CLUSTERPRO® X *for Windows*

PPガイド(JRun4)

2012.08.10
第01版

CLUSTERPRO

改版履歴

版数	改版日付	内 容
1	2012/08/10	PPガイドより分冊し、新規作成

© Copyright NEC Corporation 2008. All rights reserved.

免責事項

本書の内容は、予告なしに変更されることがあります。
日本電気株式会社は、本書の技術的もしくは編集上の間違い、欠落について、一切責任をおいません。
また、お客様が期待される効果を得るために、本書に従った導入、使用および使用効果につきましては、
お客様の責任とさせていただきます。
本書に記載されている内容の著作権は、日本電気株式会社に帰属します。本書の内容の一部または全部
を日本電気株式会社の許諾なしに複製、改変、および翻訳することは禁止されています。

商標情報

CLUSTERPRO® X は日本電気株式会社の登録商標です。
Intel、Pentium、Xeonは、Intel Corporationの登録商標または商標です。
Microsoft、Windowsは、米国Microsoft Corporationの米国およびその他の国における登録商標です。
本書に記載されたその他の製品名および標語は、各社の商標または登録商標です。
Oracle Parallel Serverは米国オラクル社の商標です。
その他のシステム名、社名、製品名等はそれぞれの会社の商標及び登録商標です。

目次

はじめに

対象読者と目的

『CLUSTERPRO® PPガイド』は、クラスタシステムに関して、システムを構築する管理者、およびユーザサポートを行うシステムエンジニア、保守員を対象にしています。
本書では、CLUSTERPRO環境下での動作確認が取れたソフトウェアをご紹介しています。ここでご紹介するソフトウェアや設定例は、あくまで参考情報としてご提供するものであり、各ソフトウェアの 動作保証をするものではありません。

適用範囲

本書は、 以下の製品を対象としています。
CLUSTERPRO X 3.1 for Windows
CLUSTERPRO X 3.0 for Windows
CLUSTERPRO X 2.1 for Windows
CLUSTERPRO X 2.0 for Windows
CLUSTERPRO X 1.0 for Windows

CLUSTERPRO マニュアル体系

CLUSTERPRO のマニュアルは、以下の 4 つに分類されます。各ガイドのタイトルと役割を以下に示します。

『CLUSTERPRO X スタートアップガイド』(Getting Started Guide)

CLUSTERPRO を使用するユーザを対象読者とし、製品概要、動作環境、アップデート情報、既知の問題などについて記載します。

『CLUSTERPRO X インストール & 設定ガイド』(Install and Configuration Guide)

CLUSTERPRO を使用したクラスタ システムの導入を行うシステム エンジニアと、クラスタシステム導入後の保守・運用を行うシステム管理者を対象読者とし、CLUSTERPRO を使用したクラスタ システム導入から運用開始前までに必須の事項について説明します。実際にクラスタ システムを導入する際の順番に則して、CLUSTERPRO を使用したクラスタ システムの設計方法、CLUSTERPRO のインストールと設定手順、設定後の確認、運用開始前の評価方法について説明します。

『CLUSTERPRO X リファレンス ガイド』(Reference Guide)

管理者、およびCLUSTERPRO を使用したクラスタ システムの導入を行うシステム エンジニアを対象とし、CLUSTERPRO の運用手順、各モジュールの機能説明、メンテナンス関連情報およびトラブルシューティング情報等を記載します。『インストール & 設定ガイド』を補完する役割を持ちます。

『CLUSTERPRO X 統合WebManager 管理者ガイド』(Integrated WebManager Administrator's Guide)

CLUSTERPRO を使用したクラスタシステムを CLUSTERPRO 統合WebManager で管理するシステム管理者、および統合WebManager の導入を行うシステムエンジニアを対象読者とし、統合WebManager を使用したクラスタシステム導入時に必須の事項について、実際の手順に則して詳細を説明します。

本書の表記規則

本書では、「注」および「重要」を以下のように表記します。

注: は、重要ではあるがデータ損失やシステムおよび機器の損傷には関連しない情報を表します。

重要: は、データ損失やシステムおよび機器の損傷を回避するために必要な情報を表します。

関連情報: は、参照先の情報の場所を表します。

また、本書では以下の表記法を使用します。

表記	使用方法	例
[ ] 角かっこ	コマンド名の前後 画面に表示される語 (ダイアログ ボックス、メニューなど) の前後	[スタート] をクリックします。 [プロパティ] ダイアログ ボックス
コマンドライン中の [ ] 角かっこ	かっこ内の値の指定が省略可能であることを示します。	`clpstat -s[-h host_name]`
モノスペースフォント (courier)	コマンド ライン、関数、パラメータ	`clpstat -s`
モノスペースフォント**太字** (courier)	ユーザが実際にコマンドプロンプトから入力する値を示します。	以下を入力します。 **`clpcl -s -a`**
モノスペースフォント (courier) 斜体	ユーザが有効な値に置き換えて入力する項目	`clpstat -s [-h host_name]`

最新情報の入手先

最新の製品情報については、以下のWebサイトを参照してください。

http://www.nec.co.jp/clusterpro

第 1 章　　JRun4

機能概要

JRun は、Web サーバを拡張して、Java サーブレット、Java Server Pages(JSP)の実行を可能にします。

JRun が提供する高機能・高性能で実績のあるサーブレットコンテナを、アプリケーションサーバと連携させることによって、短期間で高可用、かつ高信頼な Web システムを構築することが可能となります。

機能範囲

JRun4 は、クラスタ環境においてもシングルサーバと同様に動作します。

注：CLUSTERPRO でサポートする JRun は、JRun 4 Updater 6 以降です。

動作環境

本ガイドで説明する内容は、以下の環境を前提としたものです。

- OS
  - Windows Server 2003 Standard SP1
- メモリ
  - 2048MByte
- 使用JDK
  - J2RE1.4.2_08
- Webサーバ
  - Apache HTTP Server 1.3.33 ／2.0.54
  - Microsoft IIS 6.0
- CLUSTERPROの環境設定
  - JRun4 は C:ドライブの¥JRun4 にインストール
  - ミラーディスクを使用(H:ドライブとして作成)

ライセンス

JRun のライセンスは現用系、待機系に関わらず、JRun を実行する可能性のある全てのマシンが搭載している物理 CPU 数の合計に依存します。
例えば CPU を 2 個搭載した 2 台のマシンで冗長システムを構築した場合、常に動作している CPU は現用働系の 2 個で、待機系 CPU も 2 個となります。しかしシステム全体では物理 CPU が合計 4 個存在しますので、ライセンスも 4 個となります。
デュアルコア CPU などのように、物理 CPU 上に複数のコアが搭載されている場合は、物理 CPU の個数でお考えください。

JRunのインストール手順

全サーバ上のローカルパーティションに対しインストールを行います。

インストール方法の詳細については、「JRun インストールガイド」を参照して下さい。

インストール時の注意

- **CLUSTERPROと連携させる場合は、「Windows サービスとしてJRunをインストール」をチェックして、サービスとしてインストールしてください。**
  詳細については、JRunインストールガイド 31 ページの「Windows サービスに関する検討事項」を参照してください。

インストール前に JRun が使用するポート番号が使用中でないかを確認してください。
JRun サーバが default で使用するポート番号は以下の通りです。

サーバ	JNDIポート	HTTPポート	プロキシポート
admin	2910	8000	51001
default	2908	8100	51000
sample	2918	8200	51010

使用するポートの詳細については JRun 管理者ガイド 5 ページ 「JRunポート」を参照してください。

インストール終了後、管理ツールの「サービス」で以下の設定を行ってください。

- サービス項目より、「Macromedia JRun Admin Server」を選択し、「プロパティ」を表示させてください。
- 「全般」タブの「スタートアップの種類」を「手動」に設定してください。
- 同様に「Macromedia JRun default Server」も「手動」に設定してください。

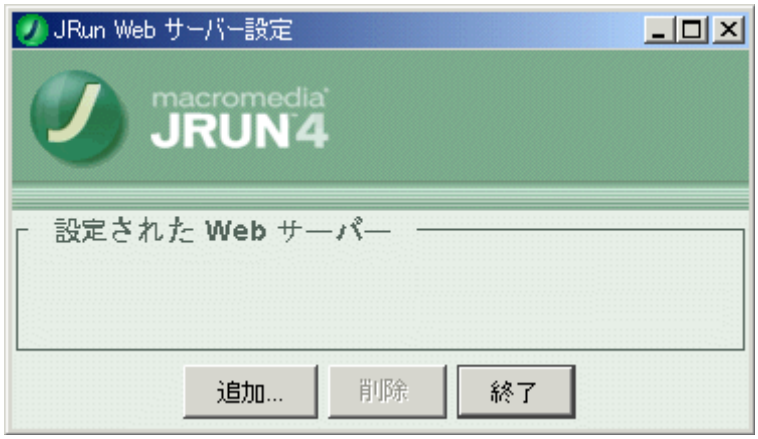
Updater 6 の適用

1. 以下のAdobeのJRun Updaterページから、Updater6 のWindows用日本語アップデータ（jrun4-win-ja-updater.exe)をダウンロードしてください。
   http://www.adobe.com/jp/support/jrun/downloads_updates.html
   また、アップデータの詳細については、 同ページにある JRun 4 アップデータ 6 リリースノート を参照してください。
2. JDK1.3.1 あるいはそれ以降のデフォルト JDK がシステム上にインストールされていることを確認してください (インストーラが必要とします)。
3. すべての JRun 4 サービスを停止します。すべての JRun プロセスを停止できなかった場合、インストレーション処理が失敗することがあります。
   まだ稼動している JRun サービスがあるかどうかは、次の手順で確認できます。
   - Windows では、JRun サーバが Windows サービスとして実行されている場合は、サービスコントロールパネルを使用してそれらのサービスを停止させてください。
4. JRun に接続しているすべての Web サーバーを停止します。IIS を JRun の Web サーバーとして使用している場合は、World Wide Web Publishing サービスをサービスコントロールパネルで停止してください。
5. JRun 4 のインストールルートディレクトリを書き留めておきます(“c:¥jrun4” など)。
6. Windows 2000、Window XP および Windows 2003 では、アップデータによって置き換えられる {jrun.home}/bin 内のいくつかの DLL がロックされることを避けるために Windows Management Instrumentation サービスを停止させます。
7. jrun4-win-ja-updater.exe を実行します。
8. インストールウィザードの指示に従ってインストール作業を完了します。
9. インストールを完了後、JRun サービスを再起動します。
10. Windows Management Instrumentation サービスをコントロールパネルで開始します。
11. JRun サーバを再起動します。

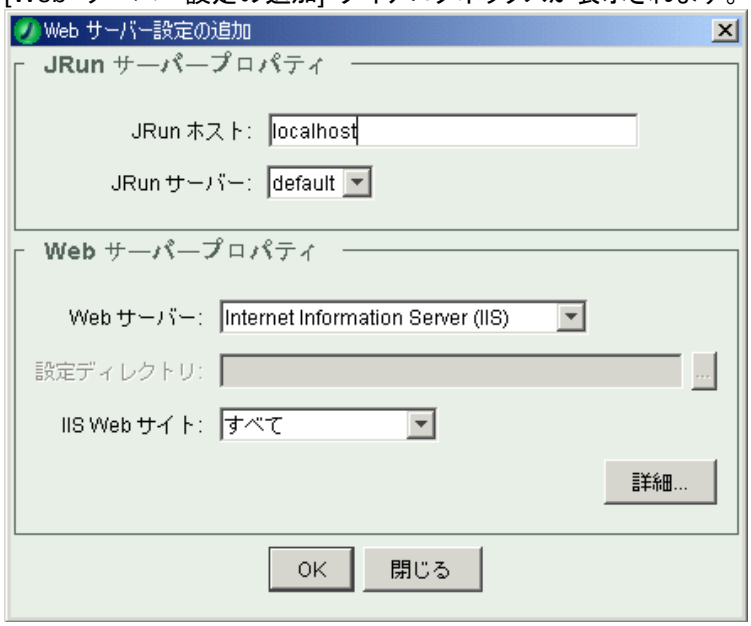
外部Webサーバ設定（IISの設定例）

1. 次のいずれかの方法で Web サーバー設定ツールを起動します。
   - (Windows) [スタート] → [プログラム] → [Macromedia JRun 4] → [Web サーバー設定ツール] を選択します。
   - コマンドラインから <JRun のルートディレクトリ>/lib にディレクトリを変更し、次のように入力します。
     javaw -jar wsconfig.jar
     メモ:
     <JRun のルートディレクトリ> は、JRun 4 がインストールされているディレクトリです。
     コマンドラインから Web サーバー設定ツールを実行する手順については、JRun インストールガイド 43 ページの「コマンドラインオプション」を参照してください。

   JRun Web サーバー設定ツールのウィンドウが表示されます。



2. [追加] をクリックします。
   [Web サーバー設定の追加] ダイアログボックスが表示されます。



3. [JRun サーバー] リストボックスで、設定する JRun サーバー名を選択します。
4. [Web サーバープロパティ] 領域で、Web サーバー情報を入力して [OK] をクリックします。
   特定の Web サーバーに関する JRun の設定手順については、JRun インストールガイドの該当するセクションを参照してください。

   注意:
   JRun に付属の JRun Web サーバ(JWS)は、実運用環境での**使用を推奨していません。**
   必ず外部 Web サーバと組み合わせて使用してください。

デプロイディレクトリの設定

クラスタ構成のマシン上に Web アプリケーションを配置する場合には、以下の2種類の方法があります。

- 各マシンのローカルディスク上に配置する。
- ミラーリングされたディレクトリ上に配置する。

Web アプリケーションをミラーディスク上に配置した場合、管理すべき Web アプリケーションが単一であるため、管理が容易になるメリットがあります。

一方、Web アプリケーションを置換する場合、JRun を停止する必要があるため、可用性が低くなるデメリットがあります。

Web アプリケーションをローカルディスク上に配置した場合、管理すべき Web アプリケーションが複数になり、管理が比較的複雑になります。

一方、Web アプリケーションを置換する場合、待機系の Web アプリケーションを置換したのち、フェイルオーバさせれば、すぐに切り替わるため、ミラーディスクの場合と比較して可用性の向上が図れます。

Web アプリケーションを配置するディレクトリのことを、デプロイディレクトリと言います。

以下、ミラーディスク上にデプロイディレクトリを設定する手順を説明します。

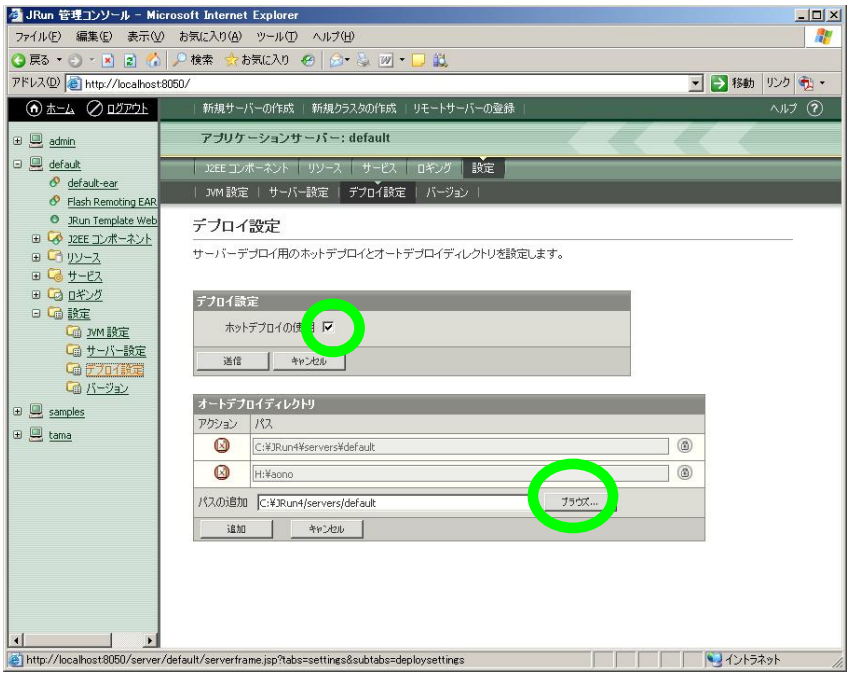
JMC→サーバ(例：default)→設定→デプロイ設定

オートデプロイディレクトリにパスを追加する。

1. ブラウザで以下のURLにアクセスしてJMC(管理コンソール)を起動する。
   ユーザー名とパスワードを入力してログインを行なう。
   http://*machine_name*:8000
   JRun管理コンソールの詳細については、JRun入門 83 ページの「JMCを開始するには」を参照してください。

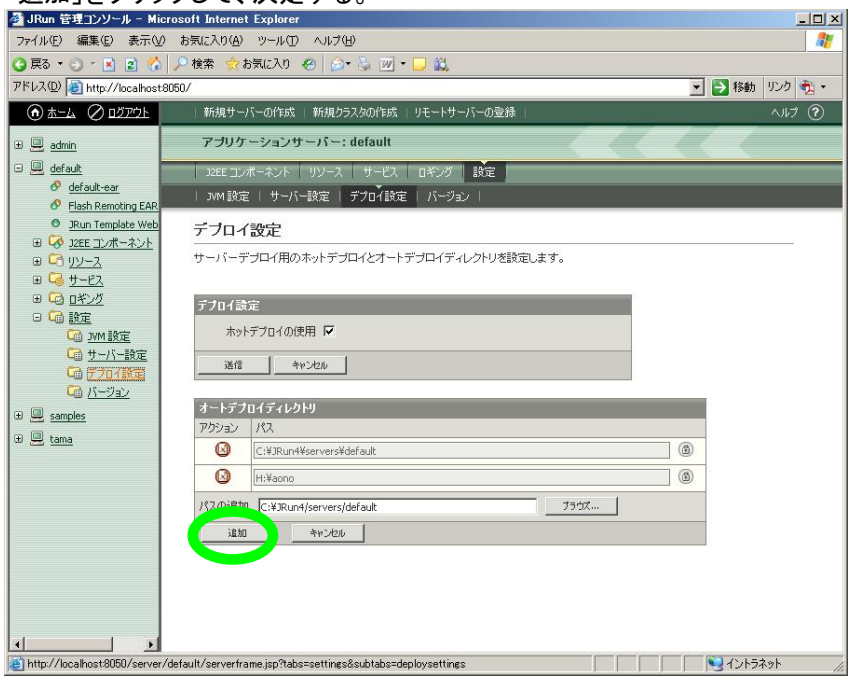
2. デプロイ設定するサーバを選択し、「設定」→「デプロイ設定」の順で開く。

3. 「デプロイ設定」ウィンドウでデプロイディレクトリを設定する。
「ホットデプロイの使用」のチェックを外した後、「ブラウズ」をクリックする。



4. オートデプロイディレクトリを参照後、「開く」をクリックして追加するパスを決定する。

5. 「追加」をクリックして、決定する。



6. 設定したディレクトリに Web アプリケーションを配備してください。

以上で、ミラーディスクへのデプロイディレクトリ設定は終了です。

参考例 - CLUSTERPROでJRunサービスのストール監視を行う場合の設定

設定手順の概要は以下の通りです。

1. フェイルオーバグループに JRun のサービスを追加する。
2. JRun のサービス監視を追加する。
3. JRunサービスのストール確認用スクリプトファイルを設定する。
   - スクリプトファイル：jrunwatch.bat
   - 文字列比較クラスファイル：Check.class
   - 比較用文字列格納ファイル：get.dat
   - 確認用 JSP ファイル：alive.jsp
4. アプリケーション監視リソースを追加する。

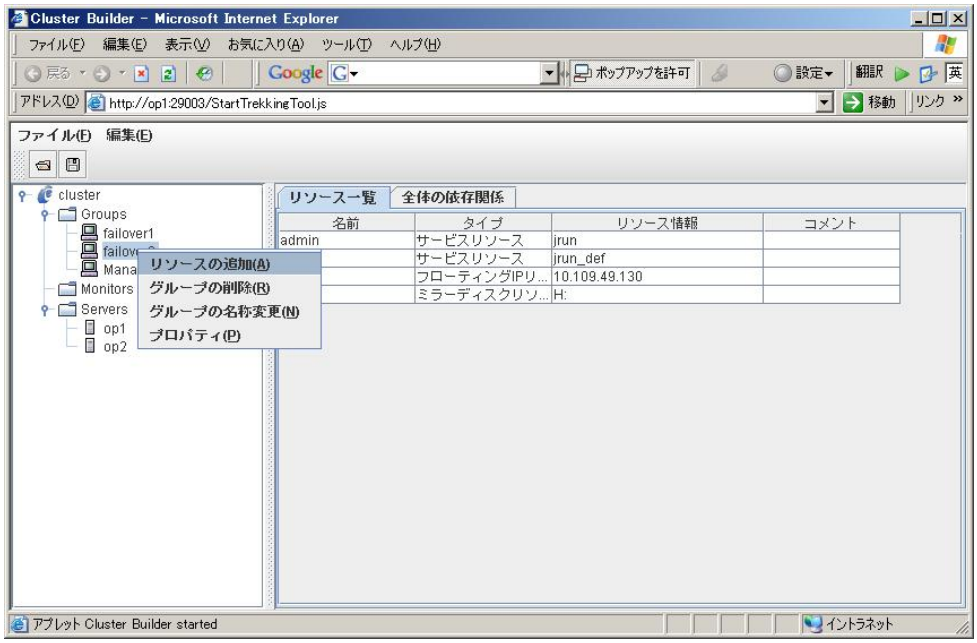
なお、設定内容の詳細と運用については、CLUSTERPRO のリファレンスガイドを参照願います。

設定手順

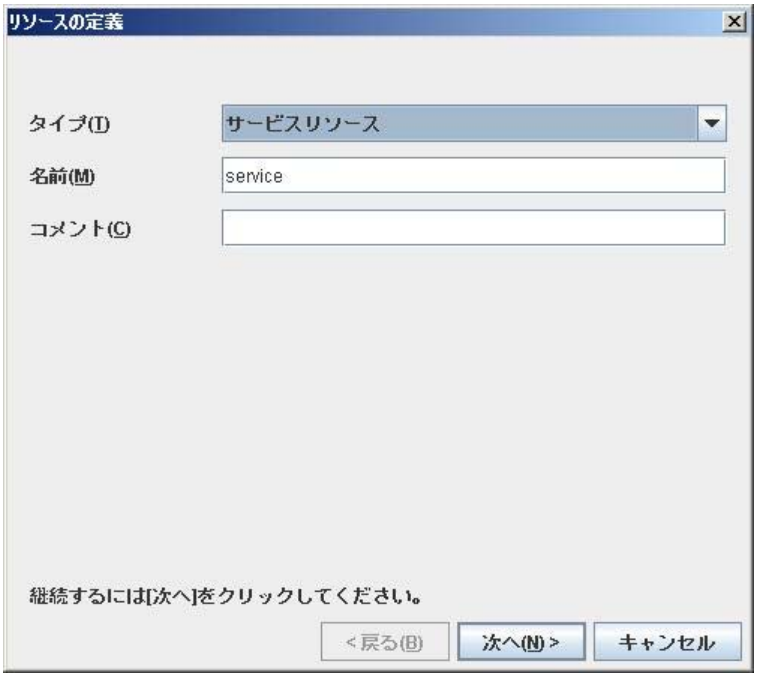
1. 作成済みのフェイルオーバグループに JRun のサービスを追加する。

Cluster Builder を起動する。

JRun を追加するフェイルオーバグループに対して「リソースの追加」を選択する。

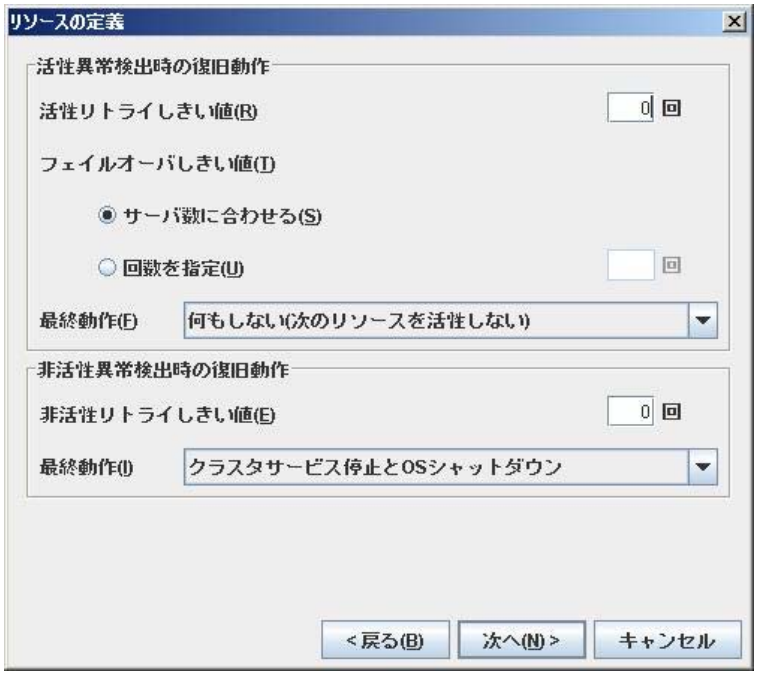


「サービスリソース」を選択し、表示に使用する名前を入力する。



起動するサービス名に Windows サービスで指定したjrunサーバのサービス名を設定する。
(図では ”default” を設定)

サービス内容の設定を行う。

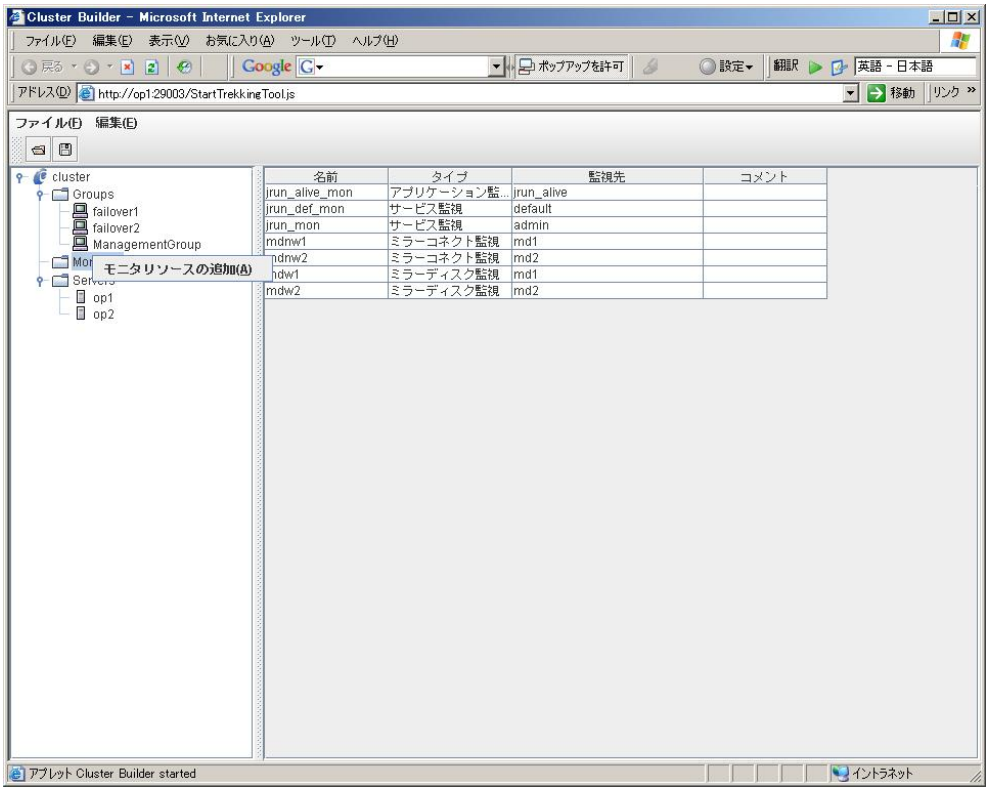


依存関係の設定を行う。

2. JRun のサービス監視を追加する。

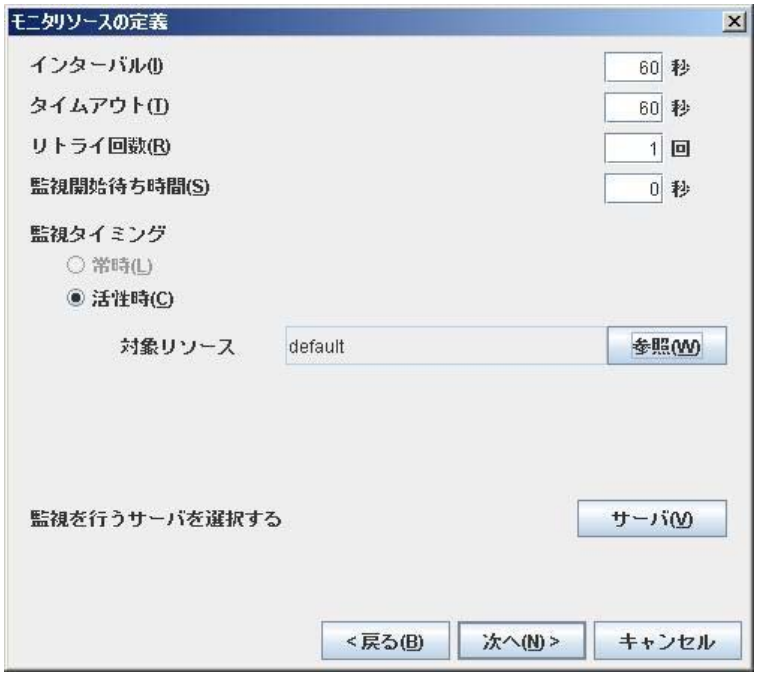
Cluster Builder を起動する。

「モニタリソースの追加」を選択する。



「タイプ」から「サービス監視」を選択し、表示名を設定する。

「対象リソース」に監視したいサービスを選択する。



「回復対象」に、異常検出時に回復させたいサービスを選択する。

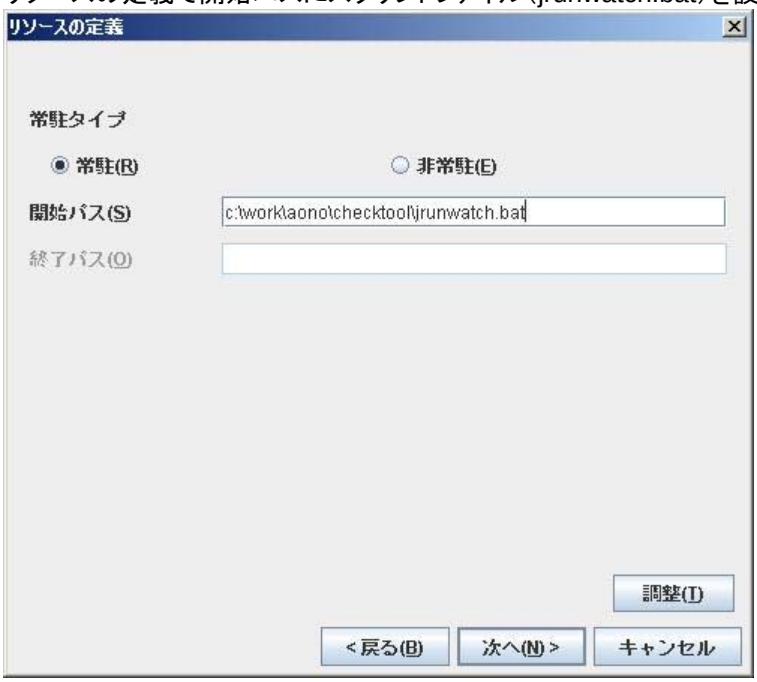
3. JRun サービスのストール確認用スクリプトファイルを設定する。

Cluster Builder を起動する。

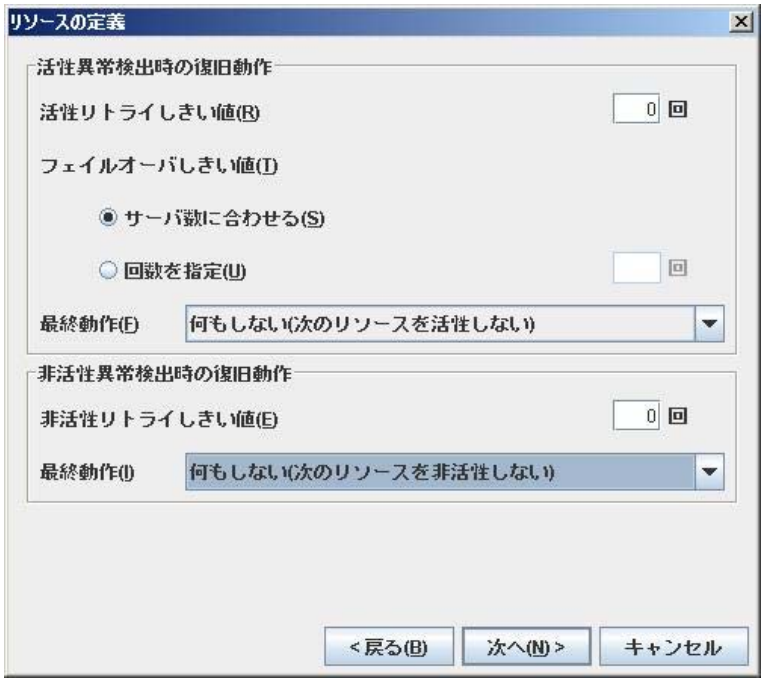
確認に使用するスクリプトファイル（jrunwatch.bat）をホストの任意の場所にコピーする。
（例では”c:¥work¥aono¥checktool¥” 配下）

アプリケーションリソースの追加を選択する。
名前は任意の表示名(例では”jrun_alive”)

リソースの定義で開始パスにスクリプトファイル（jrunwatch.bat）を設定する。



非活性異常時の復旧動作を設定する。



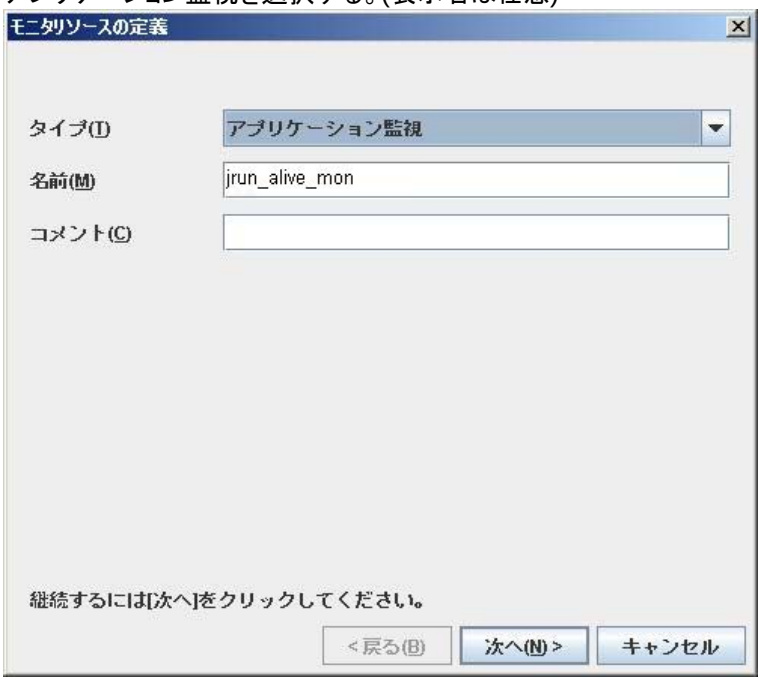
依存関係を設定する。

4. アプリケーション監視リソースを追加する。

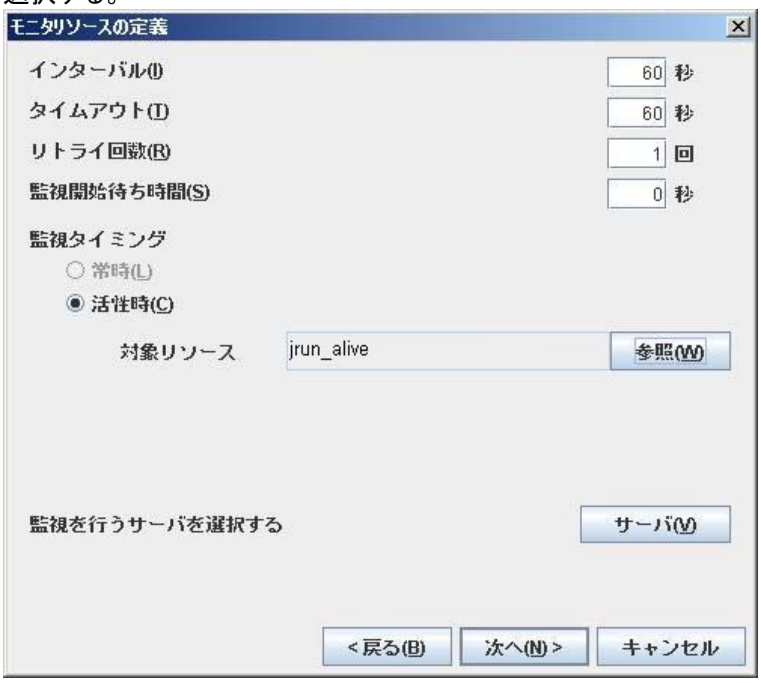
Cluster Builder を起動する。

モニタリソースの追加を選択する。

アプリケーション監視を選択する。(表示名は任意)



「対象リソース」に監視したいアプリケーションリソース(3で指定したアプリケーションリソース)を選択する。



「回復対象」に回復させたいフェイルオーバグループ（定義したフェイルオーバグループ）を設定する。

運用手順

1. アプリケーションリソースを起動すると、監視用スクリプト（jrunwatch.bat）が alive.jsp を一定間隔で繰り返し発行する。
2. JRun サービスがストールして応答を返せない状態になると、アプリケーションリソースが異常終了する。
3. アプリケーションリソースの異常終了をアプリケーション監視リソースが検出することで、フェイルオーバが発生する。

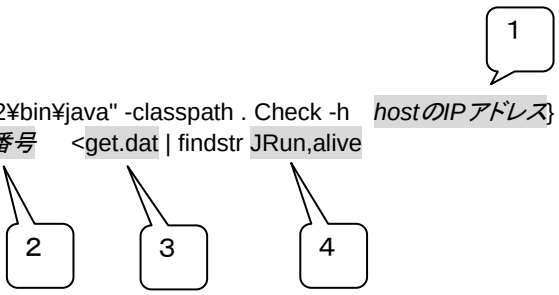
スクリプトサンプル

内部の host 名／フォルダ名などは、実際の環境に合わせて読み替えてください。

jrunwatch.bat

```
echo off

:LOOP1
call "C:¥Program Files¥Java¥j2re1.4.2_12¥bin¥java" -classpath . Check -h  hostのIPアドレス}
                                        -p port番号    <get.dat | findstr JRun,alive
if "%errorlevel%" == "0" (
    echo "OK"
     sleep 10
    goto LOOP1
) else (
    echo "NG"
     echo "%errorlevel%"
)
```



パラメータの説明

1：ホストの IP アドレス

2：ホストのポート番号

3：発行する HTTP コマンド

4：比較する文字列

get.dat

```
GET /alive.jsp   HTTP/1.1
```

jrunwatch.bat がリクエストする内容(リクエストURI)を記述する。
リクエストURIについては、JRun プログラマーズガイド 121ページを
参照願います。

sleep.exe

sleep.exe は Microsoft が提供しているリソースキットから取得してください。
Windows 2003 の Resource Kit の URL は以下の通りです。(2007/7/20 現在)
http://www.microsoft.com/downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&DisplayLang=en&displaylang=en

Check.class

```
import java.io.*;
import java.net.*;

public class Check
{
    private static int sendPort = 8100;
    private static String sendAddress = "";
    private static final int MAX_BUFFER_LEN = 1460;
    private static int buffersize = MAX_BUFFER_LEN;
    private static boolean closeflag = false;
    private static int delay = 0;

    public static void main(String argv[]){
```

```
        if(argv.length == 0){
            System.err.println("Usage: java Check [-h host] [-p port][-c][-t sec][-b buflen]");
            System.err.println("       -c:close");
            System.err.println("       -t:delay time of writing,unit:Sec.");
            System.err.println("       -b:write buffer size,unit:bytes");
            System.err.println("       -c:close");
            System.err.println("STDIN :HTTP Request Data");
            System.err.println("STDOUT:HTTP Response Data");
            System.err.println();
            System.exit(1);
        }

        for(int i = 0; i < argv.length ; i++){
            if(argv[i].equals("-h") == true){
                i++;
                sendAddress = argv[i];
                continue;
            }
            if(argv[i].equals("-p") == true){
                i++;
                sendPort = Integer.parseInt(argv[i]);
                continue;
            }
            if(argv[i].equals("-c") == true){
                closeflag = true;
                continue;
            }
            if(argv[i].equals("-t") == true){
                i++;
                delay = Integer.parseInt(argv[i]);
                continue;
            }
            if(argv[i].equals("-b") == true){
                i++;
                buffersize = Integer.parseInt(argv[i]);
                continue;
            }
        }

        String line;
        StringBuffer revline;
        int len;

        Socket client2 = null;

        InputStream is = null;
        OutputStream os = null;

        long d1 = 0;
        long d2 = 0;
        try{
            System.err.println("sendAdderss=" + sendAddress);
            System.err.println("sendPort     =" + sendPort);
            System.err.println("delay        =" + delay);
            System.err.println("buffer size=" + buffersize);
            System.err.println("closing after sending 1packet.=" + closeflag);

            System.err.println("RequestStart");
            d1 = java.lang.System.currentTimeMillis();

            client2 = new Socket(sendAddress, sendPort);

            is = client2.getInputStream();
            os = client2.getOutputStream();
```

```
            byte buffer[] = new byte[buffersize];

            boolean first = true;
            while((len = System.in.read(buffer,0,buffersize)) >= 0){
                os.write(buffer,0,len);
                os.flush();

                if(delay != 0){
                    try{
                        Thread.sleep(delay * 1000);
                    }catch(Exception ee){
                    }
                }
                System.err.write(buffer,0,len);
            }

            if(closeflag == false){
                while((len = is.read(buffer,0,buffersize)) >= 0){
                    System.out.write(buffer,0,len);
                }
            }
        }catch(InterruptedIOException e){
            System.err.println(e);
        }catch(IOException e){
            System.err.println(e);
        }finally{
            try{
                if(client2 !=null){
                    client2.close();
                    client2 = null;
                }
            }catch(Exception e2){
            }

            d2 = java.lang.System.currentTimeMillis();
            System.err.println("time(sec):" + (d2 - d1) / 1000.0);
        }
    }
}
```

alive.jsp

```
<%@ page import="java.util.Date" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=shift_jis">
<title>Check, Alive.</title>
</head>

<body>
<%      out.print("JRun,alive!<BR>"); %>
</body>
</html>
```