt) に組み込む停止コマンド

casstop /rESSRV01	
sleep 5	
killmfds.sh	

cntl ファイルにて監視用のシェルを起動するように設定

本検証では ES 管理シェル mf. sh の monitor) で記述した監視処理を起動した。 このシェルは自動的に監視対象をチェックし続け、異常を検知した時に終了するように 作成する。この監視処理は mfdump の出力の Web Service のステータスにて異常を検知す る方式とした。





<検証> rp3440-3	でパッケージ ESSRV01 を開始しておく	0
MICRO Focus	Enterprise Server Administration ^{71-932/10400} ¹⁹³⁴⁴⁰³ (172.16.1.183)	
Home	Status MDS0000I OK	
アクション 保存 復旧 インポート すべて削除	画面更新 自動更新間隔(秒) 10	
ンキットタワン 構成 オブション	タイプ 名前 フテータ ライ セ タイプ 名前 フテータ 通信 ン フラータ ブロセス ス ス	
ユーザ 追加 更新	■ SSRV0(開始) ■ SSRV0(開始) (内)240-3) ✓ 2 リスナー 詳細 (行止)	
表示 ディレクトリ 統計 セッション		
ジャーナル	1901	

次にMFDSのプロセスをkill する。 > kill -9 <mfds32のpid>

すると、ES 管理シェル mf.sh の monitor)が障害を検知しフェイルオーバーが開始する。 mfds が止まるため rp3440-3 の管理画面が表示できなくなる。

i	ページを表示できません
検索中 生して	9のページは現在、利用できません。Web サイトに技術的な問題が発 いるか、ブラウザの設定を調整する必要があります。
次のこ	とを話してください
•	○ 更新1 ボタンをクリックするか、後でやり直してください。 アドレス バーにページ アドレスを入力した場合は、ページ アドレスを 正しく入力したかどうかを確認してください。 接続の設定を確認するには、ビシール メニューの ビンターネット オ ブション)をクリックします。「接続」 タブで じイヤルアップの設定」 グ ループの 励定」 ボタン、または (LAN の設定」 グループの [LAN の 設定」 ボタンをクリックしてください。設定情報は、LAN (ローカル エ リアネット ワーグ)の管理者か、ISP (インターネット サービス プロパ イタ) が提供する情報が、マサマム、東部ではす。
•	コンパカ症状でショー時報にとなりションを外的見まで、 キットワークを調査がネットワークの接続の設定を使用可能にしてい れば、Microsoft Windows を使用して、ネットワークの接続試験を行 ったり、自動的にネットワークの接続の設定を見つけることができま

そして rp3440-4 でパッケージ ESSRV01 が開始される。

MICRO Focus	Enterprise Server Administration バージョン 10400 ŋ3440.4 (172.16.1.184)
Home	Status MDS00001 OK
アクション 保存 復旧 インポート すべて削除	画面更新 自動更新間隔(秒) 10]
メマットス リン 構成 オプション	タイプ 名前 第一ク 3月一ク ブロセス ライ ス ライ ン ライ ン MFES ESSRV0(開始) 11 cor 1/32 (6.1184** 54162**) * / 1
ユー ザ 追加 更新	編集… ● (75340-4) ✓ 10 1 2 リスナー [¥禮]
表示 ディレクトリ 統計 セッション ジャーナル	
ヘルブ	道加

<結果>

ユーザー作成の監視シェルにより検知した障害によるパッケージの移動を確認。





パターン④ 0S のみで待機

<目的>

待機側のノードで Enterprise Server の関連プロセスは起動しない状態からのフェイル オーバーを確認する。

フェイルオーバー前と後で同じサービスが継続される事も確認する。









<検証> rp3440-3	でパッケージ ESSRV01 を開始しておく。
FOCUS	Enterprise Server Administration p3440.3 (172.16.1.183)
Home	Status MDS00001 OK
アクション 保存 復旧 インボート すべて削除 シャットダウン	画面更新 自動更新間隔(秒) 10 1 - 1 of 1 Servers 1 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 -
構成 オブション	日本 現 ステータ 道信 ン タイプ 名前 フロセス ス
ユーザ 追加 更新	■集 MEES ESSRUO (開始: (1) 1027-00090 10 (1) 1027-00090 10 (1) 1027-00090 10 2 リスナー 詳細
表示 ディレクトリ 統計 セッション	
シャーチル ヘルブ	(IIII)

クライアントマシンで Java クライアントプログラムを一定の周期で実行を繰り返す。 シェル例 (Run java Test. sh)

#!/bin/sh
while ["1" == "1"]; do
java -
$\verb classpath.:/home/clinux2/test/mfcobol.jar:/home/clinux2/test/mfconnector.jar:/home/clinux2/test/mfconnector.jar:/home/clinux2/test/mfcobol.jar:/home/clinux2/test/mfconnector.jar:/home/clinux2/test/mfcobol.jar:/hom$
est/j2ee.jar clptest
#sleep 10
done

稼動側サーバに接続されたデータ LAN のケーブルを2本とも引き抜く。

すると、パッケージ ESSRV01 の停止処理により mfds が止まるため rp3440-3 の管理画面 が表示できなくなる。

1 ページを表示できません
検索中のページは現在、利用できません。Web サイトに技術的な問題が発 生しているか、ブラウザの設定を調整する必要があります。
 次のことを読してください。 ● (ご) 更新) ボタンをクリックするか、後でやり直してください。 ● アドレス バーコページ アドレスを入力した場合は、ページ アドレスを 正しく入力したかどうかを確認してください。 ● 接続の設定を確認するコよ (シール) メニューの ドレッシュネット オ ゴンョン をクリックします。(接続) タブで (ダイヤルアッゴの設定) グ ルーゴの(既定) ボタン、または LANの設定) グルーゴの(LANの)
設定] ボタンをクリックしてください。設定情報師よ、LAN (ローカル)レ リアネット ワーク)の管理者か、ISP (インターネット サービス プロバ イダ)が提休する情報とつ数する必要があります。 ・ネットワーク管理者がネットワークの接続の設定を使用可能にしてい れば、Microsoft Windows を使用して、ネットワークの接続試験を行 ったり、自動的にネットワークの接続の設定を見つけることができま す。





そして rp3440-4 でパッケージ ESSRV01 が開始される。



クライアントプログラム実行結果

[root@test]# /PunjavaTest_sh
[10010000] $[10010000]$ $[100100000]$ $[1001000000]$ $[1001000000000000000000000000000000000$
06/03/27 12:13:04.01 00LAFF01 19440-3 401
00/03/27 12:15:05:04 00LAPP01 (1)3440-3 402
06/03/2/ 12.13.06.06 CBLAPPOI (73440-3.483
06/03/27 12:13:06.72 CBLAPPOT rp3440-3 484
06/03/27 12:13:07.34 GBLAPP01 rp3440-3 485
06/03/27 12:13:07.98 CBLAPP01 rp3440-3 486 / rp3440-3 でサービスが稼動
06/03/27 12:13:08.63 CBLAPP01 rp3440-3 487
06/03/27 12:13:09.25 CBLAPP01 rp3440-3 488
06/03/27 12:13:09.90 CBLAPP01 rp3440-3 489
06/03/27 12:13:10.54 CBLAPP01 rp3440-3 490
06/03/27 12:13:11.17 CBLAPP01 rp3440-3 491
06/03/27 12:13:11.83 CBLAPP01 rp3440-3 492
javax resource spi FISSystemException: CobolException Server did not return any data or header executing
srv01 CRIAPPO1
iava nat ConnectException: Connection refused (arrac'220)
Java. Het. Johneyet. Asseption. Johneyet. Johneyet. (Although and Although and Alth
at java. Het. Fransooketingt. Sokketoonneot (Matike Method)
at java. net. Plainsocketimpi. doconnect (Plainsocketimpi. java. 305)
at java. net. PlainSocketimpl. connectioAddress (PlainSocketimpl. java: 1/1)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:158) サーヒス中断中
at java. net. Socket. connect (Socket. java:462)
at java. net. Socket. connect (Socket. java:412)
at java.net. Socket. <init>(Socket. java:308)</init>
at java.net.Socket. <init>(Socket.java:153)</init>
at com.microfocus.cobol.connector.transport.BINPStream.createSocketWithTimeout(BINPStream.java:894)
at com.microfocus.cobol.connector.transport.BINPStream.createSocketWithTimeout(BINPStream.java:825)
at com.microfocus.cobol.connector.transport.BINPStream. <init>(BINPStream.java:97)</init>
at com microfocus cobol connector transport BINPRemoteCall open (BINPRemoteCall, java:176)
at com microfocus cobol connector transport BINPRemoteCall connect(BINPRemoteCall iava:73)
at com microfocus cobol connector transport BINPRemoteCall reconnect(BINPRemoteCall java:145)
at com microfocus cobol connector PureCobolBean cobcall(PureCobolBean java:285)
at commiscroficuls cohol connector cci CoholInteraction ever (CoholInteraction is 247)
at commission focus codel connector coil obel interaction execute (chellinteraction is a commission focus code) connector coil cohellinteraction execute (chellinteraction is a commission focus code) connector coil content action is a contraction in the commission of the commission
at ciptest. main (ciptest. java.40)
Javax. resource. spi. ElssystemException. Cobolexception Exception Enrow during open (see getCause() for more
Information) (elapsed time=224.0) for 1/2. 10. 104. 1. 9003 (cause class. Java. net. ConnectException,
cause:Connection refused (errno:239)) executing srv01.CBLAPP01
at ciptest.main(ciptest.java:4/)
(繰り返し・中略)
java.net.ConnectException: Connection refused (errno:239)
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:305)
at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:171)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:158)
at java.net.Socket.connect(Socket.java:462)
at java.net.Socket.connect(Socket.java:412)
at java.net.Socket. <init>(Socket.java:308)</init>
at java net Socket <init>(Socket java:153)</init>
at commiscoficcus cohol connect juration,
at com microficus cohol connector transport BINPStream createSocketWithTimoott (BINPStream java 305)





at com.microfocus.cobol.connector.transport.BINPStream. <init>(BINPStream.java:97)</init>
at com.microfocus.cobol.connector.transport.BINPRemoteCall.open(BINPRemoteCall.java:176)
at com.microfocus.cobol.connector.transport.BINPRemoteCall.connect(BINPRemoteCall.java:73)
at com.microfocus.cobol.connector.transport.BINPRemoteCall.reconnect(BINPRemoteCall.java:145)
at com.microfocus.cobol.connector.PureCobolBean.cobcall(PureCobolBean.java:285)
at com.microfocus.cobol.connector.cci.CobolInteraction.exec(CobolInteraction.java:247)
at com.microfocus.cobol.connector.cci.CobolInteraction.execute(CobolInteraction.java:174)
at clptest.main(clptest.java:48)
javax.resource.spi.ElSSystemException: CobolException Exception throw during open (see getCause() for more
information) (elapsed time=230.0)for 172.16.104.1:9003 (cause class:java.net.ConnectException,
cause:Connection refused (errno:239)) executing srv01.CBLAPP01
06/03/27 12:13:29.88 CBLAPP01 rp3440-4 493
06/03/27 12:13:30.54 CBLAPP01 rp3440-4 494
06/03/27 12:13:31.19 CBLAPP01 rp3440-4 495
06/03/27 12:13:31.85 CBLAPP01 rp3440-4 496
06/03/27 12:13:32 49 CBLAPP01 rp3440-4 497
06/03/27 12:13:33.13 CBLAPP01 rp3440-4 498
06/03/27 12:13:33.77 CBLAPP01 rp3440-4 499
06/03/27 12:13:34.46 CBLAPP01 rp3440-4 500 rp3440-4 でサービスを引継ぎ。
06/03/27 12:13:35.09 CBLAPP01 rp3440-4 501 データファイルの連番も継続
06/03/27 12:13:35.70 CBLAPP01 rp3440-4 502
06/03/27 12:13:36.38 CBLAPP01 rp3440-4 503
06/03/27 12:13:36.97 CBLAPP01 rp3440-4 504
06/03/27 12:13:37.61 CBLAPP01 rp3440-4 505
06/03/27 12:13:38.25 CBLAPP01 rp3440-4 506
06/03/27 12:13:38 89 CBLAPP01 rp3440-4 507
06/03/27 12:13:39 54 CBLAPP01 rp3440-4 508
, , , , , , , , , , , , , , , , , , , ,
[root@test]#

<結果>

待機側のノードで Enterprise Server の関連プロセスは起動しない状態からのパッケージの移動を確認。

また、一定時間のサービス中断後、同じサービスを待機側で提供できていることも確認。





パターン⑤ MFDS 起動で待機

<目的>

待機側のノードで Enterprise Server の基本プロセス MFDS のみ起動した状態からのフェ イルオーバーを確認する。

フェイルオーバー前と後で同じサービスが継続される事も確認する。

<構成>



クラスタ構成ファイル cmclconf. ascii にてデータ LAN 両系(二重)の定義をしておく





<検証> rp3440-3 でパッケージ ESSRV01 を開始しておく。 🔏 Enterprise Server Administration FOCUS バージョン 1.04.00 rp3440-3 (172.16.1.183) Status MDS0000I OK Home **アクション** 保存 画面更新 自動更新間隔(秒) 10 液行 復旧 インポート すべて削除 シャットダウン + 1 - 1 of 1 Servers 現 ステータ **構成** オブション 通信 プロセス タイプ 名前 える 編集... MFES 開始 1 top:172.16.1.183*:60596* (rp3440-3) ✔ */ ユー**ザ** 追加 更新 2 リスナー [詳細] 停止... **表示** ディレクトリ 統計 セッション ジャーナル 追加 ヘルナ

rp3440-4 では ES サーバ(パッケージ ESSRV01)は停止している。



クライアントマシンで Java クライアントプログラムを一定の周期で実行を繰り返す。 シェル例 (Run javaTest. sh)

```
#!/bin/sh
while [ "1" == "1" ]; do
java -
classpath .:/home/clinux2/test/mfcobol.jar:/home/clinux2/test/mfconnector.jar:/home/clinux2/t
est/j2ee.jar clptest
#sleep 10
done
```

稼動側サーバに接続されたデータ LAN のケーブルを2本とも引き抜く。

すると、パッケージ ESSRV01 の停止処理により ES サーバ ESSRV01 が停止する。

注)タイミングによって ES サーバ ESSRV01 のステータスは「応答なし」なることがあり ます。











java.net.ConnectException: Connection refused (errno:239)
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:305)
at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:171)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:158)
at java. net. Socket. connect (Socket. java: 462)
at java.net.Socket.connect(Socket.java:412)
at java.net.Socket. <init>(Socket.java:308)</init>
at java.net.Socket. <init>(Socket.java:153)</init>
at com.microfocus.cobol.connector.transport.BINPStream.createSocketWithTimeout(BINPStream.java:894)
at com.microfocus.cobol.connector.transport.BINPStream.createSocketWithTimeout(BINPStream.java:825)
at com.microfocus.cobol.connector.transport.BINPStream. <init>(BINPStream.java:97)</init>
at com.microfocus.cobol.connector.transport.BINPRemoteCall.open(BINPRemoteCall.java:176)
at com.microfocus.cobol.connector.transport.BINPRemoteCall.connect(BINPRemoteCall.java:73)
at com.microfocus.cobol.connector.transport.BINPRemoteCall.reconnect(BINPRemoteCall.java:145)
at com.microfocus.cobol.connector.PureCobolBean.cobcall(PureCobolBean.java:285)
at com.microfocus.cobol.connector.cci.CobolInteraction.exec(CobolInteraction.java:247)
at com.microfocus.cobol.connector.cci.CobolInteraction.execute(CobolInteraction.java:174)
at clptest.main(clptest.iava:48)
javax resource spi EISSystemException: CobolException Exception throw during open (see getCause() for more
information) (elapsed time=220.0) for 172.16.104.1:9003 (cause class:java.net.ConnectException,
cause:Connection refused (errno:239)) executing srv01.CBLAPP01
06/03/27 12:27:54.12 CBLAPP01 rp3440-4 556
06/03/27 12:27:54.80 CBLAPP01 rp3440-4 557
06/03/27 12:27:55.65 CBLAPP01 rp3440-4 558
06/03/27 12:27:56.27 CBLAPP01 rp3440-4 559
06/03/27 12:27:56.91 CBLAPP01 rp3440-4 560 rp3440-4 でサービスを引継ぎ。
06/03/27 12:27:57 56 CBI APP01 rp3440-4 561 データファイルの連番も継続
06/03/27 12:27:58 20 GBI APP01 rp3440-4 562
06/03/27 12:27:58 84 CBI APP01 rp3440-4 563
06/03/27 12:27:59 47 CB APP01 rp3440-4 564
06/03/27 12:28:00 09 CB APP01 r03440-4 565
06/03/27 12:28:00 74 CBL 4PP01 rp340-4 566
06/03/27 12:28:01 39 CBI APP01 rp340-4 567
[root@test]#

<結果>

待機側のノードで Enterprise Server の基本プロセス MFDS のみ起動した状態からのパッケージの移動を確認。

また、一定時間のサービス中断後、同じサービスを待機側で提供できていることも確認。





パターン⑤ ES サーバ起動で待機

<目的>

待機側のノードで Enterprise Server の基本プロセス MFDS と ES サーバを起動した状態 からのフェイルオーバーを確認する。

フェイルオーバー前と後で同じサービスが継続される事も確認する。

<構成>



クライアント

<設定のポイント>

それぞれ各ノードであらかじめ MFDS と ES サーバ ESSRV01 を立ち上げておく。 パッケージでは再配置可能 IP アドレスと共有ディスクの切り替えのみを行なう。

クラスタ構成ファイル cmclconf.ascii にてデータ LAN 両系(二重)の定義をしておく





<検証> <u>rp3440-3 ⁻</u>	で ES サーバ ESSRV01 を手動で開始しておく。
MICRO Focus	Enterprise Server Administration ^{//−9} (172.16.1.183)
Home	Status MDS0000I OK
アクション 保存 復日 インポート すべて削除	画面更新 自動更新間隔(秒) 10] ■ 1 - 1 of 1 Servers
シャットシリン 構成 オブション	タイプ 名前 現 ステータ フロセス ライ と フロセス シ ス
ユー ザ 追加 更新	編集… MFES ESSRAD(開始) 県 留 留 第 1 1 cp:172.16.1133*t00500* 10 1 cp:172.16.1133*t00500* 10 1 cp:172.16.1133*t0050* 10 1 cp:172.16.113*t005* 10 1 cp:172.15.113*t005* 10 1 cp:172.15.113*t005*t005* 10 1 cp:172.1
表示 ディレクトリ 統計 セッション ジャーナル	
ヘルプ	通加

rp3440-4 で同様に ES サーバ ESSRV01 を手動で開始する。

<u>ところが、rp3440-4 でESSRV01 を立ち上げるとリスナーがエラーで立ち上がらない。</u>

FOCUS	Server Administration > ESSRV01 > U2.5 - M-932/12400 Server2 (10.34.69.36)
Home	Status MDS00001 OK
アクション 保存	▲ Server ESSRV01 [開始]
復旧 インボート すべて利用	サーバー (リスナー (2) サービス (3) ハンドラ (2) パッケージ (1)
9 へ Clenox シャットダウン	リスナー表示フィルタ プロセス: All 💌 会話タイプ: All 💌 ステー
構成 オプション	⇒ 当 通信プロセス 1 ▼ 自動起動 続
	リスナー プロセスロ コントロールチャネルアドレス ステータス 封囲の
ユーザ 追加	編集 2 追加 process top:10.34.80.38"37041" 開始 02/1405
更新	
+-	名前 アドレス ステータス 前回の ステータス変更 ステータス
表示 ディレクトリ 統計	編集 Web top-10.34.09.38.37917* (存止 02/1409-15.49.54 COMAGE COMPARIANCE COMP
セッション ジャーナル	編集 Web. top:10.34.80.30*.32043* 問語: 02/14.00.15.43:54 0K
ヘルブ このページ	

理由: Enterprise Server のリスナーは指定した IP アドレスが有効になっている必要がある。

<結果>

Enterprise Server の制限事項により、待機側の ES サーバを正常に立ち上がらせること ができないため、本パターンの前提条件を満たせなかった。 よって本パターンでの動作検証は不可。





パターン⑦ 別の業務を提供しながら待機

<目的>

待機側のノードで Enterprise Server の MFDS を起動し、別の ES サーバによるサービス を起動した状態からのフェイルオーバーを確認する。

フェイルオーバー前と後で同じサービスが継続される事も確認する。

<構成>







<設定のポイント> MFDS の制御はパッケージに含めない。

ESSRV01の開始シェルに組み込む起動コマンド

killallcas.sh ESSRV01 casstart /rESSRV01

ESSRV01の終了シェルに組み込む停止コマンド

casstop /rESSRV01

ESSRV02の開始シェルに組み込む起動コマンド

killallcas.sh ESSRV02 casstart /rESSRV02

ESSRV02の終了シェルに組み込む停止コマンド

casstop /rESSRV02

cntl ファイルにて監視用のシェルを起動するように設定

く検証>

それぞれ各ノードであらかじめ MFDS を立ち上げる。 rp3440-3 でパッケージ ESSRV01 を、rp3440-4 でパッケージ ESSRV02 を開始しておく。



クライアントマシンで各サービスに対する Java クライアントプログラムを一定の周期で 実行を繰り返す。

シェル例(RunjavaTest.sh)

```
#!/bin/sh
while [ "1" == "1" ]; do
java -
classpath .:/home/clinux2/test/mfcobol.jar:/home/clinux2/test/mfconnector.jar:/home/clinux2/t
est/j2ee.jar clptest
#sleep 10
done
```

すると、ES 管理シェル mf.sh の monitor)が障害を検知しフェイルオーバーが開始する。





1 ベージを表示できません	FOC
 検索中のページは現在、利用できません。。Web サイトに技術的な問題が発 生しているか、 ブラウザの設定を調整する必要があります。 	Pome アクション 保存 項目 インボート すべて利 シャットダ
次のことを試してください。	構成 オプション
 ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	ユーザ 通知 更新 表示

止しくハフに方からかを幅記してたさい。 接続の設定を確認するはコミレシールノメニューの「インターネットオ ガション」をグリックします。13歳間、タブで「ダイヤルアップの設定」グ ループの、国金』ボタン、または「LAN の設定」グリループの「LAN の 設定」ボタンをグリックしてください。設定値軸部よ LAN ローカル エ リアネット ワーク」の管理者が、BP イインターネットサービス ブロい イダ) が増出する情報とつ数すると変があります。 ネットワーン管理者がネットワークの接続の設定を使用可能してしい れば、Microsoft Windows を使用して、ネットワークの接続試験を行 ったり、自動的コネットワークの接続の設定を見つけることができま



クライアントプログラム実行結果(CBLAPP01)



at java. net. Socket. connect (Socket. java: 402) at java. net. Socket. connect (Socket. java: 412)





	at java.net.Socket. <init>(Socket.java:308)</init>
	at java.net.Socket. <init>(Socket.java:153)</init>
	at com.microfocus.cobol.connector.transport.BINPStream.createSocketWithTimeout(BINPStream.java:894)
	at com.microfocus.cobol.connector.transport.BINPStream.createSocketWithTimeout(BINPStream.java:825)
	at com.microfocus.cobol.connector.transport.BINPStream. <init>(BINPStream.java:97)</init>
	at com.microfocus.cobol.connector.transport.BINPRemoteCall.open(BINPRemoteCall.java:176)
	at com.microfocus.cobol.connector.transport.BINPRemoteCall.connect(BINPRemoteCall.java:73)
	at com.microfocus.cobol.connector.transport.BINPRemoteCall.reconnect(BINPRemoteCall.java:145)
	at com.microfocus.cobol.connector.PureCobolBean.cobcall(PureCobolBean.java:285)
	at com.microfocus.cobol.connector.cci.CobolInteraction.exec(CobolInteraction.java:247)
	at com.microfocus.cobol.connector.cci.CobolInteraction.execute(CobolInteraction.java:174)
	at clptest.main(clptest.java:48)
	javax.resource.spi.ElSSystemException: CobolException Exception throw during open (see getCause() for more
	information) (elapsed time=224.0) for 172.16.104.1:9003 (cause class:java.net.ConnectException,
	cause:Connection refused (errno:239)) executing srv01.CBLAPP01
	06/03/27 11:07:02.16 CBLAPP01 rp3440-4 202
	06/03/27 11:07:02.81 CBLAPP01 rp3440-4 203
	06/03/27 11:07:03.45 CBLAPP01 rp3440-4 204
	06/03/27 11:07:04.09 CBLAPP01 rp3440-4 205
	06/03/27 11:07:04.92 CBLAPP01 rp3440-4 206
	06/03/27 11:07:05.57 CBLAPP01 rp3440-4 207
	06/03/27 11:07:06.18 CBLAPP01 rp3440-4 208
	06/03/27 11:07:06.85 CBLAPP01 rp3440-4 209
	06/03/27 11:07:07.49 CBLAPP01 rp3440-4 210
	06/03/27 11:07:08.09 CBLAPP01 rp3440-4 211
	06/03/27 11:07:08.76 CBLAPP01 rp3440-4 212
	06/03/27 11:07:09.40 CBLAPP01 rp3440-4 213 <i>server2 でサービスを引継ぎ。</i>
	06/03/27 11:07:10.01 CBLAPP01 rp3440-4 214 <i>データファイルの連番も継続</i>
	06/03/27 11:07:10.65 CBLAPP01 rp3440-4 215
	06/03/27 11:07:11.27 CBLAPP01 rp3440-4 216
	06/03/27 11:07:11.96 CBLAPP01 rp3440-4 217
	06/03/27 11:07:12.74 CBLAPP01 rp3440-4 218
	06/03/27 11:07:13.48 CBLAPP01 rp3440-4 219
	06/03/27 11:07:14.23 CBLAPP01 rp3440-4 220
	06/03/27 11:07:14.87 CBLAPP01 rp3440-4 221
	06/03/27 11:07:15.53 CBLAPP01 rp3440-4 222
	06/03/27 11:07:16.39 CBLAPP01 rp3440-4 223
	06/03/27 11:07:17.03 CBLAPP01 rp3440-4 224
	06/03/27 11:07:17.68 CBLAPP01 rp3440-4 225 J
	[root@test]#
1	

クライアン	トプロ	グラム実行結果	(CBLAPP02)
-------	-----	---------	------------

		-/
	[root@test]# ./RunjavaTest2.sh	
	06/03/27 11:06:18.80 CBLAPP02 rp3440-4 348	
	06/03/27 11:06:19.43 CBLAPP02 rp3440-4 349	
	06/03/27 11:06:20.05 CBLAPP02 rp3440-4 350	
	06/03/27 11:06:20.68 CBLAPP02 rp3440-4 351	
	06/03/27 11:06:21.40 CBLAPP02 rp3440-4 352	
	06/03/27 11:06:22.05 CBLAPP02 rp3440-4 353	
	06/03/27 11:06:22.67 CBLAPP02 rp3440-4 354	
	06/03/27 11:06:23.28 CBLAPP02 rp3440-4 355	
	06/03/27 11:06:23.88 CBLAPP02 rp3440-4 356	
	06/03/27 11:06:24.50 CBLAPP02 rp3440-4 357	
	06/03/27 11:06:25. 13 CBLAPP02 rp3440-4 358	
	06/03/27 11:06:25.73 CBLAPP02 rp3440-4 359	server2 でサービス中断せず
	06/03/27 11:06:27 06 08 ADD02 rp3440-4 360	
	06/03/27 11:06:27 00 GBLAPPUZ rp3440-4 301	
	06/03/27 11:06:27 09 00LAPP02 103440-4 302	
	06/03/27 11:06:28 35 6DLAFF02 103440-4 363	
	06/03/27 11:06:20 60 CBLAFF02 103440-4 364	
	06/03/27 11:06:29:00 0DEAT 02 103440 4 303	
	06/03/27 11:06:30 84 CBLAPP02 rp3440-4 367	
	06/03/27 11:06:31 50 CBLAPP02 rp3440-4 368	
	06/03/27 11:06:32 14 CBI APPO2 rp3440-4 369	
	06/03/27 11:06:32 77 CBI APP02 rp3440-4 370	
	06/03/27 11:06:33.42 CBLAPP02 rp3440-4 371	
	06/03/27 11:06:34.04 CBLAPP02 rp3440-4 372	
	06/03/27 11:06:34.63 CBLAPP02 rp3440-4 373	
	06/03/27 11:06:35.27 CBLAPP02 rp3440-4 374	
	06/03/27 11:06:35.94 CBLAPP02 rp3440-4 375	
	06/03/27 11:06:36.77 CBLAPP02 rp3440-4 376	
- 1		





06/03/27	11:06:37.	40	CBLAPP02	rp3440-4	377
06/03/27	11:06:37.	99	CBLAPP02	rp3440-4	378
06/03/27	11:06:38.	68	CBLAPP02	rp3440-4	379
06/03/27	11:06:39.	33	CBLAPP02	rp3440-4	380
06/03/27	11:06:39	94	CBLAPP02	rp3440-4	381
06/03/27	11:06:40	59	CBLAPP02	rp3440-4	382
06/03/27	11:06:41	22	CBLAPP02	rp3440-4	383
06/03/27	11:06:41	85	CBI APP02	rn3440-4	384
06/03/27	11:06:42	46	CBI APP02	rn3440-4	385
06/03/27	11:06:43	16		rn3440-4	386
06/03/27	11:06:43	85		rn3440-4	387
06/03/27	11:06:44	46		rn3440-4	388
06/03/27	11.06.45	00	CRI ADDOO	rn3110-1	200
06/03/27	11.00.45	05 05		rp3440-4	300
06/02/27	11.00.40.	61		rp3440-4	201
06/02/27	11.00.40.	01 22		rp3440-4	201
06/03/27	11.00.47.	22 01		1 µ3440-4	202
06/02/27	11.00.4/.	91 55		rp3440-4	204
06/03/27	11.00.40.	10		rp3440-4	394 205
06/02/27	11.00.49.	10		rp3440-4	20E
	11.00.49.	/0		rp3440-4	390
	11.00.00.	40		rp3440-4	39/
	11.00.51.	09	UDLAPPU2	rp3440-4	398
	11:06:51.	/1	UBLAPP02	rp3440-4	399
06/03/27	11:06:52.	34	UBLAPP02	rp3440-4	400
06/03/27	11:06:52.	94	CBLAPP02	rp3440-4	401
06/03/27	11:06:53.	54	CBLAPP02	rp3440-4	402
06/03/27	11:06:54.	20	CBLAPP02	rp3440-4	403
06/03/27	11:06:54.	97	CBLAPP02	rp3440-4	404
06/03/27	11:06:55.	69	CBLAPP02	rp3440-4	405
06/03/27	11:06:56.	28	CBLAPP02	rp3440-4	406
06/03/27	11:06:56.	89	CBLAPP02	rp3440-4	407
06/03/27	11:06:57.	56	CBLAPP02	rp3440-4	408
06/03/27	11:06:58.	18	CBLAPP02	rp3440-4	409
06/03/27	11:06:58.	93	CBLAPP02	rp3440-4	410
06/03/27	11:06:59.	63	CBLAPP02	rp3440-4	411
06/03/27	11:07:00.	35	CBLAPP02	rp3440-4	412
06/03/27	11:07:00.	99	CBLAPP02	rp3440-4	413
06/03/27	11:07:01.	75	CBLAPP02	rp3440-4	414
06/02/27	11.07.02	10		rn2110_1	115

同様に同じ状態から rp3440-4 の MFDS を直接手動で停止し動作を確認する。

<結果>

フェイルオーバーさせる側のノードで Enterprise Server の MFDS 起動した状態かつ別の ES サーバによるサービスを継続提供したままの状態で、パッケージの移動が行える事を 双方向で確認。





パターン⑧ ディプロイパッケージの共有

<目的>

ディプロイリソースの一元管理を目的として、Enterprise Server にディプロイするパッケージを共有ファイルに配置し、フェイルオーバーを確認する。

<構成>



<設定のポイント>

ES 管理シェル mf. sh の start) に組み込む起動コマンド

※ パッケージのディプロイのコマンド mfdepinst は運用上の必要に応じて組み込む。

killallcas.sh ESSRV01 cd /u01/deploy/srv01 mfdepinst srv01.car casstart /rESSRV01	
ES 管理シェル mf. sh の halt) に組み込む停止コマ ※ アンディプロイの処理は手動前提となる。	ンド
casstop /rESSRV01 killmfds.sh	

cntl ファイルにて監視用のシェルを起動するように設定

ES サーバ ESSRV01 にディプロイするサービスは一度削除し、共有ディスクに配置したディプロイパッケージ(. car ファイル)から再度ディプロイ作業を行なう。





本検証では/u01/deploy にディプロイパッケージを配置した

FOCUS		Enterprise S パージョン 1.04.00 rp3440-3 (172.16.1.	Server	Adn	ninist	ration > ESSRVO	1 > パッケージ	
Home	Status	MDS0000I OK						
								[SC
アクション 保存		Server ESSRV01 [開始]					
復旧 インポート	サ.	-バー)リスナー	(2)] +-	ビス (3)	וכת (ドラ (2) パッケージ	(1)	
すべて削除 シャッドダウン 1-1 of 1 packages Show 1				Show 10	packag	es a		
構成 オブション		名前	現 ステータ ス	ステー タス ログ	パッケ ージ モジュ ール	ЮТ	パッケージ パス	カス ム 構成
ユーザ	「編	集 srv01.CBLAPP01	Available	ок		/u01/deploy/srv01/srv01.idt	/u01/deploy/srv01	
追加 更新	追	ta						10
表示 ディレクトリ								

この作業を両ノード共に実施する。

注) 起動時に mfdepinst コマンドにてディプロイする場合は不要

MICRO Focus	Ent ۲۳-۵ ۲۳344	erprise S 197 1.04.00 10-4 (172.16.1.	erver	Adm	ninist	ration > ESSRV0	> パッケージ		
Home	Status MDS0	000I OK							
								[SCHE	MA AD
アクション 保存	A Serve	A Server ESSRV01 [開始]							
復旧 インポート	復旧 インボート サーバー… リスナー (2) サービス (3) ハンドラ (2) パッケージ (1)								
すべて削除 シャットダウン	1 - 1 of 1	packages				Show 10	packag	es at a	a time
構成 オプション		名前	現 ステータ ス	ステー タス ログ	パッケ ージ モジュ ール	тот	パッケージ パス	カスタ ム 構成	說明
ユーザ	編集	srv01.CBLAPP01	Available	ок		Au01/deploy/srv01/srv01.idt	/u01/deploy/srv01		creat srv01
垣加 更新	追加	ll.	1				1		
表示 ディレクトリ									

<検証> rp3440-3	でパッケージ ESSRV01 を開始しておく。
MICRO Focus	Enterprise Server Administration ^{x1-2/32/10400} ^{y2440.3} (172.16.1.183)
Home	Status MDS00001 OK
アクション 保存 復旧 インボート すべて削除	画面更新 自動更新間隔 (秒) 10
シャットダウン 構成 オプション	タイプ 名前 フテータ 通信 フイ シス メフータ ブロセス ス ス ス ス MEE0 ESSRV00[開始] 11 100172 (10.1 10017400241) 人 N N
ユーザ 追加 更新	編集 副 10 s (1)3/1 (1
★示 ディレクトリ 統計 セッション ジャーナル	
ヘルプ	

クライアントマシンで Java クライアントプログラムを実行し動作を確認する。

次に rp3440-3 の MFDS のプロセスを kill する。

≻ kill -9 <mfds32のpid>

すると、ES 管理シェル mf.sh の monitor)が障害を検知しフェイルオーバーが開始する。





そして rp3440-4 でパッケージ ESSRV01 が開始される。



そこでもう一度、クライアントマシンで Java クライアントプログラムを実行し動作を確認する。

<結果>

共有ディスクにディプロイファイルを配置して2つのノード間でディプロイパッケージ を共有する事は可能。





パターン⑨ 構成リポジトリの共有

<目的>

Enterprise Server の各種設定情報を格納した構成リポジトリを共有ファイルに配置し、 フェイルオーバーでこの設定情報が引き継がれるかを確認する。

<構成>



<設定のポイント>

構成リポジトリは MFDS 単位で一箇所のみ有効になり、MFDS の起動時から常に利用され 続ける。よってパッケージの切り替え時には MFDS の制御が必須となる。

ES 管理シェル mf. sh の start) に組み込む起動コマンド

· · · - · · - · · - · ·	
killmfds.sh	
killallcas.sh ESSRV01	
mfds &	
casstart /rESSRV01	
ES 管理シェル mf.shの halt) に組み込む停止コ	マンド
casstop /rESSRV01	
killmfds.sh	

cntlファイルにて監視用のシェルを起動するように設定 設定を行なうノードで構成ファイルを利用できる共有ディスクを有効にする。 ESサーバを全て終了し、MFDSのみ起動させておく。





Enterprise Server Administration 画面のオプション(Option)のリポジトリロケーションを変更する。

MICRO Focus	Benterprise Server Administr 1/-952-10400 rp340-3 (172.16.1.183)	ation >オラション
Home	Status MDS0000I OK	Mon Mar 27 17:46:55 2006
242-24		[SCHEMA ADMINISTRATOR-Developer] [Page id: e000]
(保存 復旧)	リポジトリロケーション:	翌可クライアントセッション:
オノホート すべて削除	(/u01/mfds/	Webプラウザタイムアウト(後): -1
シャットダウン	サーバーモニタン	クライアントプログラムタイムアウト (地): 6000
オザション	有効: 🔍	ジャーナル:
ユーザ 遺加 更新	キーブアライブ間隔(秒): 60 許容応答時間(秒): 5	最大ファイルサイズ(KB): 256 表示されるエントリ: 23
表示 ディレクトリ	UDPブロードキャスト:	ジャーナルレベル:
セッション ジャーナル	UDP探索要求の生成: V	○ エラーと警告
		0 3 ~ C (0) 18 #B

構成リポジトリの格納ディレクトリはデフォルトでは、

<Enterprise Server インストールディレクトリ>/etc/mfds/

に設定されている。

これを共有ファイルに設定する。本検証では /u01/mfds/ とした。

この操作により、/u01/mfds/ に新しい構成リポジトリが生成される。

次に ES サーバ ESSRV01 にディプロイするサービスを共有ディスクに配置し、ディプロイ パッケージ(. car ファイル)から再度ディプロイ作業を行なう。 本検証では/u01/deploy にディプロイパッケージを配置した

MICRO Focus	Enterprise	Server	Adn	ninist	ration > ESSRV0	1 > パッケージ	8
Home	Status MDS0000I OK						
アクション 保存	Server ESSRV01	開始]					[SCH
復旧 インポート すべて削除	ト サーバー リスナー (2) サービス (3) ハンドラ (2) パッケージ (1)				(1)		
シャットタ リン 構成 オブション	2前 名前	現 ステータ ス	ステー タス ログ	パッケ ージ モジュ ール		パッケージ パス	es al カス よ 構成
ユー ザ 追加 更新	編集 srv01.CBLAPP0 追加	1 Available	ок		AuD1/deploy/srvD1/srvD1.idt	AuD1/deploy/srvD1	•
表示 ディレクトリ							

この作業を一つのノードで実施する。

これらの情報はフェイルオーバー時に引き継がれる。







ES サーバ DUMMY の登録を確認。



すると、ES 管理シェル mf.sh の monitor)が障害を検知しフェイルオーバーが開始する。





<結果>

共有ディスクに構成リポジトリを置く事で。2つのノード間で構成リポジトリを共有して利用できた。またディプロイファイルも同時に共有することで、Enterprise Serverの設定変更/ディプロイ作業を一元化できた。





パターン110 ログファイルの共有

<目的>

ES サーバが出力するログファイルの出力先を変更して共有化する。フェイルオーバー後に前ノードのログを参照できる事を確認する。

<構成>



<設定のポイント> ES 管理シェル mf. sh の start) に組み込む起動コマンド

kilimfds.sh kilialicas.sh ESSRVO1 mfds & casstart /rESSRVO1	
ES 管理シェル mf. sh の halt) に組み込む停止コマ	ンド
casstop /rESSRV01	

cntl ファイルにて監視用のシェルを起動するように設定

ログファイルは、デフォルトでは/var/mfcobol/esの下に各 ES サーバ毎に格納される。

各ノードでログファイルの次の様に格納先を変更する。 共有ディスクを有効にして、MFDS、ES サーバを全て終了しておく。





シェル casperm を実行し対話で設定する。(実行は\$COBDIR から。) ここではログファイルは実行時システムファイルの一つとして見なして応答する。 rp3440-3 # sh bin/casperm

続行する前に、Enterprise Server のシステム管理者用の英数字のユーザー ID を 手元に準備しているか確認してください。 詳細は、マニュアル「構成と管理」に記述されています。 終了するには、'q'を、続行するには、改行を押してください。: 次に続く質問に答えながら、グローバルトランザクションシステム 構成ファイル(cas. cfg)を構築します。- 詳細はマニュアル 「構成と管理」を参照してください。 エラーや警告メッセージを syslog デーモンから表示しますか (y/n)? У Enterprise Server のシステム管理者用の英数字のユーザー ID を 入力してください。:root Enterprise Server 実行時システムファイルをコピーするディレクトリを 絶対パス名で入力してください。環境変数は使用できません。 /var/mfcobol/es にコピーする場合は、RETURN キーを押してください。 /u01/mfcobol/es ファイルパーミッションを設定中です。お待ちください... Enterprise Server の構成を完了しました。 rp3440-3 #

<検証>

rp3440-3 でパッケージ ESSRV01 を立ち上げ、サービス要求などを数回行なう。

次に ES サーバ ESSRV01 が開始 (Started) 状態のまま、サーバ>コントロールの画面のE Sモニター&コントロール ボタンをクリックする。

MICRO Focus	Ent 71-3 1734	terpr פון לפו 104 1 04	ise : 1.00 12.16.1	Serve . 183)	r Ac	Iminis	trat	ion	> ESSR'	√01 > ∄
Home	Status MDS0	Status MDS0000I OK								
アクション 保存 海中	Serve	r ESSR	V01 (開始」						
1度日 インボート すべて削除 シャットダウン	サーバー… リスナー (2) サービス (3) ハンドラ (2) バッケージ (1)									
構成 オブション	ESモニター&ニントロール									
ユーザ 追加	5サー	ビス実行	テノロセ	:2						1
更新	Count	Туре	PID	TR Count	State	Executing	Start Time	Local Time	Duration	
表示	1	Normal	23969	10	Idle					
ディレクトリ	2	Normal	23968	0	Idle		-		-	
統計	3	Normal	23967	0	Idle		-	-		
セッション 4 Normal 23966 0 Idle										
シャーナル	5	Normal	23965	0	Idle		-	+		
ヘルブ このページ Contents	Auxilia	iry proc	esses							L





表示された Server Information の画面からログを参照する。



Log、C/x(現用ログ)の内容

ESSRV01			
System Log: \$TXRFDI	R/console.log		
060328 11155543		CASCD01001	ES Daemon Initialized (Ver CAS 3.0.12) 11:15:55
060328 11155643		CASCD01201	Server manager created for ES ESSRV01, process-id = 06141 11:15:
060328 11155646	6141 ESSRV01	CASS100001	Server manager initialization started 11:15:56
060328 11155847	6141 ESSRV01	CASS140051	Retrieving ES configuration from MFDS (127.0.0.1:86) 11:15:58
060328 11155953		CASCD10201	JCP service process created for region ESSRV01, process-id = 061
060328 11155954		CASCD01271	SEP 0001 created for ES ESSRV01, process-id = 06143 11:15:59
060328 11155956	6142 ESSRV01	CASJC00011	Journal control initialized 11:15:59
060328 11155959	6143 ESSRV01	CASSI15001	SEP initialization started 11:15:59
060328 11155961		CASCD01271	SEP 0002 created for ES ESSRV01, process-id = 06144 11:15:59
060328 11155962	6143 ESSRV01	CASSI16001	SEP initialization completed successfully 11:15:59
060328 11155967	6144 ESSRV01	CASSI15001	SEP initialization started 11:15:59
060328 11155969		CASCD01271	SEP 0003 created for ES ESSRV01, process-id = 06145 11:15:59

次に rp3440-3 の MFDS のプロセスを kill する。

kill -9 <mfds32のpid>

すると、ES 管理シェル mf.sh の monitor)が障害を検知しフェイルオーバーが開始する。 そして rp3440-4 でパッケージ ESSRV01 が開始される。

```
同様に Server Information の画面からログを参照する。
Log、B(1世代前のログ)の内容
```

ESSRV01			
System Log: \$	TXRFDIR/console.bak		
060328 111555	43	CASCD01001 ES Daemon Initialized (Ver CAS 3.0.12) 11:15:55	
060328 111556	43	CASCD01201 Server manager created for ES ESSRV01, process-id = 06141	11:15:
060328 111556	46 6141 ESSRV01	CASSI00001 Server manager initialization started 11:15:56	
060328 111558	47 6141 ESSRV01	CASSI40051 Retrieving ES configuration from MFDS (127.0.0.1:86) 11:15	:58
060328 111559	53	CASCD10201 JCP service process created for region ESSRV01, process-id	= 061
060328 111559	54	CASCD01271 SEP 0001 created for ES ESSRV01, process-id = 06143 11:15:	59
060328 111559	56 6142 ESSRV01	CASJC00011 Journal control initialized 11:15:59	
060328 111559	59 6143 ESSRV01	CASSI15001 SEP initialization started 11:15:59	
060328 111559	61	CASCD01271 SEP 0002 created for ES ESSRV01, process-id = 06144 11:15:	59
060328 111559	62 6143 ESSRV01	CASSI16001 SEP initialization completed successfully 11:15:59	
060328 111559	67 6144 ESSRV01	CASSI1500I SEP initialization started 11:15:59	
060328 111559	69	CASCD01271 SEP 0003 created for ES ESSRV01, process-id = 06145 11:15:	59

<結果>

共有ディスクにログファイル格納先を変更する事でフェイルオーバー後にログを参照することができた。但し、MFDSの立ち上げ時にログファイルは世代が変わるため、フェイルオーバー後は過去ログとしての参照となる。



まとめ

本検証では下記の10パターンについて動作を確認しました。

~ - •	A Z L H		水 司 吉 语
バタージ	91 FN	アスト結果	唯認争頃
1	手動切り替え	0	SERVICEGUARD 標準機能で切り替え
2	データ LAN 両系(二重)	0	SERVICEGUARD 標準機能で切り替え
	障害		
3	業務障害	0	MDUMP ユーティリティの出力を解析するシ
			ェルを自作して切り替え
4	OS のみで待機	0	MFDS 制御、ES サーバ制御をパッケージに
			含め切り替え
5	MFDS 起動で待機	0	ES サーバ制御をパッケージに含め切り替
			え
6	DS サーバ起動で待機	×	Enterprise Server のリスナーは定義する
			IP が立ち上げ時に実在しなければ立ち上
			がらないため、このパターンは実装不可
$\overline{(7)}$	別の業務を提供したが	0	パッケージを複数に分けて切り替え
	いの未初 と 近 氏 し な が こ た 機	0	
Ô	 ディプロイパッケー	0	サカディスクにディプロイパッケージを
0		0	共有) イスクに) イフロイバックーシを
	ンの共有		配直して切り替え
9	構成リポジトリの共	0	ディプロイパッケージの共有と合わせ、
	有		共有ディスクに構成リポジトリを配置し
			て切り替え
(10)	ログファイルの共有	0	共有ディスクにログファイルの出力先
			を指定して切り替え

異常検出のパターンでは、今回マイクロフォーカス提供の mdump ユーティリティを使用 し、ステータスを監視する事で異常を検出することが出来ました。 本運用においては、ユーザプログラム又は関連ソフトウェアの監視なども含め、監視方 法や監視対象を再検討する必要があります。

待機状態のパターンでは、待機系のノードでは ES サーバをあらかじめ立ち上げておく事ができない事が分かりました。

また MFDS は開始状態でも停止状態からもフェイルオーバーは可能でした。

但し、複数のパッケージで ES サーバを立ち上げるケース等では MFDS は ES サーバの起動 前にあらかじめ起動しておく必要があるため、基本的には OS の起動時に MFDS を自動起 動するように設定することをお奨めします。

Enterprise Server システムの情報の共有化のテストではディプロイパッケージ、構成 リポジトリ、ログファイルの共有してフェイルオーバーすることを確認できました。





付録

サーノ	ヾ゙アプリケーシ	ョンソース(cbl	app01.cbl)	
	FILE-CONTROL.			
	SELECT APPFILE ASSIGN TO EXTERNAL APPFILE01			
	ORGANIZATION INDEXED RECORD KEY RKEY			
	ACCESS MODE RANDOM			
	FILE STATU			
	DATA D	DIVISION.		
	FILE	SECTION.		
	FD APPFILE.			
	01 AF-REC.			
	05 RKEY F	PIC X(6).		
	05 AF-COUNTER F	PIC 9(10).		
	WORKING-STORAGE	SECTION.		
	01 WK-DATE F	PIC X(6).		
	01 WK-TIME F	PIC X(8).		
	01 AP-STATUS	pic x(02).		
	LINKAGE S	SECTION.		
	01 LK-DATETIME F	PIC X(20).		
	01 LK-PGMNAME	PIC X(10).		
	01 LK-HOSTNAME	PIC X(20).		
	01 LK-COUNTER F	PIC 9(10).		
	PROCEDURE DIVISION			
	USING LK-DATETIME LK-PGMNAME LK-HOSTNAME LK-COUNTER.			
	PERFORM GET-DA	ATETIME.		
	MOVE 'CBLAF	PP01 ' TO LK-PGMNAM	E.	
	CALL 'getho	ostname' USING LK-HOS	TNAME BY VALUE 20.	
	OPEN I-O	APPFILE.		
	MOVE SPACE	TO RKEY.		
	READ APPEIL	LE INVALID KEY	CONT INUE.	
	MOVE AF-COL	UNTER TO LK-COUNTE	R.	
	ADD 1	TO AF-COUNTE	R.	
	REWRITE AF-REG	C INVALID KEY	CONT INUE.	
	CLOSE APPFIL	LE.		
	EXIT PROGRA	AM.		
	GET-DATETIME.			
	ACCEPT WK-DATE	FROM DATE.		
	ACCEPT WK-TIME	FROM TIME.		
	STRING WK-DATE(1	1:2) '/' WK-DATE(3:2)	'/' WK-DATE(5:2) ''	
	WK-TIME(1:2) ':' WK-TIME(3:2) ':' WK-TIME(5:2) '.'			
	WK-TIME (7	7:2) DELIMITED BY SIZ	E INTO LK-DATETIME.	





<u>クライアントアプリケーションソース (clptest. java)</u>

```
import com.microfocus.cobol.connector.spi.*
import com.microfocus.cobol.connector.cci.*;
import com.microfocus.cobol.lang.*;
public class clptest
  public static void main(String[] args)
          CobolNoTxManagedConnectionFactory mcf;
          javax.resource.cci.ConnectionFactory cxf;
          javax.resource.cci.Connection connection=null;
          iavax resource cci Interaction interaction=null:
          try{
            mcf = new CobolNoTxManagedConnectionFactory();
            mcf.setServerHost("necp2");
            mcf.setServerPort("9003");
            mcf.setTrace(new Boolean(false));
            cxf = (javax.resource.cci.ConnectionFactory) mcf.createConnectionFactory();
            connection = cxf.getConnection();
            {
              javax.resource.cci.Interaction ix =
              connection.createInteraction();
              com. microfocus. cobol. connector. cci. CobolInteractionSpec
              iSpec = new
                     com.microfocus.cobol.connector.cci.CobolInteractionSpec();
               iSpec.setFunctionName("initialize");
              javax.resource.cci.RecordFactory rf = cxf.getRecordFactory();
              iavax. resource. cci. IndexedRecord irec =
              rf.createIndexedRecord("beanArgs");
              irec.add(new Boolean(true));
              javax.resource.cci.Record orec = ix.execute(iSpec, irec);
              ix.close();
            }
            javax.resource.cci.Interaction ix = connection.createInteraction();
            CobolInteractionSpec iSpec = new CobolInteractionSpec();
            iSpec.setFunctionName("cblapp01S.CBLAPP01");
            iSpec.setArgument(0, java.lang.String.class, com.microfocus.cobol.RuntimeProperties.OUTPUT_ONLY);
            iSpec.setArgument(1, java.lang.String.class, com.microfocus.cobol.RuntimeProperties.OUTPUT_ONLY);
            iSpec.setArgument(2, java.lang.String.class, com.microfocus.cobol.RuntimeProperties.OUTPUT_ONLY);
            iSpec.setArgument(3, Long.class, com.microfocus.cobol.RuntimeProperties.OUTPUT_ONLY);
            javax.resource.cci.RecordFactory rf = cxf.getRecordFactory();
            javax.resource.cci.IndexedRecord iRec = rf.createIndexedRecord ("CBLAPP01In");
            javax.resource.cci.IndexedRecord oRec = rf.createIndexedRecord("CBLAPP010ut");
            ix.execute(iSpec, iRec, oRec);
            ix.close();
            System. out. println (oRec. get (0) + " " + oRec. get (1) + " " + oRec. get (2) + " " + oRec. get (3));
          }
          catch(Exception e) {
                    System.err.println(e);
          }
          return;
 }
}
```