CLUSTERPRO® X *for Windows*

PPガイド (仮想化ソフト)

2010.10.1
第3版

CLUSTERPRO

改版履歴

版数	改版日付	内 容
1	2009/04/14	・ 新規作成
2	2009/10/20	・ Hyper-V 2.0に対応 ・ サンプルスクリプトを修正 ・ 注意事項を追記
3	2010/10/1	・ CLUSTERPRO X 3.0に対応

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。
AMD, AMD VirtualizationはAdvanced Micro Devices, Incの商標です。
Intel, Pentium, Xeon, Intel VTは、Intel Corporationの登録商標または商標です。
Microsoft, Windows, Windows Server, Hyper-V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。
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、下記のバージョンのCLUSTERPROを対象としています。
- CLUSTERPRO X 2.0/2.1 for Windows
- CLUSTERPRO X 2.0/2.1 for Linux
- CLUSTERPRO X 3.0 for Windows
- CLUSTERPRO X 3.0 for Linux

本書の構成

セクション Ⅰ　仮想化ソフト

第 1 章　「Hyper-V」: Hyper-VとCLUSTERPROの連携手順および注意事項について説明します。

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、注意すべき事項、重要な事項および関連情報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)	パス名、コマンド ライン、システムからの出力 (メッセージ、プロンプトなど)、ディレクトリ、ファイル名、関数、パラメータ	`C:¥Program Files¥CLUSTERPRO`
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/

セクション Ⅰ　仮想化ソフト

このセクションでは、仮想化ソフトと CLUSTERPRO を連携する場合に必要となる設定および注意事項について説明します。

第 1 章　Hyper-V

本章では、Hyper-V と CLUSTERPRO の連携手順および注意事項について説明します。

本章で説明する項目は以下のとおりです。

本章で用いる用語

本章で用いる用語について説明します。

用語	説明
物理マシン	Hyper-Vが動作しているサーバです。
ホストOS	物理マシンにインストールされているOS、つまりWindows Server 2008およびWindows Server 2008 R2です。
仮想マシン	物理マシン上に作成される仮想的なサーバまたはクライアントです。
ゲストOS	仮想マシンにインストールされているOSです。
統合サービス	仮想マシンからハイパーバイザを呼び出すための追加ソフトです。後述のホストOS間クラスタを構築する場合、必ず仮想マシンに統合サービスをインストールしてください。
フェイルオーバグループ	CLUSTERPROのグループで、タイプが [フェイルオーバ] のグループを本章ではフェイルオーバグループとします。
仮想マシングループ	CLUSTERPROのグループで、タイプが [仮想マシン] のグループを本章では仮想マシングループとします。

機能概要

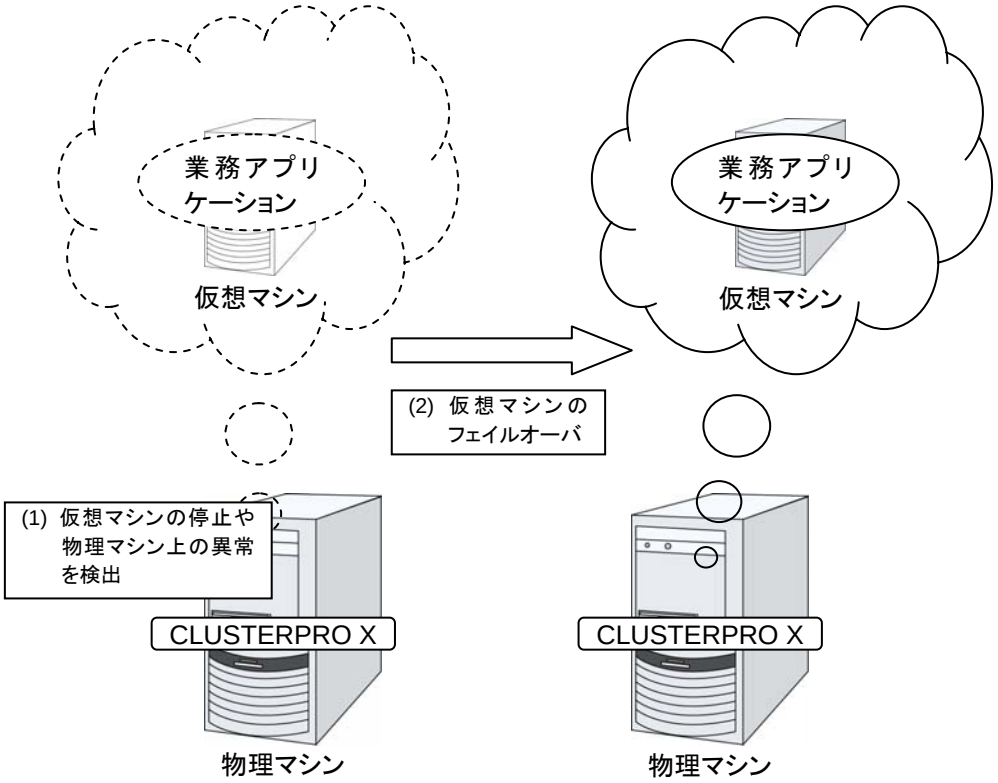
Hyper-V と CLUSTERPRO を組み合わせることで、下記構成のクラスタを構築することができます。

ホスト OS 間クラスタ

CLUSTERPRO X をホスト OS にインストールし、物理マシン同士でクラスタリングを行います。通常の業務アプリケーションのフェイルオーバのみならず、ゲスト OS をフェイルオーバさせることができます。

本章ではゲスト OS をフェイルオーバ対象としたクラスタの構築手順について説明しています。ゲスト OS の起動/停止/監視は CLUSTERPRO で行います。ゲスト OS 上のアプリケーションは、クラスタを意識する必要はありません。仮想マシンの構成ファイルはミラーディスクまたは共有ディスク上に保存します。

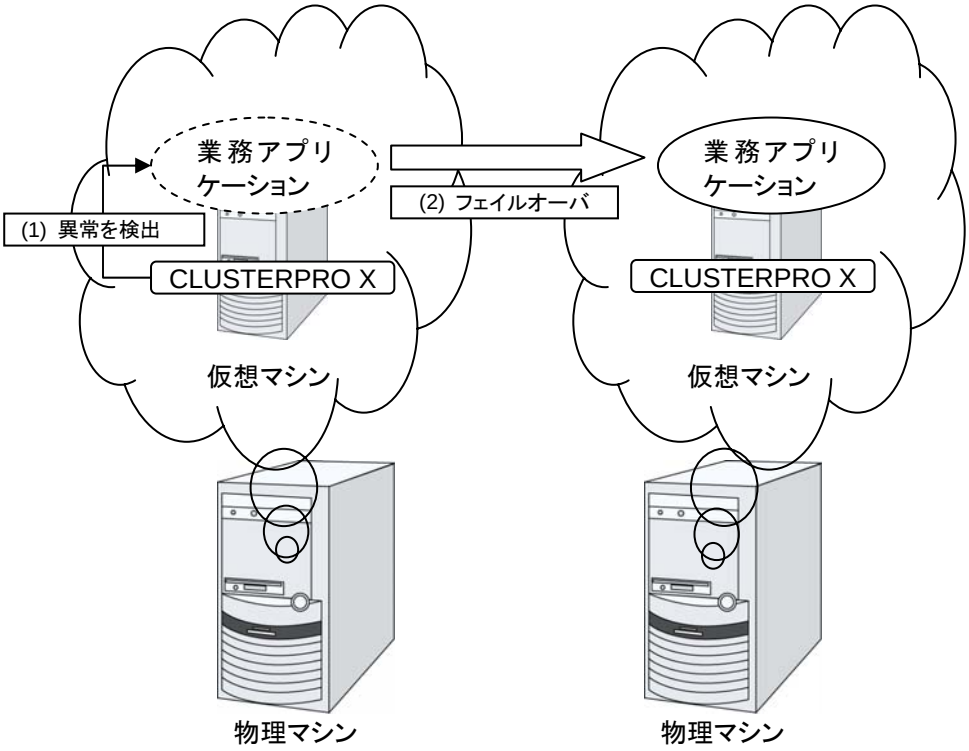
計画停止時には、Windows Server 2008 Failover Cluster (WSFC) で提供されている Quick Migration と同様、仮想マシンをシャットダウンすることなく、ゲスト OS を別の物理マシンへ移動させることができます。



ゲスト OS 間クラスタ

CLUSTERPRO X をゲスト OS にインストールし、仮想マシン同士でクラスタリングを行います。ゲスト OS 上で起動する業務アプリケーションをフェイルオーバ対象とすることで、ゲスト OS 上の業務の可用性を高めることができます。

業務アプリケーションをフェイルオーバ対象とした通常のクラスタと同様の機能を提供します。



ゲスト OS - ホスト OS 間連携

CLUSTERPRO X または CLUSTERPRO X SingleServerSafe をゲスト OS およびホスト OS にインストールし、ゲスト OS 上のクラスタから、ホスト OS 上のクラスタにフェイルオーバなどの要求を送ることができます。また、その逆にホスト OS のクラスタからゲスト OS のクラスタに要求を送ることもできます。

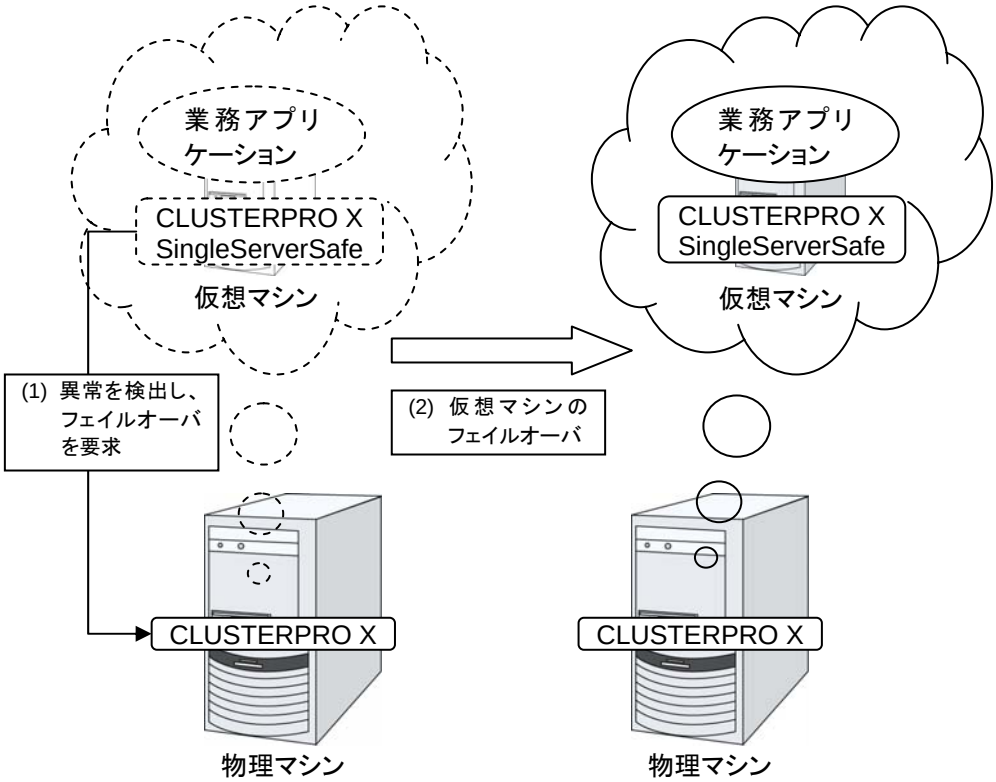
CLUSTERPRO X 2.0/2.1 の場合、要求の送信に clptrnreq コマンドを使います。

CLUSTERPRO X 3.0 の場合、要求の送信に clptrnreq コマンドまたは clprexec コマンドを使うことができます。

例えば、下記のような構成の場合、ゲスト OS 側の CLUSTERPRO X SingleServerSafe の監視リソースで障害を検出したことを契機に、ホスト OS 側の CLUSTERPRO X にフェイルオーバの要求を送信し、正常な物理マシン側にゲスト OS を移動させることができます。ゲスト OS が Linux であっても、ホスト OS 上の CLUSTERPRO X に要求を送信することができます。

この機能を使うためには、下記バージョン以降の CLUSTERPRO をインストールする必要があります。

- CLUSTERPRO X 2.0/2.1 for Windows: 内部バージョン 10.02 以降
- CLUSTERPRO X 2.0/2.1 for Linux: rpm バージョン 2.0.2-1 以降
- CLUSTERPRO X 3.0 for Windows: 内部バージョン 11.00 以降
- CLUSTERPRO X 3.0 for Linux: rpm バージョン 3.0.0-1 以降



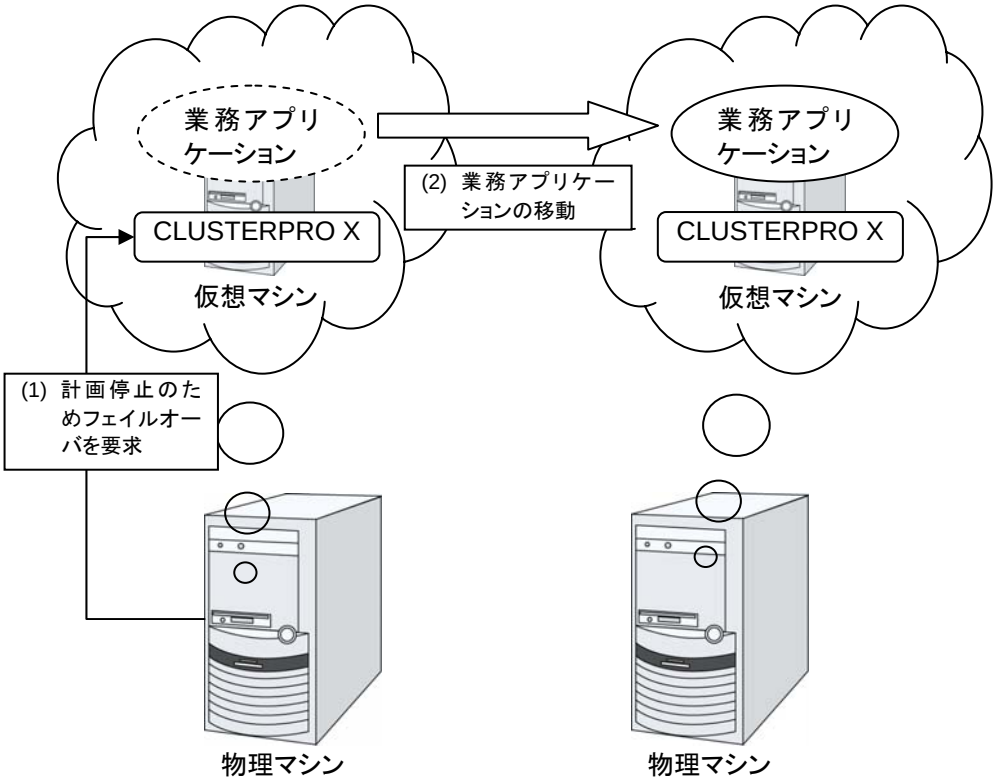
外部連携 (CLUSTERPRO X 3.0 の場合)

CLUSTERPRO X 3.0 では CLUSTERPRO がインストールされていないサーバから、クラスタにフェイルオーバなどの要求を送ることもできます。要求の送信には clprexec コマンドを使います。

例えば、下記のような構成の場合、CLUSTERPRO がインストールされてないホスト OS から、ゲスト OS 間クラスタに対して、計画停止時などにグループ移動を要求することができます。

この機能を使うためには、下記バージョン以降の CLUSTERPRO をインストールする必要があります。

- CLUSTERPRO X 3.0 for Windows: 内部バージョン 11.00 以降
- CLUSTERPRO X 3.0 for Linux: rpm バージョン 3.0.0-1 以降



動作環境

- Hyper-V は Windows Server 2008 Standard, Enterprise, Datacenter の x64 Edition ならびに Windows Server 2008 R2 Standard, Enterprise, Datacenter でのみサポートされています。

- Hyper-V の機能を有効にするためには CPU が Intel Virtualization Technology (Intel VT) または AMD Virtualization (AMD-V) に対応している必要があります。また、データ実行保護 (DEP) が有効になっている必要があります。これらの機能が有効になっていることを BIOS の設定画面で確認してください。

注意事項

ホスト OS 間クラスタの注意事項 (CLUSTERPRO X 2.0/2.1 の場合)

- ホスト OS 間クラスタを構築する場合、仮想マシンに統合サービスを必ずインストールしてください。

- 仮想マシンを起動するスクリプト (vmstart.vbs) の実行前に、Hyper-V 仮想マシン管理サービス (vmms) が起動していない場合、仮想マシンの起動に失敗します。vmstart.vbs のパラメータ設定ファイル W#HyperV.bat で仮想マシンの起動リトライ回数 (W#HyperV6) を設定してください。

- フェイルオーバグループ停止時またはクラスタ停止時に、ディスクリソースまたはミラーディスクリソースの切断に失敗することがあります。W#HyperV.bat でディスク切断エラー回避用待ち時間 (W#HyperV7) を設定してください。

- フェイルオーバグループの移動時に、ゲスト OS のサスペンドではなく、シャットダウンを行いたい場合、W#HyperV.bat 内のパラメータ W#HyperV8 を変更してください。

- Hyper-V 2.0 (Windows Server 2008 R2) では、構成情報のみのエクスポートは行えません。このため、エクスポート実行時、一時的にゲスト OS のディスクサイズの 2 倍の容量が必要となります。

- Hyper-V マネージャで仮想マシンのスナップショットを取得した直後に、そのサーバが CLUSTERPRO 以外からのシャットダウンや停電などで停止した場合、スナップショットが無効となります。スナップショット取得後に、仮想マシングループを移動、マイグレーションまたは停止のいずれかの操作をすることで、この現象を回避することができます。

ホスト OS 間クラスタの注意事項 (CLUSTERPRO X 3.0 の場合)

- ホスト OS 間クラスタを構築する場合、仮想マシンに統合サービスを必ずインストールしてください。

- Hyper-V 2.0 (Windows Server 2008 R2) では、構成情報のみのエクスポートは行えません。このため、エクスポート実行時、一時的にゲスト OS のディスクサイズの 2 倍の容量が必要となります。

- 仮想マシンリソースの [仮想マシン停止待ち時間] の既定値は 60 秒に設定されていますが、環境によっては仮想マシンの停止にそれ以上時間がかかる場合があります。お使いの環境に合わせて、時間を適宜調整してください。

- Hyper-V マネージャで仮想マシンのスナップショットを取得した直後に、そのサーバが CLUSTERPRO 以外からのシャットダウンや停電などで停止した場合、スナップショットが無効となります。スナップショット取得後に、仮想マシングループを移動、マイグレーションまたは停止のいずれかの操作をすることで、この現象を回避することができます。

ゲスト OS 間クラスタの注意事項

- ゲスト OS 間クラスタで共有ディスク型クラスタを構築する場合、共有ディスク装置 (物理ハードディスク) が必要となります。仮想ハードディスクを共有ディスク装置の代わりとして使用することはできません。また、あらかじめ物理マシンの [ディスクの管理] でディスクリソースとして用いる共有ディスク上の論理ドライブ (LUN) をオフラインにしておく必要があります。

- 同一物理マシン内で起動するゲスト OS 間で、共有ディスク型クラスタを構築することはできません。仮想マシンの起動中に共有ディスクがロックされるため、同一物理マシン内の他の仮想マシンの起動時にエラーが出力され、起動に失敗します。

- SUSE Linux Enterprise Server 10 with Service Pack 1 同士で共有ディスク型またはミラーディスク型クラスタを構築する場合、仮想ハードディスクまたは物理ハードディスクを IDE コントローラに接続してください。デフォルトの設定では、IDE コントローラ 1 に DVD ドライブが接続されています。OS のインストール後、DVD ドライブを削除し、仮想ハードディスクまたは物理ハードディスクを IDE コントローラ 1 に接続してください。

- クラスタ運用時には、仮想マシンの [一時停止] または [保存] を行わないでください。仮想マシンの [一時停止] または [保存] を行うと、CLUSTERPRO がハートビートタイムアウトを検出し、他のサーバでフェイルオーバグループを起動します。この状態で、[一時停止] または [保存] していた仮想マシンを [起動] すると、データ保護の観点から、そのフェイルオーバグループが起動している両方の仮想マシンをシャットダウンします。

Hyper-Vをインストールする

ホストOS間クラスタまたはゲストOS間クラスタを構築する前に、Hyper-Vの役割をホストOSに追加する必要があります。下記の手順に従い、Hyper-Vの役割を追加してください。

(1) [スタート] メニューの [管理ツール] から [サーバマネージャ] を起動し、[役割の追加] をクリックしてください。

(2) 役割から [Hyper-V] を選択し、[次へ] をクリックしてください。

(3) 注意事項が表示されます。注意事項をお読みになった後、[次へ] をクリックしてください。

(4) 仮想ネットワークの作成画面が表示されます。仮想マシンで利用するネットワーク アダプタ[1]にチェックを入れ、[次へ] をクリックしてください。

(5) [インストール] をクリックしてください。

(6) インストール後、物理マシンを再起動してください。

(7) Hyper-Vのアップデートがあれば、適宜適用してください。

[1] CLUSTERPROでは、クラスタ間のハートビートの経路として、2系統のLANを設定することを推奨しています。ゲストOS間クラスタを構築する場合、仮想マシンが利用するネットワーク アダプタを2つご用意ください。また、ホストOS間クラスタを構築する場合、ゲストOSに割り当てる仮想ネットワーク名を、ホストOS間で同じにしてください。

ホストOS間クラスタを構築する (CLUSTERPRO X 2.0/2.1の場合)

ホスト OS 間クラスタの構築の流れは以下のようになります。

1. CLUSTERPRO X のインストール
2. ディスク系のリソースを持ったクラスタの構築
3. 仮想マシンの作成
4. 仮想マシンのエクスポートおよび削除
5. 仮想マシンを制御するためのスクリプトリソースの登録
6. 構成情報のアップロード

また、本手順では、便宜上、フェイルオーバグループ名、リソース名、モニタリソース名を以下のように設定しています。これらの名前は環境に合わせて適宜変更してください。

グループ/リソース/モニタリソース	名前
フェイルオーバグループ	failover-vm
ディスクリソース/ミラーディスクリソース	sd-vm/md-vm
スクリプトリソース	script-vm
カスタム監視リソース	genw-vm
ディスクRW監視リソース	diskw-vm

以降の手順に従い、ホスト OS 間クラスタを構築します。

(1) 『CLUSTERPRO X インストール & 設定ガイド』に従い、各ホスト OS に CLUSTERPRO X をインストールしてください。

(2) 『CLUSTERPRO X インストール & 設定ガイド』を参考に、ディスクリソース (sd-vm) またはミラーディスク (md-vm) を持つフェイルオーバグループ (failover-vm) を作成してください。また、そのフェイルオーバグループが問題なく起動することを確認してください。

(3) 仮想マシンを作成するホスト OS で、フェイルオーバグループを起動してください。

(4) 仮想マシンを任意の場所に作成します [1]。

A) [スタート] メニューの [管理ツール] から [Hyper-Vマネージャ] を起動してください。
B) Hyper-Vマネージャに表示されるホストOS名を右クリックし、[新規] の [仮想マシン] をクリックしてください。
C) [名前] に任意の仮想マシン名を入力してください。また、仮想マシンの構成情報を保存する場所を指定し、[次へ] をクリックしてください。
D) 仮想マシンに割り当てるメモリを設定し、[次へ] をクリックしてください。
E) ネットワーク アダプタを [接続しない] に設定し、[次へ] をクリックしてください。
F) 仮想ハードディスク (vhdファイル) の名前と、保存場所を指定し、[次へ] をクリックしてください。
G) インストールメディアを選択肢、[次へ] をクリックしてください。

[1] Hyper-V 1.0の場合、構成情報のみのエクスポート (仮想ハードディスクのコピーが不要な移行処理) が行えるので、仮想マシンを予め共有ディスクまたはミラーディスク上に作成することをお勧めします。

H) [作成後に仮想マシンを起動する] にチェックを入れ、[完了] をクリックしてください。
I) ゲストOSのインストールが始まります。ゲストOSのインストールを行ってください。
J) ゲストOSのインストール後、統合サービスをインストールしてください[1]。
K) ゲストOSのIPアドレスやコンピュータ名を適宜設定してください。

(5) 仮想マシンのネットワーク アダプタと自動開始アクションの設定を行います。

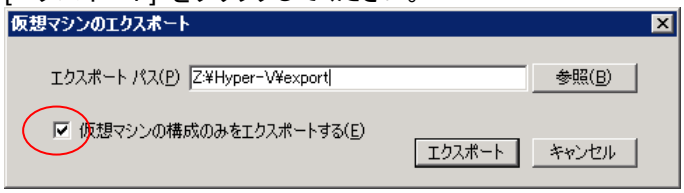
A) 仮想マシンが起動している場合、その仮想マシンをシャットダウンしてください。
B) Hyper-Vマネージャに表示されているフェイルオーバ対象とする仮想マシン名を右クリックし、[設定] をクリックしてください。
C) 画面左側の [ハードウェア] にある [ネットワーク アダプタ] をクリックしてください。[ネットワーク] のプルダウンメニューから仮想マシンに割り当てるネットワーク アダプタを選択してください。MACアドレスの設定を [静的] にし、任意のMACアドレスを割り当ててください。
D) 画面左側の [管理] にある [自動開始アクション] をクリックしてください。自動開始アクションとして [何もしない] を選択してください。
E) 上記の設定完了後、[OK] をクリックしてください。

(6) フェイルオーバ対象とする仮想マシンを共有ディスクまたはミラーディスク上にエクスポートします。

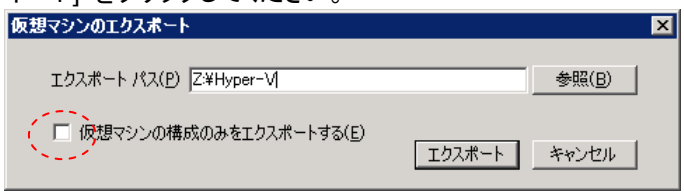
A) フェイルオーバ対象とする仮想マシンが起動している場合、その仮想マシンをシャットダウンしてください。
B) Hyper-Vマネージャに表示されているエクスポート対象の仮想マシン名を右クリックし、[エクスポート] をクリックしてください。Hyper-V 1.0とHyper-V 2.0で手順が少し異なります。

- Hyper-V 1.0の場合

  共有ディスクまたはミラーディスク上に仮想マシンの構成情報および仮想ハードディスクがある場合、エクスポート パスとして仮想マシンの構成情報を格納しているフォルダ以外の、任意のフォルダへのパス (例 Z:¥Hyper-V¥export) を入力してください。[仮想マシンの構成のみをエクスポートする] にチェックを入れ、[エクスポート] をクリックしてください。



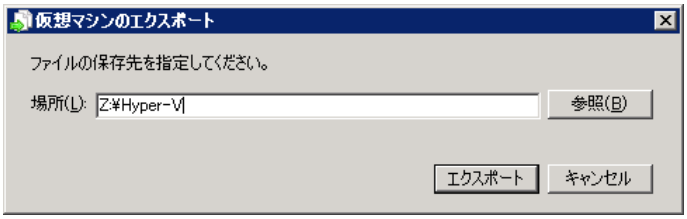
  共有ディスクまたはミラーディスク以外の場所に仮想マシンの構成情報および仮想ハードディスクがある場合、エクスポート パスとして、共有ディスクまたはミラーディスク上の任意のフォルダへのパス (例 Z:¥Hyper-V) を入力してください。[仮想マシンの構成のみをエクスポートする] にチェックを入れず、[エクスポート] をクリックしてください。



- Hyper-V 2.0の場合

  ファイルの保存先として、共有ディスクまたはミラーディスク上の任意のフォルダ (例 Z:¥Hyper-V) を入力し、[エクスポート] をクリックしてください。

[1] 仮想マシンを仮想マシンリソースで管理する場合、**統合サービスが必須となります。**



(7) Hyper-Vマネージャから、仮想マシンの削除を行います。エクスポートを行った仮想マシン名を右クリックし、[削除] をクリックしてください [1]。

(8) Hyper-V 1.0 で、既に共有ディスクまたはミラーディスク上に保存していた仮想マシンをエクスポートした場合のみ、本手順が必要となります。上記 (6) のエクスポートによって Z:¥Hyper-V¥export配下に作成されたフォルダを、仮想マシンの構成情報が保存されていたフォルダ (例 Z:¥Hyper-V¥<仮想マシン名>) にコピーします。各ファイルとフォルダが、下記のようにコピーされていることを確認してください。

- Z:¥Hyper-V¥<仮想マシン名>配下に config.xml ファイルがある。
- Z:¥Hyper-V¥<仮想マシン名>¥Virtual Machines 配下に<ゲスト OS の ID>.exp ファイルと、そのファイル名と同名のフォルダがある。

(9) CLUSTERPRO Builder を起動してください。

(10) CLUSTERPROから仮想マシンの起動/停止を行うためのVBScriptを作成します。本章末尾の サンプルスクリプトを参考に、vmstart.vbs, vmstop.vbs, W#HyperV.batを作成し、CLUSTERPRO Builderを起動しているマシン上の任意の場所に保存してください。

(11) CLUSTERPRO から仮想マシンの電源状態を確認する VBScript を作成します。本章末尾のサンプルスクリプトを参考に、checkstatus.vbs を作成し、各物理マシンの任意の場所 (例 C:¥Program Files¥CLUSTERPRO¥bin) に保存してください。

(12) CLUSTERPRO Builder でフェイルオーバグループにスクリプトリソースの追加および編集を行います。

A) CLUSTERPRO Builderの画面で、ツリービューのフェイルオーバグループ名 (failover-vm) を右クリックし、[リソースの追加] をクリックしてください。
B) [タイプ] から [スクリプトリソース] を選択してください。スクリプトリソースに名前 (script-vm) を設定し、[次へ] をクリックしてください。
C) [Start script] を選択し、[編集] をクリックしてください。エディタが起動するので、本章末尾のスクリプトサンプルを参考に、start.batを編集してください。編集内容を保存後、エディタを終了してください。
D) [Stop script] を選択し、[編集] をクリックしてください。エディタが起動するので、本章末尾のスクリプトサンプルを参考に、stop.batを編集してください。編集内容を保存後、エディタを終了させてください。
E) [追加] をクリックし、vmstart.vbs, vmstop.vbs, W#HyperV.batを追加してください。

[1] 共有ディスクまたはミラーディスク以外の場所に保存していた仮想マシンの構成情報および仮想ハードディスクのエクスポートを行った場合、エクスポート元に残されている仮想ハードディスクは不要となりますので、手動で削除を行ってください。

(13) CLUSTERPRO Builderを用いて、クラスタにカスタム監視リソース [1]を追加します。

A) CLUSTERPRO Builderの画面で、ツリービューの [Monitors] を右クリックし、[モニタリソースの追加] をクリックしてください。

B) [タイプ] から [カスタム監視] を選択してください。カスタム監視リソースに任意の名前 (genw-vm) を設定し、[次へ] をクリックしてください。

C) [編集] をクリックし、genw.batを編集します。エディタが起動するので、本章末尾のスクリプトサンプルを参考にgenw.batを編集してください。編集内容を保存後、エディタを終了させ、[次へ] をクリックしてください。

D) [監視タイミング] として [活性時] を選択し、[参照] をクリックしてください。選択可能なリソースの一覧が表示されますので、先ほど作成したスクリプトリソース (script-vm) を選択してください[2]。対象リソースにスクリプトリソースが設定されていることを確認し、[次へ] をクリックしてください。

E) [回復対象] としてスクリプトリソース (script-vm) が属するフェイルオーバグループ名 (failover-vm) を選択し、[OK] をクリックしてください。[再活性化しきい値] などのパラメータを適宜編集し、[完了] をクリックしてください。

(14) クラスタにディスク RW 監視リソースを追加します。

A) CLUSTERPRO Builderの画面で、ツリービューの [Monitors] を右クリックし、[モニタリソースの追加] をクリックしてください。

B) [タイプ] から [ディスクRW監視] を選択してください。ディスクRW監視リソースに任意の名前 (diskw-vm) を設定し [次へ] をクリックしてください。

C) [ファイル名] に共有ディスクまたはミラーディスク上のパスを指定します。ファイルには絶対パスを指定してください。指定されたファイルは自動で作成します。設定後、[次へ] をクリックしてください。

D) 監視タイミングを [活性時] に設定し、対象リソースとして、ディスクリソースまたはミラーディスクリソースを指定してください。設定後、[次へ] をクリックしてください。

---

[1] カスタム監視リソースを使うためには、CLUSTERPROのアップデート (CPRO-XW020-01以降) を適用する必要があります。

[2] この設定により、スクリプトリソースの起動後、すなわちゲストOSの起動後からゲストOSの監視を開始します。また、スクリプトリソースの停止処理開始時に、すなわちゲストOSの保存処理開始時に、ゲストOSの監視を停止します。

E) [回復対象] として、クラスタを選択してください。[最終動作] として[クラスタサービス停止とOSシャットダウン] を選択してください。設定完了後、[完了] をクリックしてください。

(15) WebManager または clpgrp コマンドでフェイルオーバグループ (failover-vm) を停止してください。

(16) WebManager または clpcl コマンドでクラスタをサスペンドしてください。

(17) CLUSTERPRO Builder で作成した構成情報をアップロードします。Builder のメニューの [ファイル] から [情報ファイルのアップロード] を選択し、構成情報をアップロードしてください。

(18) 構成情報アップロード後、『ホストOS間クラスタの動作を確認する (CLUSTERPRO X 2.0/2.1 の場合)』に進んでください。

ホストOS間クラスタの動作を確認する (CLUSTERPRO X 2.0/2.1の場合)

(1) WebManager または clpcl コマンドで、クラスタをリジュームしてください。

(2) WebManager または clpgrp コマンドでフェイルオーバグループを起動してください。フェイルオーバグループが起動しているサーバで、ゲスト OS が起動していることを Hyper-V マネージャで確認してください。

(3) WebManager または clpgrp コマンドでフェイルオーバグループを移動してください。フェイルオーバグループの移動先のサーバで、ゲストOS が起動していることを Hyper-V マネージャで確認してください。

(4) WebManager または clpdown コマンドで、フェイルオーバグループが起動している物理マシンのシャットダウンまたはリブートを行ってください。この時、フェイルオーバグループが他のサーバへ移動し、ゲスト OS が起動していることを Hyper-V マネージャで確認してください。

(5) Hyper-V マネージャでゲスト OS をシャットダウンすると、vmw1 または genw-vm が異常を検出し、回復対象の再活性またはフェイルオーバを行うことを確認してください。また、フェイルオーバ後にゲストOS が再起動されていることを Hyper-V マネージャで確認してください。

(6) CLUSTERPRO 以外から物理マシンの電源を落とした場合に、他方のサーバで相手サーバの停止を検出し、フェイルオーバグループを起動し、ゲスト OS が再起動されていることを Hyper-V マネージャで確認してください。

(7) 上記に加え、『CLUSTERPRO X インストール & 設定ガイド 第 7 章 動作チェックを行う 動作確認テストを行う』に記載されている項目を適宜実施してください。

ホストOS間クラスタを構築する (CLUSTERPRO X 3.0の場合)

ホスト OS 間クラスタの構築の流れは以下のようになります。

1. CLUSTERPRO X のインストール
2. ディスク系のリソースを持ったクラスタの構築
3. 仮想マシンの作成
4. 仮想マシンのエクスポートおよび削除
5. 仮想マシンリソースの登録
6. 構成情報の反映

また、本手順では、便宜上、フェイルオーバグループ名、リソース名、モニタリソース名を以下のように設定しています。これらの名前は環境に合わせて適宜変更してください。

グループ/リソース/モニタリソース	名前
仮想マシングループ	virtualmachine
仮想マシンリソース	vm
下記いずれかのリソース ハイブリッドディスクリソース ミラーディスクリソース NASリソース ディスクリソース	 hd-vm md-vm nas-vm sd-vm
仮想マシン監視リソース	vmw1 仮想マシン監視リソースは仮想マシンリソース作成時に、自動で追加されます。

以降の手順に従い、ホスト OS 間クラスタを構築します。

(1) 『CLUSTERPRO X インストール & 設定ガイド』に従い、各ホスト OS に CLUSTERPRO X をインストールしてください。

(2) 『CLUSTERPRO X インストール & 設定ガイド』に従い、Builder を起動し、サーバをクラスタに追加してください。

(3) 仮想マシングループをクラスタに追加します。

A) グループのタイプに [仮想マシン] を選択してください。[名前] を設定後、[次へ] をクリックしてください。
B) 起動可能サーバを設定し、[次へ] をクリックしてください。
C) グループ属性を設定し、[次へ] をクリックしてください。
D) グループリソースに仮想マシンのディスクを格納するためのハイブリッドディスクリソース、ミラーディスクリソース、NASリソース、ディスクリソースのいずれかのリソースを追加してください。この時点で仮想マシンリソースは追加しないでください。

(4) 構成情報作成後、Builder の [ファイル] メニューから [設定の反映] をクリックしてください。

(5) WebManager または clpcl コマンドで、クラスタを起動してください。

(6) 仮想マシンを作成するホスト OS で、仮想マシングループを起動してください。

(7) 仮想マシンを任意の場所に作成します [1]。

A) [スタート] メニューの [管理ツール] から [Hyper-Vマネージャ] を起動してください。
B) Hyper-Vマネージャに表示されるホストOS名を右クリックし、[新規] の [仮想マシン] をクリックしてください。
C) [名前] に任意の仮想マシン名を入力してください。また、仮想マシンの構成情報を保存する場所を指定し、[次へ] をクリックしてください。
D) 仮想マシンに割り当てるメモリを設定し、[次へ] をクリックしてください。
E) ネットワーク アダプタを [接続しない] に設定し、[次へ] をクリックしてください。
F) 仮想ハードディスク (vhdファイル) の名前と、保存場所を指定し、[次へ] をクリックしてください。
G) インストールメディアを選択肢、[次へ] をクリックしてください。
H) [作成後に仮想マシンを起動する] にチェックを入れ、[完了] をクリックしてください。
I) ゲストOSのインストールが始まります。ゲストOSのインストールを行ってください。
J) ゲストOSのインストール後、統合サービスをインストールしてください[2]。
K) ゲストOSのIPアドレスやコンピュータ名を適宜設定してください。

(8) 仮想マシンのネットワーク アダプタと自動開始アクションの設定を行います。

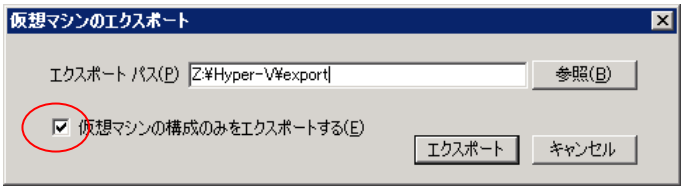
A) 仮想マシンが起動している場合、その仮想マシンをシャットダウンしてください。
B) Hyper-Vマネージャに表示されているフェイルオーバ対象とする仮想マシン名を右クリックし、[設定] をクリックしてください。
C) 画面左側の [ハードウェア] にある [ネットワーク アダプタ] をクリックしてください。[ネットワーク] のプルダウンメニューから仮想マシンに割り当てるネットワーク アダプタを選択してください。MACアドレスの設定を [静的] にし、任意のMACアドレスを割り当ててください。
D) 画面左側の [管理] にある [自動開始アクション] をクリックしてください。自動開始アクションとして [何もしない] を選択してください。
E) 上記の設定完了後、[OK] をクリックしてください。

(9) フェイルオーバ対象とする仮想マシンを共有ディスクまたはミラーディスク上にエクスポートします。

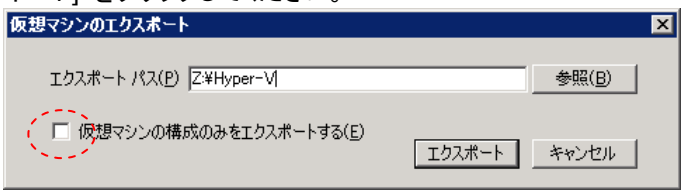
A) フェイルオーバ対象とする仮想マシンが起動している場合、その仮想マシンをシャットダウンしてください。
B) Hyper-Vマネージャに表示されているエクスポート対象の仮想マシン名を右クリックし、[エクスポート] をクリックしてください。Hyper-V 1.0とHyper-V 2.0で手順が少し異なります。
   - Hyper-V 1.0の場合
     共有ディスクまたはミラーディスク上に仮想マシンの構成情報および仮想ハードディスクがある場合、エクスポート パスとして仮想マシンの構成情報を格納しているフォルダ以外の、任意のフォルダへのパス (例 Z:¥Hyper-V¥export) を入力してください。[仮想マシンの構成のみをエクスポートする] にチェックを入れ、[エクスポート] をクリックしてください。

[1] Hyper-V 1.0の場合、構成情報のみのエクスポート (仮想ハードディスクのコピーが不要な移行処理) が行えるので、仮想マシンを予め共有ディスクまたはミラーディスク上に作成することをお勧めします。

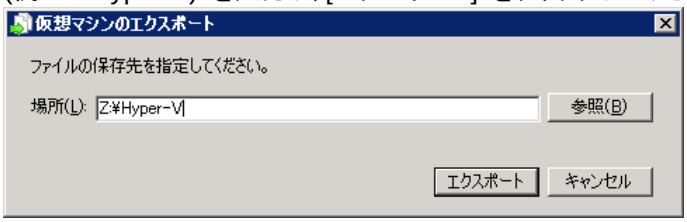
[2] 仮想マシンを仮想マシンリソースで管理する場合、**統合サービスが必須となります。**



共有ディスクまたはミラーディスク以外の場所に仮想マシンの構成情報および仮想ハードディスクがある場合、エクスポート パスとして、共有ディスクまたはミラーディスク上の任意のフォルダへのパス (例 Z:¥Hyper-V) を入力してください。[仮想マシンの構成のみをエクスポートする] にチェックを入れず、[エクスポート] をクリックしてください。



- Hyper-V 2.0の場合
  ファイルの保存先として、共有ディスクまたはミラーディスク上の任意のフォルダ (例 Z:¥Hyper-V) を入力し、[エクスポート] をクリックしてください。



(10) Hyper-Vマネージャから、仮想マシンの削除を行います。エクスポートを行った仮想マシン名を右クリックし、[削除] をクリックしてください [1]。

(11) Hyper-V 1.0 で、既に共有ディスクまたはミラーディスク上に保存していた仮想マシンをエクスポートした場合のみ、本手順が必要となります。上記 (6) のエクスポートによって Z:¥Hyper-V¥export配下に作成されたフォルダを、仮想マシンの構成情報が保存されていたフォルダ (例 Z:¥Hyper-V¥<仮想マシン名>) にコピーします。各ファイルとフォルダが、下記のようにコピーされていることを確認してください。

- Z:¥Hyper-V¥<仮想マシン名>配下に config.xml ファイルがある。
- Z:¥Hyper-V¥<仮想マシン名>¥Virtual Machines 配下に<ゲスト OS の ID>.exp ファイルと、そのファイル名と同名のフォルダがある。

(12) CLUSTERPRO Builder を起動してください。

(13) CLUSTERPRO Builder で仮想マシングループに仮想マシンリソースを追加します。

A) 仮想マシングループ名 (virtualmachine) を右クリックし、[リソースの追加] をクリックしてください。
B) タイプから [仮想マシンリソース] を選択し、名前 (vm) を設定し、[次へ] をクリックしてください。
C) 依存関係を設定し、[次へ] をクリックしてください。

[1] 共有ディスクまたはミラーディスク以外の場所に保存していた仮想マシンの構成情報および仮想ハードディスクのエクスポートを行った場合、エクスポート元に残されている仮想ハードディスクは不要となりますので、手動で削除を行ってください。

D) 活性/非活性異常検出時の復旧動作を設定し、[次へ] をクリックしてください。
E) [仮想マシンの種類] として、[Hyper-V] を選択してください。[仮想マシン名] には Hyper-Vマネージャに表示される仮想マシン名を入力してください。[VM構成ファイルのパス] に、上記 (11) でエクスポートされたexpファイルへの完全パスを入力してください。パスの途中にスペースがある場合でも、行頭、行末にダブルクォーテーション (“) を付ける必要はありません。[完了] をクリックしてください。

(14) 構成情報作成後、Builder の [ファイル] メニューから [設定の反映] をクリックしてください。

(15) 構成情報の反映後、『ホストOS間クラスタの動作を確認する (CLUSTERPRO X 3.0 の場合)』に進んでください。

ホストOS間クラスタの動作を確認する (CLUSTERPRO X 3.0の場合)

(1) WebManager または clpcl コマンドで、クラスタをリジュームしてください。

(2) WebManager または clpgrp コマンドで仮想マシングループを起動してください。仮想マシングループが起動しているサーバで、ゲスト OS が起動していることを Hyper-V マネージャで確認してください。

(3) WebManager または clpgrp コマンドで仮想マシングループを移動してください。仮想マシングループの移動先のサーバで、ゲスト OS が起動していることを Hyper-V マネージャで確認してください。

(4) WebManager または clpgrp コマンドで仮想マシングループのマイグレーションを行なってください。仮想マシングループのマイグレーション先のサーバで、ゲスト OS の起動状態が維持されたままになっていることを Hyper-V マネージャなどで確認してください。

(5) WebManager または clpdown コマンドで、仮想マシングループが起動している物理マシンのシャットダウンまたはリブートを行ってください。この時、フェイルオーバグループが他のサーバへ移動し、ゲスト OS が起動していることを Hyper-V マネージャで確認してください。

(6) Hyper-V マネージャでゲスト OS をシャットダウンすると、vmw1 が異常を検出し、回復対象の再活性またはフェイルオーバを行うことを確認してください。また、フェイルオーバ後にゲスト OS が再起動されていることを Hyper-V マネージャで確認してください。

(7) CLUSTERPRO 以外から物理マシンの電源を落とした場合に、他方のサーバで相手サーバの停止を検出し、仮想マシングループを起動し、ゲスト OS が再起動されていることを Hyper-V マネージャで確認してください。

(8) 上記に加え、『CLUSTERPRO X インストール & 設定ガイド 第 7 章 動作チェックを行う 動作確認テストを行う』に記載されている項目を適宜実施してください。

ゲストOS間クラスタを構築する

ゲスト OS 間クラスタの構築の流れは以下のようになります。

1. 仮想マシンの作成
2. CLUSTERPRO X をゲスト OS にインストール
3. CLUSTERPRO Builder でクラスタを作成

以降の手順に従い、ゲスト OS 間クラスタを構築します。

(1) 仮想マシンを任意の場所に作成します。

A) [スタート] メニューの [管理ツール] から [Hyper-Vマネージャ] を起動してください。
B) Hyper-Vマネージャに表示されるホストOS名を右クリックし、[新規] の [仮想マシン] をクリックしてください。
C) [名前] に任意の仮想マシン名を入力してください。また、仮想マシンの構成情報を保存する場所を指定し、[次へ] をクリックしてください。
D) 仮想マシンに割り当てるメモリを設定し、[次へ] をクリックしてください。
E) [接続] からクラスタ間の通信で利用するネットワーク アダプタを設定し、[次へ] をクリックしてください。
F) 仮想ハードディスク (vhdファイル) の名前と、保存場所を指定し、[次へ] をクリックしてください。
G) インストールメディアを選択肢、[次へ] をクリックしてください。
H) [作成後に仮想マシンを起動する] にチェックを入れ、[完了] をクリックしてください。
I) ゲストOSのインストールが始まります。ゲストOSのインストールを行ってください。
J) ゲストOSのインストール後、統合サービスをインストールしてください。
K) ゲストOSのIPアドレスやコンピュータ名を適宜設定してください。

(2) ディスクリソースまたはミラーディスクリソース用のディスクを追加します。

A) 仮想マシンが起動している場合、その仮想マシンをシャットダウンしてください。
B) Hyper-Vマネージャに表示される仮想マシンを右クリックし、[設定] をクリックしてください。
C) 画面左側の [ハードウェア] にある [ハードウェアの追加] をクリックし、[SCSIコントローラ] を選択し、[追加] をクリックしてください。
C) 画面左側の [ハードウェア] の一覧に追加された [SCSIコントローラ] をクリックし、[追加]をクリックしてください。

- ディスクリソースの場合
  ディスクリソースとして利用可能なのは物理ハードディスク[1]のみです。物理マシンに接続されている共有ディスク上の論理ドライブを選択してください。全ての仮想マシンで同じ論理ドライブを指定してください。

- ミラーディスクリソースの場合
  ミラーディスクリソースとして仮想ハードディスクまたは物理ハードディスク[2]が選択できます。

[1] 物理ハードディスクを利用するためには、ディスクがオフラインになっている必要があります。物理マシンの[ディスクの管理]でオフラインになっていることを確認してください。

[2] 上述の脚注のように、ディスクがオフラインになっていることを確認してください。

(3) その他、追加するハードウェアがあれば、適宜追加してください。

(4) 仮想マシンの設定完了後、仮想マシンを起動してください。

(5) 『CLUSTERPRO X インストール & 設定ガイド』に従い、CLUSTERPRO がサポートするゲスト OS を仮想マシンにインストールしてください。

(6) 『CLUSTERPRO X インストール & 設定ガイド』に従い、CLUSTERPRO Builder でクラスタを構築してください。

(7) CLUSTERPRO Builder で作成した構成情報を反映してください。

ゲストOS間クラスタの動作を確認する

(1) WebManager または clpcl コマンドでクラスタを起動してください。

(2) WebManager または clpgrp コマンドでフェイルオーバグループを移動してください。フェイルオーバグループの移動先のサーバで、フェイルオーバグループが起動していることを WebManager または clpstat コマンドで確認してください。

(3) WebManager または clpdown コマンドで、フェイルオーバグループが起動している仮想マシンのシャットダウンまたはリブートを行ってください。この時、フェイルオーバグループが他のサーバで起動していることを WebManager または clpstat コマンドで確認してください。

(4) CLUSTERPRO 以外から物理マシンの電源を落とした場合に、他方のサーバで相手サーバの停止を検出し、フェイルオーバグループを起動していることを WebManager または clpstat コマンドで確認してください。

(5) 上記に加え、『CLUSTERPRO X インストール & 設定ガイド 第 7 章 動作チェックを行う 動作確認テストを行う』に記載されている項目を適宜実施してください。

ゲストOS - ホストOS間連携の設定

本節では、ゲスト OS 上の CLUSTERPRO X SingleServerSafe が異常を検出した場合に、ホスト OS に仮想マシンのフェイルオーバを要求するための設定方法を示します。

ゲスト OS - ホスト OS 間連携の設定の流れは以下のようになります。

1. ホスト OS 間クラスタの構築
2. ゲスト OS に CLUSTERPRO X SingleServerSafe をインストール
3. ゲスト OS 上の CLUSTERPRO X SingleServerSafe の監視リソース異常検出時に、ホスト OS 上の CLUSTERPRO に対してフェイルオーバ要求を行うように設定

以降の手順に従い、ゲスト OS - ホスト OS 間連携の設定を行ないます。

(1) 「ホストOS間クラスタを構築する (CLUSTERPRO X 2.0/2.1 の場合)」または「ホストOS間クラスタを構築する (CLUSTERPRO X 3.0 の場合)」に従い、ホストOS間クラスタを構築してください。

(2) ゲスト OS に CLUSTERPRO X SingleServerSafe をインストールしてください。

(3) ゲスト OS 上の CLUSTERPRO X SingleServerSafe の構成情報を編集します。

A) CLUSTERPRO Builderを起動してください。
B) CLUSTERPRO Builderの画面で [Monitors] を右クリックし、[モニタリソースの追加] を選択してください。
C) [タイプ] から任意の監視リソースを選択してください。
D) 画面に従い、各種パラメータを設定してください。[最終動作] の設定画面まで進めてください。
E) [最終動作] として、[何もしない] を選択してください。
F) [最終動作前にスクリプトを実行する] にチェックを入れ、[設定] をクリックしてください。
G) [編集] をクリックし、末尾のサンプルスクリプトを参考に、preaction.batを編集してください。
H) 設定完了後、[完了] をクリックしてください。

(4) CLUSTERPRO Builder で作成した構成情報を反映してください。

ゲストOS - ホストOS間連携の確認

(1) WebManager または clpcl コマンドでクラスタを起動してください。

(2) 前節で設定した監視リソースの状態が [正常] となっていることを WebManager で確認してください。もしくは、clpstat コマンドで [Normal] となっていることを確認してください。

(3) 疑似障害 (アプリケーションの終了、サービスの停止、ネットワーク断線など) を発生させ、先ほど設定した監視リソースに異常を検出させます。

(4) 異常検出後、ホスト OS 側の CLUSTERPRO X がフェイルオーバの要求を受け取り、仮想マシンをフェイルオーバさせることを確認してください。

外部監視連携の設定と動作の確認

本節では、CLUSTERPRO がインストールされていないホスト OS から、ゲスト OS 上のクラスタに対して、フェイルオーバ要求を行なうための設定を示します。

1. ゲスト OS 間クラスタの構築
2. ホスト OS の任意の場所に clprexec コマンドを保存

以降の手順に従い、ホスト OS 間クラスタを構築します。

(1) 「ゲストOS間クラスタを構築する」に従い、ゲストOS間クラスタを作成してください。

(2) ホスト OS の任意の場所に、clprexec コマンドを保存します。clprexec コマンドは、CLUSTERPRO のインストールフォルダ配下またはインストール CD に格納されています。

- <CLUSTERPRO のインストールフォルダ>¥bin
- <CD ドライブ>:¥Windows¥<バージョン番号>¥common¥tools¥<アーキテクチャ>

(3) ゲスト OS 上でグループが起動していることを確認し、ホスト OS で下記のコマンドを実行し、グループが移動することを確認してください。

clprexec --failover <グループ名> -h <グループが起動しているゲスト OS の IP>

サンプルスクリプト

本章に示されているホスト OS 間クラスタを構築するために必要となるスクリプトは下記になります。CLUSTERPRO X 3.0 の場合、仮想マシンリソースがあるため、下記のスクリプトは不要です。W#HyperV.bat, genw.bat のスクリプト内で**太字**になっている箇所は、お使いになられる環境に合わせて適宜編集してください。

- vmstart.vbs
- vmstop.vbs
- W#HyperV.bat
- start.bat
- stop.bat
- genw.bat

本章に示されているゲスト OS - ホスト OS 間連携に必要となるスクリプトは下記になります。

- preaction.bat

vmstart.vbs

仮想マシンを起動するためのスクリプトです。

```
'=============================================================
' vmstart.vbs
'=============================================================
' Title   : Import VM's config file and start the VM
' Date    : 2009/04/07
' Revised : 2009/10/20
'=============================================================

option explicit

dim objWMIService
dim managementService
dim fileSystem
dim objStdOut
const JobStarting = 3
const JobRunning = 4
const JobCompleted = 7
const wmiStarted = 4096
const wmiSuccessful = 0
const Enabled = 2
const Disabled = 3
const Suspended = 32769
' Instance for standard output
Set objStdOut = Wscript.StdOut

Main()
```

```
'---------------------------------------------------------------------
' Main
'---------------------------------------------------------------------
Sub Main()

    dim computer, objArgs, vmName, vm, importDirectory, backupDirectory,_
        vmGUID, vmDefaultRootDirectory
    dim FromPath, DestPath, ConfigXMLPath, BackupPath
    dim objManagementServiceList
    dim retry, conRetryMax, conInterval
    set fileSystem = Wscript.CreateObject("Scripting.FileSystemObject")
    computer = "."

    set objArgs = WScript.Arguments

    if WScript.Arguments.Count = 6 then
        vmName = objArgs.Unnamed.Item(0)
        importDirectory = objArgs.Unnamed.Item(1)
        backupDirectory = objArgs.Unnamed.Item(2)
        vmGUID = objArgs.Unnamed.Item(3)
        vmDefaultRootDirectory = objArgs.Unnamed.Item(4)
        conRetryMax = CInt(objArgs.Unnamed.Item(5))
    else
   objStdOut.WriteLine "usage: cscript vmstart.vbs vmName " & _
        "importDirectoryName backupDirectoryName " & _
        "vmGUID vmDefaultRootDirectoryName " & _
        "vmmsWaitTime(second)"
        WScript.Quit(1)
    end if

    set objWMIService = GetObject("winmgmts:¥¥" & computer & _
    "¥root¥virtualization")

    conInterval = 1000
    for retry = 0 To conRetryMax
        set objManagementServiceList = objWMIService.ExecQuery(_
            "select * from Msvm_VirtualSystemManagementService")
        if objManagementServiceList.count = 0 then
            if retry < conRetryMax then
                objStdOut.WriteLine "Error! Virtual Machine Management " & _
                "Service(VMMS) is not started, retry..."
            else
                objStdOut.WriteLine "Error! Virtual Machine Management " & _
                "Service(VMMS) is not started."
                WScript.Quit(1)
            end If
        else
            objStdOut.WriteLine "Virtual Machine Management Service " & _
            "(VMMS) has been started."
            Exit For
```

```
    end if
    WScript.Sleep(conInterval)
next

set managementService = objManagementServiceList.ItemIndex(0)
FromPath = backupDirectory & "¥" & vmName
DestPath = importDirectory

'check if GuestOS(vmName) exist ,if exist destroy it
if CheckVMIfExist(vmName, importDirectory, vmGUID, _
   vmDefaultRootDirectory, backupDirectory) then
    if fileSystem.FolderExists(FromPath) then
        set vm = GetComputerSystem(vmName)
        if DestroyVirtualSystem(vm) then
            objStdOut.WriteLine "Destroy previous vm done."
        else
            objStdOut.WriteLine "DestroyVirtualSystem Failed."
            WScript.Quit(1)
        end if
        'copy previous vm's config file to importDirectory
        fileSystem.CopyFolder FromPath, DestPath, true
    else
        objStdOut.WriteLine "error! not first time, " & _
        "but previous backupd vm's config file doesn't exist!"
        objStdOut.WriteLine "Maybe backup file was deleted by user, " & _
        "please check it."
        WScript.Quit(1)
    end if
end if

ConfigXMLPath = importDirectory & "¥" & "config.xml"
if not fileSystem.FileExists(ConfigXMLPath) then
    'copy previous vm's config file to importDirectory
    fileSystem.CopyFolder FromPath, DestPath, true
end if

'edit config.xml
UpdateConfigFile ConfigXMLPath
objStdOut.WriteLine "UpdateConfigFile done."

'clear previous vm that have the same GUID.
ClearVmGuid vmDefaultRootDirectory, vmGUID

'import VM's config file
if ImportVirtualSystem(importDirectory, vmGUID, "false") then
    objStdOut.WriteLine "Import done."
else
    objStdOut.WriteLine "ImportVirtualSystem Failed."
    WScript.Quit(1)
end if
```

```
    set vm = GetComputerSystem(vmName)

    'objStdOut.WriteLine "importDirectory :" & importDirectory
    'objStdOut.WriteLine "backupDirectory :" & backupDirectory

    BackupPath = backupDirectory & "¥" & vmName
    'objStdOut.WriteLine "BackupPath :" & BackupPath

    if not fileSystem.FolderExists(BackupPath) then
        objStdOut.WriteLine "first time start, vm's configuration data " & _
        "is backuping"
        'Clear VM's Scopes.
        ClearVMScopes(vm)
        'Backup vm's config file
        if BackupVMConfigFile(vm, backupDirectory, importDirectory) then
            objStdOut.WriteLine "BackupVMConfigFile done."
        else
            objStdOut.WriteLine  "BackupVMConfigFile failed"
            WScript.Quit(1)
        end if
    end if

    'start VM
    if RequestStateChange(vm, "start") then
        objStdOut.WriteLine " start done."
   WScript.Quit(0)
    else
        objStdOut.WriteLine  "RequestStateChange failed"
        WScript.Quit(1)
    end if




End Sub

'---------------------------------------------------------------------
' Edit config.xml, set 'VmStateCopied' to 'true'
'---------------------------------------------------------------------
Sub UpdateConfigFile(ConfigFilePath)
    dim objDOM
    dim subNode
    dim rtResult

    Set objDOM = WScript.CreateObject("MSXML2.DOMDocument")

    objDOM.async = False
    ' Load copy source file
    rtResult = objDOM.load(ConfigFilePath)
    If rtResult = False Then
        objStdOut.WriteLine "Failed to load XML file " & ConfigFilePath & " ."
```

```
   Set objDOM = Nothing
   WScript.Quit(1)
    End If

    set subNode = objDOM.selectSingleNode("/configuration/VmStateCopied" & _
                  "[@type = 'bool']")
    objStdOut.WriteLine Format1("VmStateCopied = {0}", subNode.text)
    subNode.text = "true"
    objStdOut.WriteLine Format1(" change VmStateCopied to '{0}'", _
    subNode.text)

    'Save the config.xml file
    objDOM.Save(ConfigFilePath)
    Set objDOM = Nothing

End Sub

'---------------------------------------------------------------------
' Retrieve Msvm_VirtualComputerSystem from base on its ElementName
'---------------------------------------------------------------------
Function GetComputerSystem(vmElementName)
    On Error Resume Next
    dim query
    query = Format1("select * from Msvm_ComputerSystem where " & _
            "ElementName = '{0}'", vmElementName)
    set GetComputerSystem = objWMIService.ExecQuery(query).ItemIndex(0)
    if (Err.Number <> 0) then
        objStdOut.WriteLine Format1("Err.Number: {0}", Err.Number)
        objStdOut.WriteLine Format1("Err.Description:{0}",Err.Description)
        WScript.Quit(1)
    end if
End Function

'---------------------------------------------------------------------
' check if GuestOS vmName exist ,if exist destroy it
'---------------------------------------------------------------------
Function CheckVMIfExist(vmElementName, importDirectory, vmGUID, _
                        vmDefaultRootDirectory, backupDirectory)
    'On Error Resume Next
    dim query, query2
    dim vmList, vmList2, vm
    dim count
    dim ConfigXMLPath, vmXMLPath, vmDefaultRootPath, BinFilePath, VSVFilePath
    dim objFolder, objFiles, objFile

    CheckVMIfExist = false
    count = 0
    ConfigXMLPath = importDirectory & "¥" & "config.xml"
    vmXMLPath = importDirectory & "¥" & "Virtual Machines" & "¥" & vmGUID & ".xml"
    vmDefaultRootPath = vmDefaultRootDirectory & "¥" & "Virtual Machines" & _
                        "¥" & vmGUID & ".xml"
```

```
    objStdOut.WriteLine "vmXMLPath: " & vmXMLPath
    'set objFolder=filesystem.GetFolder(vmXMLDirectroy)
    'set objFiles=objFolder.Files
    'objStdOut.WriteLine "objFiles.count =" & objFiles.count

    Do
        query = Format1("select * from Msvm_ComputerSystem where " & _
                "ElementName = '{0}'", vmElementName)
        set vmList = objWMIService.ExecQuery(query)

        objStdOut.WriteLine "vmList.count =" & vmList.count

        if vmList.count <> 0 then
           if (Err.Number <> 0) then
                objStdOut.WriteLine Format1(_
                "CheckVMIfExist Err.Number: {0}", Err.Number)
                objStdOut.WriteLine Format1(_
                "CheckVMIfExist Err.Description:{0}",Err.Description)
                WScript.Quit(1)
            end if
            CheckVMIfExist = true
            Exit Do
        elseif (not fileSystem.FileExists(ConfigXMLPath)) and _
               fileSystem.FileExists(vmXMLPath) then
            query2 = Format1("select * from Msvm_ComputerSystem where " & _
                     "ElementName = '{0}'", vmGUID)
            set vmList2 = objWMIService.ExecQuery(query2)
            objStdOut.WriteLine "vmList2.count =" & vmList2.count
            if vmList2.count <> 0 then
                objStdOut.WriteLine "Loop count: " & count & _
                ", per loop sleep 10 seconds."
                WScript.Sleep(10000)
            else
                Exit Do
            end if
        else
            objStdOut.WriteLine "CheckVMIfExist check OK " & _
            "(previous VM doesn't exist)."
            Exit Do
        end if
    Loop

    if CheckVMIfExist = false and fileSystem.FileExists(vmXMLPath) then
        fileSystem.DeleteFile vmXMLPath, true
        if fileSystem.FileExists(vmDefaultRootPath) then
            fileSystem.DeleteFile vmDefaultRootPath, true
        end if

        'delete previous <guid>.bin and <guid>.vsv file in backup folder
        BinFilePath = backupDirectory & "¥" & vmElementName & "¥" & _
        "Virtual Machines" & "¥" & vmGUID & "¥" & vmGUID & ".bin"
```

```
        VSVFilePath = backupDirectory & "¥" & vmElementName & "¥" & _
        "Virtual Machines" & "¥" & vmGUID & "¥" & vmGUID & ".vsv"
        'objStdOut.WriteLine "BinFilePath : " & BinFilePath
        'objStdOut.WriteLine "VSVFilePath : " & VSVFilePath
        if fileSystem.FileExists(BinFilePath) then
            fileSystem.DeleteFile BinFilePath, true
            objStdOut.WriteLine "delete previous vm's "& vmGUID & _
            ".bin file in backupDirectory success."
        end if
        if fileSystem.FileExists(VSVFilePath) then
            fileSystem.DeleteFile VSVFilePath, true
       objStdOut.WriteLine "delete previous vm's "& vmGUID & _
            ".vsv file in backupDirectory  success."
        end if

        'delete previous <guid>.bin and <guid>.vsv file in import folder
        BinFilePath = importDirectory & "¥" & "Virtual Machines" & "¥" & _
        vmGUID & "¥" & vmGUID & ".bin"
        VSVFilePath = importDirectory & "¥" & "Virtual Machines" & "¥" & _
        vmGUID & "¥" & vmGUID & ".vsv"
        if fileSystem.FileExists(BinFilePath) then
            fileSystem.DeleteFile BinFilePath, true
            objStdOut.WriteLine "delete previous vm's "& vmGUID & _
            ".bin file in importDirectory success."
        end if
        if fileSystem.FileExists(VSVFilePath) then
            fileSystem.DeleteFile VSVFilePath, true
        objStdOut.WriteLine "delete previous vm's "& vmGUID & _
        ".vsv file in importDirectory success."
        end if
    end if

End Function

'--------------------------------------------------------------------
' Change status of a virtual machine
'--------------------------------------------------------------------
Function RequestStateChange(computerSystem, action)
    dim objInParam, objOutParams

    objStdOut.WriteLine Format2("RequestStateChange({0}, {1})", _
    computerSystem.ElementName, action)

    RequestStateChange = false
    set objInParam = computerSystem.Methods_("RequestStateChange")_
                    .InParameters.SpawnInstance_()

    if action = "Suspended" then
        objInParam.RequestedState = Suspended
    elseif action = "start" then
        objInParam.RequestedState = Enabled
```

```
    elseif action = "stop" then
        objInParam.RequestedState = Disabled
    else
        objStdOut.WriteLine Format1("change state to {0} is not support!", _
        action)
        WScript.Quit(1)
    end if

    set objOutParams = computerSystem.ExecMethod_("RequestStateChange", _
                       objInParam)

    if (WMIMethodStarted(objOutParams)) then
        if (WMIJobCompleted(objOutParams)) then
            objStdOut.WriteLine Format2("VM {0} was {1} successfully", _
            computerSystem.ElementName, action)
            RequestStateChange = true
        end if
    end if

End Function

'--------------------------------------------------------------------
' Define a virtual system
'--------------------------------------------------------------------
Function ImportVirtualSystem(importDirectory, vmGUID, generateNewID)

    dim objInParam, objOutParams, vmXMLPath, binPath, vsvPath
    ImportVirtualSystem = false

    'if xml, bin, vsv file exits, delete it to import
    vmXMLPath = importDirectory & "¥" & "Virtual Machines" & "¥" & vmGUID & ".xml"
    if fileSystem.FileExists(vmXMLPath) then
        objStdOut.WriteLine vmXMLPath & "exists."
        fileSystem.DeleteFile vmXMLPath, true

        ' When xml file exists, VM was terminated by not responding to cluster.
        ' So, remove bin and vsv file to start VM without bin nor vsv file.
        binPath = importDirectory & "¥" & "Virtual Machines" & "¥" & _
                  vmGUID & "¥" & vmGUID & "¥" & ".bin"
        if fileSystem.FileExists(binPath) then
            objStdOut.WriteLine binPath & "exists."
            fileSystem.DeleteFile binPath, true
        end if
        vsvPath = importDirectory & "¥" & "Virtual Machines" & "¥" & _
                  vmGUID & "¥" & vmGUID & "¥" & ".vsv"
        if fileSystem.FileExists(vsvPath) then
            objStdOut.WriteLine vsvPath & "exists."
            fileSystem.DeleteFile vsvPath, true
        end if
    end if
```

```
    if Not fileSystem.FolderExists(importDirectory) then
        objStdOut.WriteLine Format1("importDirectory({0}) doesn't exists.", _
        importDirectory)
    end if

    set objInParam = managementService.Methods_("ImportVirtualSystem")._
                     InParameters.SpawnInstance_()
    objInParam.GenerateNewID = generateNewID
    objInParam.ImportDirectory = importDirectory

    set objOutParams = managementService.ExecMethod_(_
                       "ImportVirtualSystem", objInParam)

    if objOutParams.ReturnValue = wmiStarted then
        if (WMIJobCompleted(objOutParams)) then
            ImportVirtualSystem = true
        end if
    elseif (objOutParams.ReturnValue = wmiSuccessful) then
        ImportVirtualSystem = true
    else
        objStdOut.WriteLine Format1("DefineVirtualSystem failed with " & _
        "ReturnValue {0}", wmiStatus)
    end if

End Function

'---------------------------------------------------------------------
' Destroy a virtual system
'---------------------------------------------------------------------
Function DestroyVirtualSystem(computerSystem)

    dim objInParam, objOutParams

    DestroyVirtualSystem = false
    set objInParam = managementService.Methods_("DestroyVirtualSystem")._
    InParameters.SpawnInstance_()
    objInParam.ComputerSystem = computerSystem.Path_.Path

    set objOutParams = managementService.ExecMethod_(_
    "DestroyVirtualSystem", objInParam)

    if (objOutParams.ReturnValue = wmiStarted) then
        if (WMIJobCompleted(objOutParams)) then
            DestroyVirtualSystem = true
        end if
    elseif (objOutParams.ReturnValue = wmiSuccessful) then
        DestroyVirtualSystem = true
    else
        objStdOut.WriteLine Format1("DestroyVirtualSystem failed with " & _
        "ReturnValue {0}", objOutParams.ReturnValue)
    end if
```

```
End Function

'--------------------------------------------------------------------
' Export a virtual system
'--------------------------------------------------------------------
Function ExportVirtualSystem(computerSystem, exportDirectory)
    dim vmPath, vmTmpPath, tmpExportDirectory
    dim objInParam, objOutParams

    ExportVirtualSystem = false
    tmpExportDirectory = exportDirectory & "¥" & "tmp"
    vmPath = exportDirectory & "¥" & computerSystem.ElementName
    vmTmpPath = tmpExportDirectory & "¥" & computerSystem.ElementName

    if Not fileSystem.FolderExists(exportDirectory) then
        fileSystem.CreateFolder(exportDirectory)
        objStdOut.WriteLine "Warn, previous backuped vm's config file " & _
        "doesn't exist!"
    else
        if fileSystem.FolderExists(vmTmpPath) then
            fileSystem.DeleteFolder vmTmpPath, true
        end if

    end if

    set objInParam = managementService.Methods_("ExportVirtualSystem")._
    InParameters.SpawnInstance_()
    objInParam.ComputerSystem = computerSystem.Path_.Path
    objInParam.CopyVmState = false
    objInParam.ExportDirectory = tmpExportDirectory

    set objOutParams = managementService.ExecMethod_(_
    "ExportVirtualSystem", objInParam)

    if objOutParams.ReturnValue = wmiStarted then
        if (WMIJobCompleted(objOutParams)) then
            ExportVirtualSystem = true
            objStdOut.WriteLine "vmPath: " & vmPath
            if fileSystem.FolderExists(vmPath) then
                fileSystem.DeleteFolder vmPath, true
            end if
            fileSystem.MoveFolder vmTmpPath, vmPath
        end if
    elseif (objOutParams.ReturnValue = wmiSuccessful) then
        ExportVirtualSystem = true

        if fileSystem.FolderExists(vmPath) then
            fileSystem.DeleteFolder vmPath, true
        end if
        fileSystem.MoveFolder vmTmpPath, vmPath
```

```
    else
        objStdOut.WriteLine Format1("DefineVirtualSystem failed with " & _
        "ReturnValue {0}", wmiStatus)
    end if

End Function

'--------------------------------------------------------------------
' Backup a virtual system's config file
'--------------------------------------------------------------------
Function BackupVMConfigFile(computerSystem, exportDirectory, vmrootDirectory)
    Dim tmpExportDirectory
    Dim FromPath, DestPath, BinFilePath

    BackupVMConfigFile = false

    'export VM's config file
    if ExportVirtualSystem(computerSystem, exportDirectory) then
        objStdOut.WriteLine "Export done."
    else
        objStdOut.WriteLine "ExportVirtualSystem Failed."
        Exit Function
    end if

    'move status file to temp folder

    FromPath = vmrootDirectory & "¥" & "Virtual Machines" & "¥" & _
               computerSystem.Name & "¥" & "*"
    DestPath = exportDirectory & "¥" & computerSystem.ElementName & "¥" & _
               "Virtual Machines" & "¥" & computerSystem.Name
    BinFilePath = vmrootDirectory & "¥" & "Virtual Machines" & "¥" & _
                  computerSystem.Name & "¥" & computerSystem.Name & ".bin"

    if fileSystem.FileExists(BinFilePath) then
        fileSystem.CopyFile FromPath, DestPath, true
        objStdOut.WriteLine "move status file done"
    else
        objStdOut.WriteLine "vm's status file doesn't exist."
    end if

    BackupVMConfigFile = true

End Function

'--------------------------------------------------------------------
' Handle wmi return values
'--------------------------------------------------------------------
Function WMIMethodStarted(outParam)

    dim wmiStatus
```

```
    WMIMethodStarted = false

    if Not IsNull(outParam) then
        wmiStatus = outParam.ReturnValue

        if  wmiStatus = wmiStarted then
            WMIMethodStarted = true
        end if

    end if

End Function

'--------------------------------------------------------------------
' Handle wmi Job object
'--------------------------------------------------------------------
Function WMIJobCompleted(outParam)

    dim WMIJob, jobState

    set WMIJob = objWMIService.Get(outParam.Job)

    WMIJobCompleted = true

    jobState = WMIJob.JobState

    while jobState = JobRunning or jobState = JobStarting
        objStdOut.WriteLine Format1("In progress...  {0}%% completed.", _
        WMIJob.PercentComplete)
        WScript.Sleep(1000)
        set WMIJob = objWMIService.Get(outParam.Job)
        jobState = WMIJob.JobState
    wend

    if (jobState <> JobCompleted) then
        objStdOut.WriteLine Format1("ErrorCode:{0}", WMIJob.ErrorCode)
        objStdOut.WriteLine Format1("ErrorDescription:{0}", _
        WMIJob.ErrorDescription)
        WMIJobCompleted = false
    end if

End Function

'--------------------------------------------------------------------
' The string formating functions to avoid string concatenation.
'--------------------------------------------------------------------
Function Format2(myString, arg0, arg1)
    Format2 = Format1(myString, arg0)
    Format2 = Replace(Format2, "{1}", arg1)
End Function
```

```
'---------------------------------------------------------------------
' The string formating functions to avoid string concatenation.
'---------------------------------------------------------------------
Function Format1(myString, arg0)
    Format1 = Replace(myString, "{0}", arg0)
End Function

'---------------------------------------------------------------------
'  clear vm's ScopeOfResidence value
'  vm import failer be due to SVCMM installed.
'---------------------------------------------------------------------
Function ClearVMScopes(computerSystem)

    Dim VMSystemGlobalSettingData
    Dim Result

    Set VMSystemGlobalSettingData = _
        (computerSystem.Associators_("MSVM_ElementSettingData", _
        "Msvm_VirtualSystemGlobalSettingData")).ItemIndex(0)
    VMSystemGlobalSettingData.ScopeOfResidence = ""
    Result = _
    managementService.ModifyVirtualSystem(computerSystem.Path_.Path, _
    VMSystemGlobalSettingData.GetText_(1))

    objStdOut.WriteLine "ClearVMScopes Result:" & Result
    if Result <> 0 then
        objStdOut.WriteLine "ClearVMScopes Error : " & Result
    end if

End Function

'---------------------------------------------------------------------
'  clear previous vm that have the same GUID.
'---------------------------------------------------------------------
Sub ClearVmGuid(vmDefaultRootDirectory, vmGUID)
    Dim vmDefaultRootPath
    vmDefaultRootPath = vmDefaultRootDirectory & "¥" & "Virtual Machines" & _
    "¥" & vmGUID & ".xml"

    if (fileSystem.FileExists(vmDefaultRootPath)) then
        fileSystem.DeleteFile vmDefaultRootPath, true
        objStdOut.WriteLine "previous vm that have the same GUID is deleted."
    end if

End Sub
```

vmstop.vbs

仮想マシンを保存するためのスクリプトです。

```
'=========================================================
' vmstop.vbs
```

```
'=============================================================
' Title   : Save VM's state, export VM's config file then destroy the VM
' Date    : 2009/04/10
' Revised : 2009/10/20
'=============================================================

option explicit

dim objWMIService
dim managementService
dim fileSystem
dim objStdOut
const JobStarting = 3
const JobRunning = 4
const JobCompleted = 7
const wmiStarted = 4096
const wmiSuccessful = 0
const Enabled = 2
const Disabled = 3
const Suspended = 32769
' Instance for standard output
Set objStdOut = Wscript.StdOut

Main()


'---------------------------------------------------------------------
' Main
'---------------------------------------------------------------------
Sub Main()

    dim computer, objArgs, vmName, vm, backupDirectory, vmrootDirectory, _
        OffAction
    dim DestPath, FromPath

    set fileSystem = Wscript.CreateObject("Scripting.FileSystemObject")
    computer = "."
    set objWMIService = GetObject("winmgmts:¥¥" & computer & _
    "¥root¥virtualization")
    set managementService = objWMIService.ExecQuery("select * from " & _
    "Msvm_VirtualSystemManagementService").ItemIndex(0)

    set objArgs = WScript.Arguments
    if WScript.Arguments.Count = 4 then
        vmName = objArgs.Unnamed.Item(0)
        backupDirectory = objArgs.Unnamed.Item(1)
        vmrootDirectory = objArgs.Unnamed.Item(2)
        OffAction = objArgs.Unnamed.Item(3)
    else
        objStdOut.WriteLine "usage: cscript vmstop.vbs vmName " & _
        "backupDirectoryName vmrootDirectoryName"
```

```
        WScript.Quit(1)
    end if

    set vm = GetComputerSystem(vmName)
    objStdOut.WriteLine "GetComputerSystem complete"

    if IsVMStarted(vm) then
        if OffAction = "shutdown" then
            if RequestShutdown(vm, vmName) then
                objStdOut.WriteLine "RequestShutdown success."
            else
                objStdOut.WriteLine "RequestShutdown failed."
            end if
        else
            'change VM's status
            if RequestStateChange(vm, OffAction) then
                objStdOut.WriteLine "Save done."
            else
                objStdOut.WriteLine  "RequestStateChange failed"
                WScript.Quit(1)
            end if
        end if
    else
        objStdOut.WriteLine "warn! vm isn't started, " & _
        Don't backup vm then destroy it."
    end if

    'Clear VM's Scopes.
    ClearVMScopes(vm)

    'Backup vm's config file
    if BackupVMConfigFile(vm, backupDirectory, vmrootDirectory) then
        objStdOut.WriteLine "BackupVMConfigFile done."
    else
        objStdOut.WriteLine  "BackupVMConfigFile failed"
        WScript.Quit(1)
    end if

    'destroy the VM
    if DestroyVirtualSystem(vm) then
        objStdOut.WriteLine "Destroy done."
        'WScript.Quit(0)
    else
        objStdOut.WriteLine "DestroyVirtualSystem Failed."
        WScript.Quit(1)

    end if

    'copy exported vm's config file to up folder
    FromPath = backupDirectory & "¥" & vm.ElementName
    DestPath = vmrootDirectory
```

```
    fileSystem.CopyFolder FromPath, DestPath, true
    'fileSystem.DeleteFolder backupDirectory, true
    objStdOut.WriteLine "copy exported vm's config file done"

End Sub

'---------------------------------------------------------------------
' Retrieve Msvm_VirtualComputerSystem from base on its ElementName
'---------------------------------------------------------------------
Function GetComputerSystem(vmElementName)
    On Error Resume Next
    dim query
    query = Format1("select * from Msvm_ComputerSystem where " & _
            "ElementName = '{0}'", vmElementName)
    set GetComputerSystem = objWMIService.ExecQuery(query).ItemIndex(0)
    if (Err.Number <> 0) then
        objStdOut.WriteLine Format1("Err.Number: {0}", Err.Number)
        objStdOut.WriteLine Format1("Err.Description:{0}",Err.Description)
        WScript.Quit(1)
    end if
End Function

'---------------------------------------------------------------------
' Change status of a virtual machine
'---------------------------------------------------------------------
Function RequestStateChange(computerSystem, action)
    dim objInParam, objOutParams

    objStdOut.WriteLine Format2("RequestStateChange({0}, {1})", _
    computerSystem.ElementName, action)

    RequestStateChange = false
    set objInParam = computerSystem.Methods_("RequestStateChange")._
                     InParameters.SpawnInstance_()

    if action = "suspended" then
        objInParam.RequestedState = Suspended
    elseif action = "start" then
        objInParam.RequestedState = Enabled
    elseif action = "stop" then
        objInParam.RequestedState = Disabled
    else
        objStdOut.WriteLine Format1("change state to {0} is not support!", _
        action)
        WScript.Quit(1)
    end if

    set objOutParams = computerSystem.ExecMethod_("RequestStateChange", _
    objInParam)

    if (WMIMethodStarted(objOutParams)) then
```

```
        if (WMIJobCompleted(objOutParams)) then
            objStdOut.WriteLine Format2("VM {0} was {1} successfully", _
            computerSystem.ElementName, action)
            RequestStateChange = true
        end if
    end if

End Function

'--------------------------------------------------------------------
' Export a virtual system
'--------------------------------------------------------------------
Function ExportVirtualSystem(computerSystem, exportDirectory)
    dim vmPath, vmTmpPath, tmpExportDirectory
    dim objInParam, objOutParams

    ExportVirtualSystem = false
    tmpExportDirectory = exportDirectory & "¥" & "tmp"
    vmPath = exportDirectory & "¥" & computerSystem.ElementName
    vmTmpPath = tmpExportDirectory & "¥" & computerSystem.ElementName

    if Not fileSystem.FolderExists(exportDirectory) then
        fileSystem.CreateFolder(exportDirectory)
        objStdOut.WriteLine "Warn, previous backuped vm's config file " & _
        "doesn't exist!"
    else
        if fileSystem.FolderExists(vmTmpPath) then
            fileSystem.DeleteFolder vmTmpPath, true
        end if

    end if

    set objInParam = managementService.Methods_("ExportVirtualSystem")._
                     InParameters.SpawnInstance_()
    objInParam.ComputerSystem = computerSystem.Path_.Path
    objInParam.CopyVmState = false
    objInParam.ExportDirectory = tmpExportDirectory

    set objOutParams = managementService.ExecMethod_("ExportVirtualSystem", _
                       objInParam)

    if objOutParams.ReturnValue = wmiStarted then
        if (WMIJobCompleted(objOutParams)) then
            ExportVirtualSystem = true

            if fileSystem.FolderExists(vmPath) then
                fileSystem.DeleteFolder vmPath, true
            end if
            fileSystem.MoveFolder vmTmpPath, vmPath
        end if
    elseif (objOutParams.ReturnValue = wmiSuccessful) then
```

```
        ExportVirtualSystem = true

        if fileSystem.FolderExists(vmPath) then
            fileSystem.DeleteFolder vmPath, true
        end if
        fileSystem.MoveFolder vmTmpPath, vmPath
    else
        objStdOut.WriteLine Format1("DefineVirtualSystem failed with " & _
        "ReturnValue {0}", objOutParams.ReturnValue)
    end if

End Function

'--------------------------------------------------------------------
' Backup a virtual system's config file
'--------------------------------------------------------------------
Function BackupVMConfigFile(computerSystem, exportDirectory, vmrootDirectory)
    Dim tmpExportDirectory
    Dim FromPath, DestPath, BinFilePath

    BackupVMConfigFile = false

    'export VM's config file
    if ExportVirtualSystem(computerSystem, exportDirectory) then
        objStdOut.WriteLine "Export done."
    else
        objStdOut.WriteLine "ExportVirtualSystem Failed."
        Exit Function
    end if

    'move status file to temp folder

    FromPath = vmrootDirectory & "¥" & "Virtual Machines" & "¥" & _
               computerSystem.Name & "¥" & "*"
    DestPath = exportDirectory & "¥" & computerSystem.ElementName & "¥" & _
               "Virtual Machines" & "¥" & computerSystem.Name
    BinFilePath = vmrootDirectory & "¥" & "Virtual Machines" & "¥" & _
                  computerSystem.Name & "¥" & computerSystem.Name & ".bin"
    objStdOut.WriteLine "FromPath: " & FromPath
    objStdOut.WriteLine "DestPath : " & DestPath
    objStdOut.WriteLine "BinFilePath: " & BinFilePath
    if fileSystem.FileExists(BinFilePath) then
        if not fileSystem.FolderExists(DestPath) then
            fileSystem.CreateFolder(DestPath)
        end if
        fileSystem.CopyFile FromPath, DestPath, true
        objStdOut.WriteLine "move status file done"
    else
        objStdOut.WriteLine "vm's status file doesn't exists."
    end if
```

```
    BackupVMConfigFile = true

End Function

'--------------------------------------------------------------------
' Destroy a virtual system
'--------------------------------------------------------------------
Function DestroyVirtualSystem(computerSystem)

    dim objInParam, objOutParams

    DestroyVirtualSystem = false
    set objInParam = _
        managementService.Methods_("DestroyVirtualSystem")._
        InParameters.SpawnInstance_()
    objInParam.ComputerSystem = computerSystem.Path_.Path

    set objOutParams = _
        managementService.ExecMethod_("DestroyVirtualSystem", objInParam)

    if (objOutParams.ReturnValue = wmiStarted) then
        if (WMIJobCompleted(objOutParams)) then
            DestroyVirtualSystem = true
        end if
    elseif (objOutParams.ReturnValue = wmiSuccessful) then
        DestroyVirtualSystem = true
    else
        objStdOut.WriteLine Format1("DestroyVirtualSystem failed with " & _
        "ReturnValue {0}", objOutParams.ReturnValue)
    end if

End Function

'--------------------------------------------------------------------
' Handle wmi return values
'--------------------------------------------------------------------
Function WMIMethodStarted(outParam)

    dim wmiStatus

    WMIMethodStarted = false

    if Not IsNull(outParam) then
        wmiStatus = outParam.ReturnValue

        if  wmiStatus = wmiStarted then
            WMIMethodStarted = true
        end if

    end if
```

```
End Function

'--------------------------------------------------------------------
' Handle wmi Job object
'--------------------------------------------------------------------
Function WMIJobCompleted(outParam)

    dim WMIJob, jobState

    set WMIJob = objWMIService.Get(outParam.Job)

    WMIJobCompleted = true

    jobState = WMIJob.JobState

    while jobState = JobRunning or jobState = JobStarting
        objStdOut.WriteLine Format1("In progress... {0}%% completed.", _
        WMIJob.PercentComplete)
        WScript.Sleep(1000)
        set WMIJob = objWMIService.Get(outParam.Job)
        jobState = WMIJob.JobState
    wend

    if (jobState <> JobCompleted) then
        objStdOut.WriteLine Format1("ErrorCode:{0}", WMIJob.ErrorCode)
        objStdOut.WriteLine Format1("ErrorDescription:{0}", _
        WMIJob.ErrorDescription)
        WMIJobCompleted = false
    end if

End Function

'--------------------------------------------------------------------
' The string formating functions to avoid string concatenation.
'--------------------------------------------------------------------
Function Format2(myString, arg0, arg1)
    Format2 = Format1(myString, arg0)
    Format2 = Replace(Format2, "{1}", arg1)
End Function

'--------------------------------------------------------------------
' The string formating functions to avoid string concatenation.
'--------------------------------------------------------------------
Function Format1(myString, arg0)
    Format1 = Replace(myString, "{0}", arg0)
End Function

'--------------------------------------------------------------------
' Check VM if is started
'--------------------------------------------------------------------
Function IsVMStarted(computerSystem)
```

```
    if computerSystem.EnabledState <> Enabled then
        IsVMStarted = false
    else
        IsVMStarted = true
    end if

End Function

'--------------------------------------------------------------------
' clear vm's ScopeOfResidence value
' vm import failer be due to SVCMM installed.
'--------------------------------------------------------------------
Function ClearVMScopes(computerSystem)

    Dim VMSystemGlobalSettingData
    Dim Result

    Set VMSystemGlobalSettingData = _
        (computerSystem.Associators_("MSVM_ElementSettingData", _
        "Msvm_VirtualSystemGlobalSettingData")).ItemIndex(0)
    VMSystemGlobalSettingData.ScopeOfResidence = ""
    Result = _
    managementService.ModifyVirtualSystem(computerSystem.Path_.Path, _
    VMSystemGlobalSettingData.GetText_(1))

    objStdOut.WriteLine "ClearVMScopes Result:" & Result
    if Result <> 0 then
        objStdOut.WriteLine "ClearVMScopes Error : " & Result
    end if

End Function

'--------------------------------------------------------------------
' Shutdown vm
'--------------------------------------------------------------------
Function RequestShutdown(vm, vmName)
    Dim vmShut
    Dim vmReturn
    Dim objWMI
    Dim vmList
    Dim vmState

    Set vmshut = objWMIService.ExecQuery(_
    "SELECT * FROM Msvm_ShutdownComponent WHERE SystemName='" & vm.Name & "'")
    vmReturn = vmShut.ItemIndex(0).InitiateShutdown(True,"Scripted Shutdown")
    if vmReturn = 0 then
        RequestShutdown = true
    else
        RequestShutdown = false
    end if
```

```
    Do
        Set objWMI = GetObject("winmgmts:¥¥.¥root¥virtualization")
        Set vmList = objWMI.ExecQuery(_
        "SELECT * FROM Msvm_ComputerSystem Where ElementName='" & vmName & "'")
        vmState = vmList.ItemIndex(0).EnabledState
        ' For debug
        ' objStdOut.WriteLine "Status: " & vmState
        if vmState = 3 then
            Exit Do
        end if
        WScript.Sleep(1000)
    Loop

End Function
```

W#HyperV.bat

vmstart.vbs, vmstop.vbs 用のパラメータファイルです。**太字**の箇所を適宜編集してお使いください。

```
rem ****************************************
rem *           W#HyperV.BAT               *
rem *                                      *
rem * Title   : Hyper-V Parameters         *
rem * Date    : 2008/04/14                *
rem * Reviesd : 2009/10/16                 *
rem ****************************************

rem ------------------------------------------------
rem W#HyperV.BATで設定する環境変数一覧
rem W#HyperV1 : 仮想マシン名
rem W#HyperV2 : 仮想マシンの構成ファイルを保存するためのDirectory名
rem W#HyperV3 : 仮想マシンのbackupDirectory名
rem W#HyperV4 : 仮想マシンのGUID名
rem W#HyperV5 : 仮想マシンの構成ファイルを保存するための既定のDirectory名
rem W#HyperV6 : 仮想マシンの起動リトライ回数
rem W#HyperV7 : ディスク切断エラー回避用待ち時間［秒］
rem W#HyperV8 : 停止時のアクション
rem             suspended: ゲストOSの起動状態を保存
rem             shutdown : ゲストOSのシャットダウン
rem ------------------------------------------------

SET W#HyperV1=w2k3-x64jp-vm
SET W#HyperV2="Z:¥Hyper-V¥w2k3-x64jp-vm"
SET W#HyperV3="Z:¥Hyper-V¥w2k3-x64jp-vm¥backup"
SET W#HyperV4=1DE5E97B-F1C7-4699-96C2-40FAAE271ECA
SET W#HyperV5="C:¥ProgramData¥Microsoft¥Windows¥Hyper-V"
SET W#HyperV6=30
SET W#HyperV7=30
SET W#HyperV8="suspended"
```

checkstatus.vbs

仮想マシンの起動状態を確認するためのスクリプトです。

```
'=============================================================
' Title   : Check virtual machine status
' Date    : 2009/04/07
' Revised : 2009/10/20
'=============================================================
' VM State
'     0: Unknown
'     2: Enabled (実行中)
'     3: Disabled
'    10: Rebooted
'    11: Reset
' 32768: Paused (一時停止)
' 32769: Suspended (保存完了)

Option Explicit

'-------------------------------------------------------------
' Variables
'-------------------------------------------------------------
Public objWMI
Public VM
Public VMList
Public VMName
Public VMEnabledState

Main()


'-------------------------------------------------------------
' Main()
'-------------------------------------------------------------
Sub Main()
    dim objArgs

    set objArgs = WScript.Arguments

    VMName = objArgs.Unnamed.Item(0)

    checkVMState()
    For Each VM In VMList
        VMEnabledState = VMList.ItemIndex(0).EnabledState

        WScript.Stdout.WriteLine VMEnabledState

        If VMEnabledState=0 then
            WScript.Stdout.WriteLine "Unknown"
            WScript.Quit (1)
        ElseIf VMEnabledState=3 then
            WScript.Stdout.WriteLine "Disabled"
```

```
            WScript.Quit (2)
        Else
            WScript.Quit (0)
        End If
    Next
End Sub


'-------------------------------------------------------------------
' checkVMState()
'-------------------------------------------------------------------
Sub checkVMState()
    set objWMI = GetObject("winmgmts:¥¥.¥root¥virtualization")
    set VMList = objWMI.ExecQuery(_
    "SELECT * FROM Msvm_ComputerSystem Where ElementName='" & VMName & "'")
    if VMList.count = 0 then
        WScript.Stdout.WriteLine "There is no such VM '" & VMName & _
        "' in this system."
        WScript.Quit (3)
    end if
End Sub
```

start.bat

vmstart.vbs を起動するためのスクリプトです。

```
rem ****************************************
rem *              start.bat               *
rem *                                      *
rem * Title   : Start script file          *
rem * Date    : 2009/04/07                *
rem * Revised : 2009/10/20                 *
rem ****************************************

rem ------------------------------------------
rem batchで使用する環境変数設定
rem ------------------------------------------

CALL W#HyperV.BAT

rem ****************************************
rem 起動要因チェック
rem ****************************************
IF "%CLP_EVENT%" == "START" GOTO NORMAL
IF "%CLP_EVENT%" == "FAILOVER" GOTO FAILOVER
IF "%CLP_EVENT%" == "RECOVER" GOTO RECOVER

rem CLUSTERPRO Server 未動作
GOTO no_arm
```

```
rem ************************************
rem 通常起動対応処理
rem ************************************
:NORMAL


rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK

rem scripts パスへ移動
cd %CLP_SCRIPT_PATH%

rem *************
rem 業務通常処理
rem *************
cscript
vmstart.vbs %W#HyperV1% %W#HyperV2% %W#HyperV3% %W#HyperV4% %W#HyperV5% ^
%W#HyperV6%

rem プライオリティ チェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER1

rem *************
rem 最高プライオリティ での処理
rem (例) ARMBCAST /MSG "最高プライオリティサーバで起動中です" /A
rem *************
GOTO EXIT

:ON_OTHER1
rem *************
rem 最高プライオリティ 以外での処理
rem (例) ARMBCAST /MSG "プライオリティサーバ以外で起動中です" /A
rem *************
GOTO EXIT




rem ***************************************
rem リカバリ対応処理
rem ***************************************
:RECOVER

rem *************
rem クラスタ復帰後のリカバリ処理
rem *************

GOTO EXIT
```

```
rem ****************************************
rem フェイルオーバ対応処理
rem ****************************************
:FAILOVER

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK

rem scripts パスへ移動
cd %CLP_SCRIPT_PATH%

rem *************
rem フェイルオーバ後の業務起動ならびに復旧処理
rem *************
cscript
vmstart.vbs %W#HyperV1% %W#HyperV2% %W#HyperV3% %W#HyperV4% %W#HyperV5% ^
%W#HyperV6%

rem プライオリティ のチェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER2

rem *************
rem 最高プライオリティ での処理
rem (例)
rem ARMBCAST /MSG ^
rem "最高プライオリティサーバで起動中です（フェイルオーバ後）" /A
rem *************
GOTO EXIT

:ON_OTHER2
rem *************
rem 最高プライオリティ 以外での処理
rem (例)
rem ARMBCAST /MSG ^
rem "プライオリティサーバ以外で起動中です（フェイルオーバ後）" /A
rem *************
GOTO EXIT



rem ****************************************
rem 例外処理
rem ****************************************

rem ディスク関連エラー処理
:ERROR_DISK
```

```
ARMBCAST /MSG "切替パーティションの接続に失敗しました" /A
GOTO EXIT


rem ARM 未動作
:no_arm
ARMBCAST /MSG "CLUSTERPRO Server が動作状態にありません" /A



:EXIT
```

stop.bat

vmstop.vbs を起動するためのスクリプトです。

```
rem ****************************************
rem *              stop.bat                *
rem *                                      *
rem * Title   : Stop script file           *
rem * Date    : 2009/04/07                *
rem * Revised : 2009/10/16                 *
rem ****************************************

rem ----------------------------------------
rem batchで使用する環境変数設定
rem ----------------------------------------

CALL W#HyperV.BAT

rem ****************************************
rem 起動要因チェック
rem ****************************************
IF "%CLP_EVENT%" == "START" GOTO NORMAL
IF "%CLP_EVENT%" == "FAILOVER" GOTO FAILOVER

rem CLUSTERPRO Server 未動作
GOTO no_arm




rem ****************************************
rem 通常終了対応処理
rem ****************************************
:NORMAL

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK

rem scripts パスへ移動
```

```
cd %CLP_SCRIPT_PATH%

rem *************
rem 業務通常処理
rem *************
cscript vmstop.vbs %W#HyperV1% %W#HyperV3% %W#HyperV2% %W#HyperV8%

rem ディスク切断時のエラー回避用
armsleep %W#HyperV7%

rem プライオリティ チェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER1

rem *************
rem 最高プライオリティ での処理
rem (例)ARMBCAST /MSG "最高プライオリティサーバで終了中です" /A
rem *************
GOTO EXIT


:ON_OTHER1
rem *************
rem 最高プライオリティ 以外での処理
rem (例)ARMBCAST /MSG "プライオリティサーバ以外で終了です" /A
rem *************
GOTO EXIT




rem ***************************************
rem フェイルオーバ対応処理
rem ***************************************
:FAILOVER

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK

rem scripts パスへ移動
cd %CLP_SCRIPT_PATH%

rem *************
rem フェイルオーバ後の業務起動ならびに復旧処理
rem *************
cscript vmstop.vbs %W#HyperV1% %W#HyperV3% %W#HyperV2% %W#HyperV8%

rem ディスク切断時のエラー回避用
armsleep %W#HyperV7%

rem プライオリティ のチェック
```

```
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER2

rem *************
rem 最高プライオリティ での処理
rem (例)ARMBCAST /MSG "最高プライオリティサーバで終了中です（フェイルオーバ後）" /A
rem *************
GOTO EXIT

:ON_OTHER2
rem *************
rem 最高プライオリティ 以外での処理
rem (例)ARMBCAST /MSG "プライオリティサーバ以外で終了中です（フェイルオーバ後）" /A
rem *************
GOTO EXIT




rem ***************************************
rem 例外処理
rem ***************************************

rem ディスク関連エラー処理
:ERROR_DISK
ARMBCAST /MSG "切替パーティションの接続に失敗しました" /A
GOTO EXIT


rem ARM 未動作
:no_arm
ARMBCAST /MSG " CLUSTERPRO Server が動作状態にありません" /A


:EXIT
```

genw.bat

checkstatus.vbs を起動するためのスクリプトです。**太字**の箇所を適宜編集してお使いください。

```
rem ***************************************
rem *               genw.bat              *
rem * Title   : Start up checkstatus.vbs  *
rem * Date    : 2009/04/07               *
rem * Revised : 2009/04/07               *
rem ***************************************

set vmname=w2k8-x86jp-vm
set vbs_path=C:¥Program Files¥CLUSTERPRO¥bin¥checkstatus.vbs
```

```
echo START

cscript "%vbs_path%" %vmname%
if errorlevel 1 goto ERROR

:NORMAL
exit 0

:ERROR
exit 1
```

preaction.bat

監視リソース異常検出時の最終動作の実行前に、他のクラスタ (例えば、ゲスト OS 上のクラスタからホストOS 上のクラスタ) にフェイルオーバ要求を送信するためのスクリプトです。**太字**の箇所を適宜編集してお使いください。

要求の送信には clptrnreq コマンドを使います。clptrnreq コマンドの使い方については『CLUSTERPRO X リファレンス ガイド』を参照してください。

```
rem ****************************************
rem *            preaction.bat             *
rem ****************************************

echo START
echo %CLP_MONITORNAME%

rem フェイルオーバ対象に属するリソース名を指定
rem ex. set CLPRSC=script
set CLPRSC=script-vm

rem カンマ区切りで各サーバのIPを記述
rem ex. set CLPIP=10.0.0.1,10.0.0.2
set CLPIP=192.168.0.1,192.168.0.2

rem フェイルオーバ要求を送信
clptrnreq -t GRP_FAILOVER -r %CLPRSC% -h %CLPIP%

echo EXIT
```

preaction.bat (CLUSTERPRO X 3.0 の場合)

監視リソース異常検出時の最終動作の実行前に、他のクラスタ (例えば、ゲスト OS 上のクラスタからホストOS 上のクラスタ) にフェイルオーバ要求を送信するためのスクリプトです。**太字**の箇所を適宜編集してお使いください。

CLUSTERPRO X 3.0 では、要求の送信には clptrnreq または clprexec コマンドを使うことができます。ここでは clprexec コマンドを用いた場合のサンプルスクリプトを示します。

```
rem ***************************************
rem *            preaction.bat            *
rem ***************************************

echo START
echo %CLP_MONITORNAME%

rem フェイルオーバグループ名を指定
rem ex. set CLPGRP=failover
set CLPGRP=failover

rem カンマ区切りで各サーバのIPを記述
rem ex. set CLPIP=10.0.0.1,10.0.0.2
set CLPIP=192.168.0.1,192.168.0.2

rem フェイルオーバ要求を送信
clprexec --failover %CLPGRP% -h %CLPIP%

echo EXIT
```