構築ガイド

VMware:
AXシリーズとVMware vCenterによるダイナミックプロビジョニング

Table of Contents

構築ガイド

VMware: AXシリーズとVMware vCenterによるダイナミックプロビジョニング

はじめに

本ドキュメントは、AXシリーズとVMware環境による動的なアプリケーション配信インフラストラクチャを構築するための設定手順を記述しています。AXシリーズとVMwareは、動的に設定を行うことができる強力なAPIをサポートしています。ここでは、基本となるアーキテクチャの構築例を説明しています。

本ドキュメントの設定では、VMware vCenterの閾値設定による動作を使用しています。これにより、追加のリソースが必要になった場合、速やかに仮想マシン（Virtual Machine）を追加することが可能となり、同時にAXに対して追加した仮想マシンに関わる設定を動的に行うことができます。

VMwareに関する詳細は、下記のサイトをご覧ください。

http://www.vmware.com/products/vsphere/
http://www.vmware.com/products/vcenter-server/

前提条件

- バージョン2.2.4以降が動作するAXシリーズを使用していること。
- AXシリーズとVMware vSphereによる管理に関する基本的な知識があること。

本ドキュメントで使用したソフトウェアのバージョンは下記のとおりです。

AX ソフトウェア	リリース 2.2.4 以降
VMware vCenter	Version 2.5
VMware vSphere	Version 4.0
VMware ESX	Version 3.4

ご注意: バージョン2.2.4をサポートしているAXモデルは、AX 1000, AX 2000, AX 2100, AX 2200, AX 3100, AX 3200です。
バージョン2.4.1をサポートしているAXモデルは、 AX 2500, AX 2600, AX 3000, AX 5200です。

INTRODUCTION

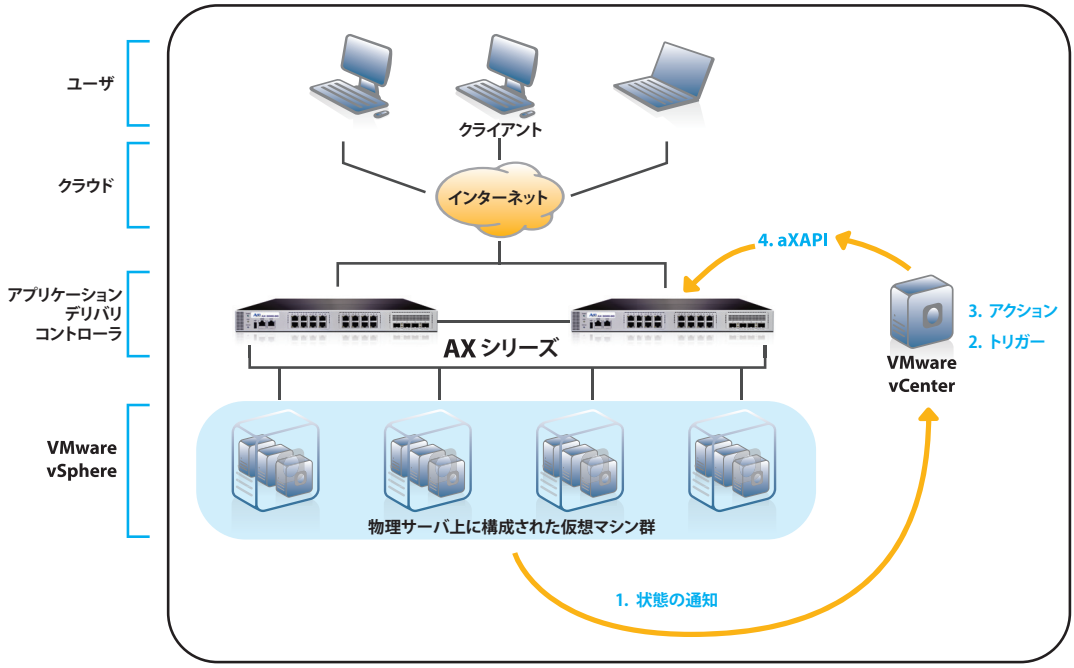
ダイナミックプロビジョニングを実現する、AXシリーズ＋VMware vCenterインテグレーション

概要

ここからAXシリーズとvCenterを使用した、新規仮想マシンインスタンス生成時のインテグレーションについて記述します。関連するAXシリーズの設定は、新規に仮想マシンが生成されたことをトリガーとして変更されます。全ての設定は自動で行われ、トラフィックは新規に作成された仮想マシンにも割り振られるようになります。

構成イメージ

次の図は、本インテグレーションの論理的な構成イメージを示しています。仮想マシンは、vCenterによって管理されています。二台のAXシリーズは、HA構成となっており、高速化やトラフィック操作を行っています。図中では、ダイナミックプロビジョニングの論理的な設定シーケンスについても記述されています。



図：VMwareとAXシリーズの論理構成

設定手順の概要

設定手順の概要と簡単な説明を以下に記述しています。

- ステップ **1:** VMwareでWebサーバが動作する仮想マシンを作成します。
- ステップ **2:** VMwareでvCenterとESX HTTP Proxyを設定します。
- ステップ **3:** AXで動的設定をサポートするための設定を行います。（仮想マシンの追加後アップデートされるAXの設定に関するものです。）
- ステップ **4:** aXAPIアプリケーションをロードします。（vCenter上のVMPowerOpsを設定）
- ステップ **5:** aXAPIアプリケーションVMPowerOpsの設定
- ステップ **6:** VMwareで、VMPowerOpsを呼び出すアプリケーションバッチプログラムとともに、仮想マシンのトリガーを設定します。
- ステップ **7:** VMwareでCPUの閾値を超えると動作するバッチプログラムとして、vSphereからアプリケーショントリガーを設定します。

動的なアプリケーション配信サービスを構築するための VMwareにおける設定

動的に仮想マシンを作成・削除する機能は、今日のクラウドコンピューティングプロバイダによるオンデマンドアプリケーションにとって重要な技術です。動的な仮想サーバをサポートするためには、そのサーバにトラフィックを割り振るためのアプリケーション配信インフラストラクチャも必須となります。このインフラストラクチャは、オンデマンドに拡張可能なものである必要があります。

本構成に必要な手順は、以下の7つのステップです。

ステップ 1: 仮想マシンの設定

本構成に必要な仮想マシンを最初の物理マシンあるいはホスト上にインストールします。必要であれば、続けて二番目以降の物理マシンにインストールします。（仮想マシンの構成は、お使いの環境に合わせて読み替えてください）

ご注意: 本構成では、インストール終了後にVMware Toolsのインストールが必要です。これは、VMwareのプロセスを正しく動作させるために必要な手順となります。VMware Toolsをインストールするには、仮想マシン上で右クリックして、ゲスト>VM Toolsのインストールを選択します。全ての仮想マシンは、AXのサービスグループにリアルサーバとして設定します。

Webサーバとして使用するゲストOS

上記で設定した仮想マシンを使用して、ポート80でWebサーバを動作させます。こちらの設定は、インストールしたWebサーバによって異なります。異なる手段として、以下のサイトから設定済みの仮想アプライアンスをVMware Virtual Appliance Marketplaceから入手することができます。

http://www.vmware.com/appliances/directory/cat/53

各Webサーバは、異なるIPアドレスを設定して、AXデバイスからアクセスできるようになっている必要があります。

ステップ 2: vCenterとESX HTTP Proxyのセットアップ

本構成では、vCenterサーバとVMware vSphere client 4.0が必要になります。デフォルトでは、VMware Virtual Center Management Webservicesは、自動起動になっていません。そのため、手動でこのサービスを起動する必要があります。

VMware Virtual Center Management Webservicesを起動するには、スタート>すべてのプログラム>コントロールパネル>管理ツール>サービス からこちらのサービスを探して、右クリックして開始する必要があります。

その後VMware vSphere client 4.0を開始して上記サービスが動作しているかを確認します。vSphereがvCenterと同一のマシン上で実行されている場合は、Windowsのユーザアカウント・パスワード及び、ローカルホストのIPアドレスを使用してサービスにログインします。異なるマシン上で実行されている場合は、ログイン時にvCenterのアドレスを入力します。

ご注意：
デフォルトでは、vCenterのWebサービスは、HTTPSコネクションのみを受信します。そのため、VMAPIは、vCenterに接続する際にエラーとなります。これを解決するためには、HTTPSの証明書をインストールするか、vCenterをHTTPアクセスできるように変更する必要があります。本ドキュメントでは、二つ目の方法を用いています。

ESX 3.5 for HTTP 上でのWeb Proxyサービス設定

1. サービスコンソールにルートユーザとしてログインします。
2. ディレクトリを/etc/vmware/hostdに変更します。
3. テキストエディタを使用して、proxy.xmlファイルを開きます。
4. ファイルの最後にある（<EndpointList>タグがあります）設定は、SDKをサポートするWebサービスのセッティングを含んでいます。ネストされたタグは、以下のような記述になっています。

   ```
   ...
   <e id="1">
   <_type>vim.ProxyService.NamedPipeServiceSpec</_type>
   <accessMode>httpsWithRedirect</accessMode>
   <pipeName>/var/run/vmware/proxy-sdk</pipeName>
   <serverNamespace>/sdk</serverNamespace>
   </e>
   ...
   ```

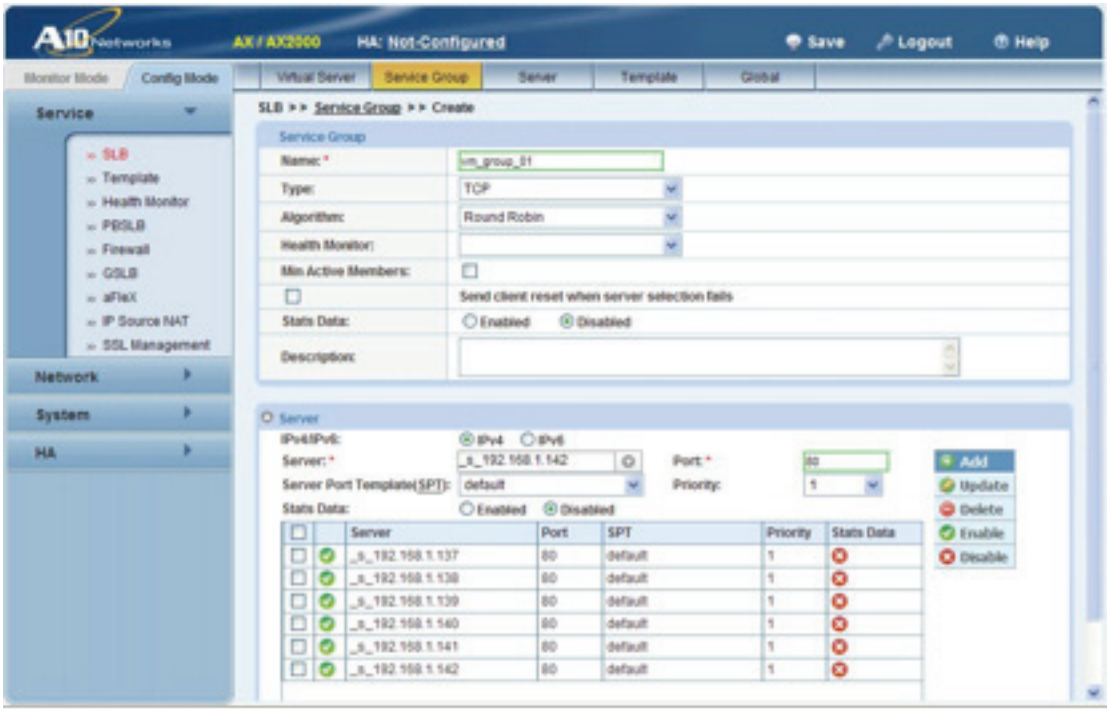
5. 次のようにアクセスモードをHTTP and HTTPSに変更します。
   `<accessMode>httpAndHttps</accessMode>`
   あるいは、次のようにHTTPSを無効にします：
   `<accessMode>httpOnly</accessMode>`

Proxy.xmlの変更が有効になるように、管理ネットワーク（ホストではなく）のリスタートを行います。

ステップ 3: AXの設定

AXの設定では、仮想マシンをサービスグループに追加します。この設定は、後ほど使用するため、追加したサービスグループの名前とサービスポートをメモしておきます。

ご注意：バーチャルサーバやリアルサーバなど、その他の設定も必要ですが、本ドキュメントには記述しておりません。



図：仮想マシンを追加したサービスグループ

ステップ 4: AXのAPIアプリケーションをvCenterにロード

AXのAPIであるaXAPIは、新規サーバの追加やサービスグループ、バーチャルサーバの作成、新規セッションの作成やサービスグループメソッドの更新など、様々な機能を持っています。本構成のためのaXAPIコードは、VMPowerOps.csと呼ばれる別のファイルで提供されます。このファイルは、vCenter内にロードして使用します。仮想マシンが動作するホストシステム上では、リソースの使用率が事前に設定していたトリガーに達するとvSphereプログラムが自動的にVMPowerOpsアプリケーションを呼び出します。このアプリケーションは、次のように動作します。

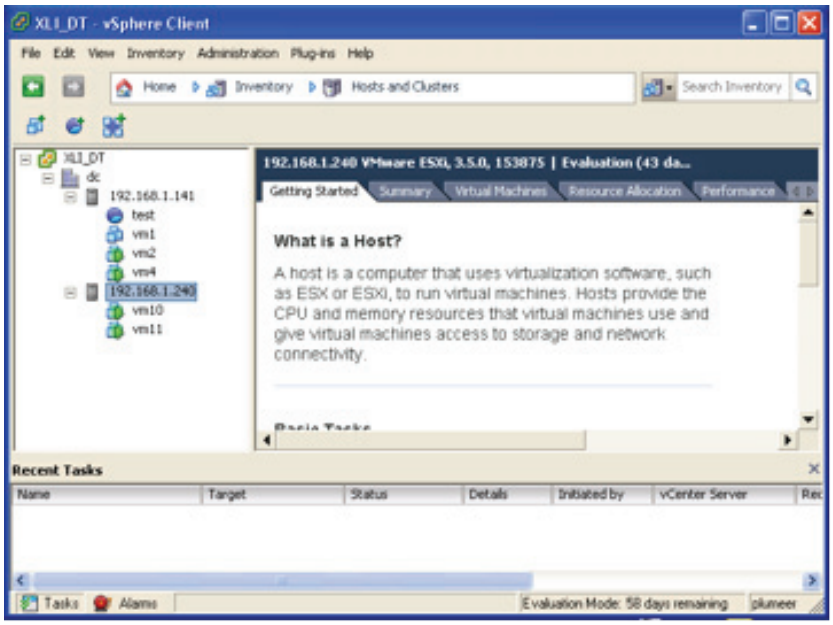
- ESX内にあるパワーオフ状態の仮想マシンを見つけて起動します。
- この仮想マシンがリアルサーバとしてトラフィックを受信できるよう、起動した仮想マシンをAXのサービスグループに追加します。そして、既にトラフィックを処理している他の仮想マシンと共に処理を共有します。
- アプリケーションが、パワーオフ状態の仮想マシンを発見できなかった場合で、仮想マシン数の合計が制限より少ない場合、アプリケーションは仮想マシンのクローンを一つ作成して起動します。仮想マシン数の合計が制限を超えていた場合、アプリケーションはなにも処理を加えません。

ご注意：ここで言うアプリケーションとは、vSphereが呼び出すVMPowerOpsのことを指しています。VMPowerOpsに関する詳細は、弊社までお問い合わせください。

ステップ 5: vSphere Clientで使用するaXAPIアプリケーションの設定

vSphere clientで、aXAPIアプリケーションを関係するイベントと共に設定します。

- vSphere clientを起動して、操作するホストを選択します。



ステップ 6: VMPowerOpsを呼び出すVMトリガーバッチプログラムのセットアップ

以下のフォーマットでVMパワートリガーを設定します。環境に合わせて、変数や引数を変更してください。
ご注意：設定では、全部で13のパラメータがあります。全てのパラメータが必須となります。各パラメータについては、例の後で説明しています。

例:

VmPowerOps.exe --url http://192.168.32.150/sdk --username TestUser --password mypasswd --hostName 192.168.1.240 --threshold 5 --DatacenterName DC --vmPath dc/vm/vm10 --AXIP 192.168.215.43 --AXUsrName admin --AXPasswd a10 --AXServiceGroup vm_group_01 --AX-Port 80 --multipleAllow VmPowerOps

変数・引数について:

Url < type String >
ログインするデータセンタのURLです。使用する環境に合わせてIPアドレスを変更します。サフィックスの/sdkは必須です。

Username < type String >
VMAPIがデータセンタにログインする際に必要となるユーザ名です。-urlに関連してます。

Password < type String >
usernameと urlに関連するパスワードです。

DatacenterName < type String >
操作の対象となるデータセンタ名です。

MultipleAllow < type String >
マルチプロセスを許可するかどうかを指定します。1は許可、2は拒否です。

AXServiceGroup < type String >
AXのサービスグループ名です。

HostName; < type String >
仮想マシンを起動するホストです。

AXPort < type String >
サーバのポート番号です。

AXIP < type String >
AXデバイスの管理IPです。

AXPasswd < type String >
AXにログインするためのパスワードです。

vmPath < type String >
clone_templateのためのパスです。

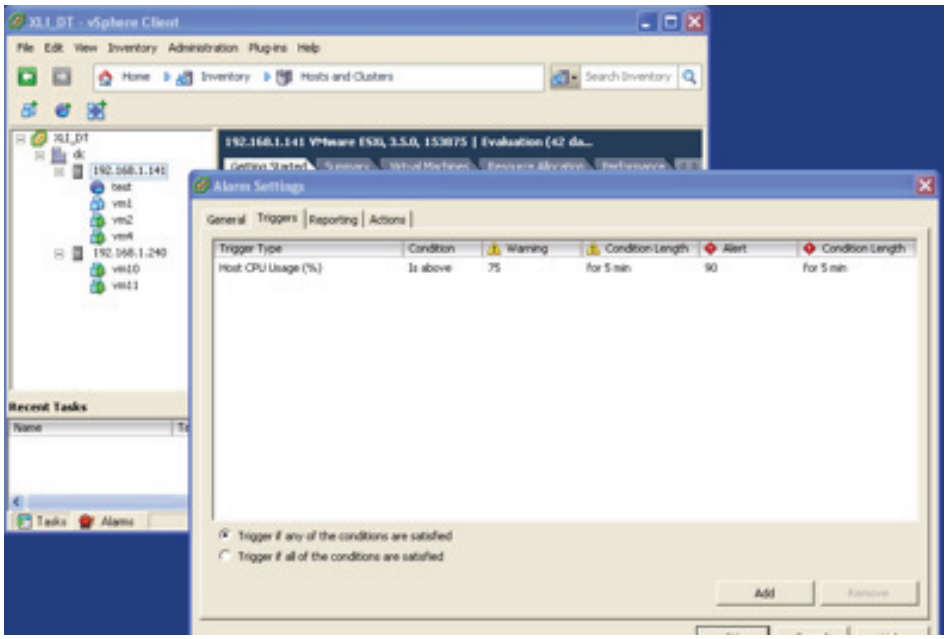
Threshold < type String >
新規仮想マシンクローンを作成する際に判断の材料となる閾値です。

AXUsrName < type String >
AXにログインするユーザ名です。

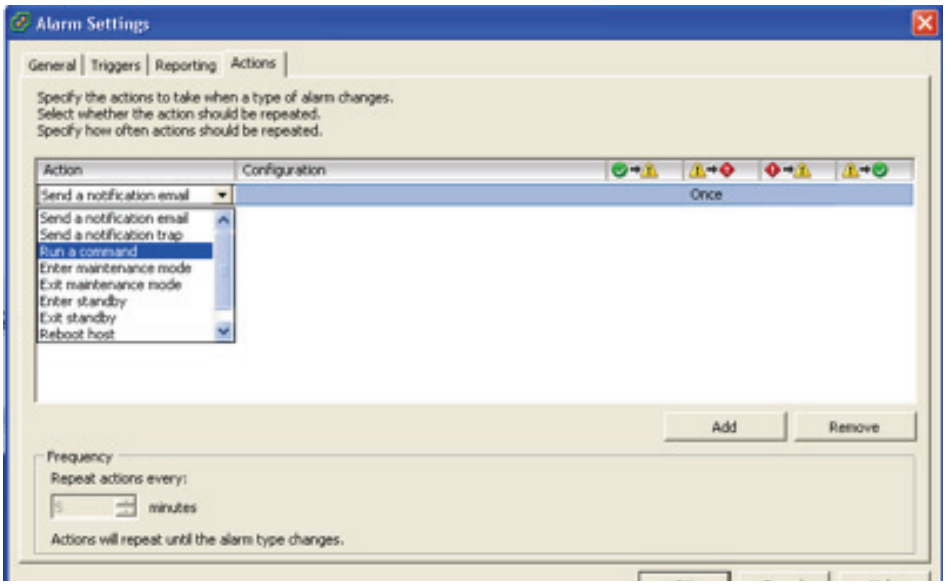
全てのパラメータを入力したら、「Power.bat」など、バッチファイルを示す.batの拡張子と共にアプリケーション名を付けて保存します。

ステップ 7: アプリケーションとトリガーの設定

vSphere clientを開いて、ログインします。
ドロップダウンリストから編集したいホストを右クリックして アラーム>アラームの追加 を選択します。
アラーム名を設定して、トリガータブをクリックします。トリガーを追加して、トリガータイプを選択します。



アクションタグをクリックして、アクションの追加を行います。アクションタイプエリアで「コマンドの実行」を選択します。



設定エリアで、.batファイルのフルパスを記述します。ここでは、「Power.bat」を指定しています。
例: C:\windows\system32\cmd.exe /c C:\Debug\Power.bat

最初のホストがトリガーに達すると、システムはこのVMPowerOpsアプリケーションを実行します。そして、AXの設定に対して動的にサーバを追加します。

補足: リソース使用率トリガープログラム

リソースの使用率が事前に設定していたトリガーに達するとvSphereプログラムが自動的にアプリケーションを呼び出します。
このアプリケーションは、次のように動作します。

- プログラムは、他のホストにあるパワーオフ状態の仮想マシンを見つけて起動します。関連するコード部分を以下に示します。
  ご注意: 全てのコードを確認するには、VMPowerOps.csファイルを参照してください。

```
public string getApowerOffVMFromHostSys(string hostSys)
    {
        Object[] objary;
        ObjectContent[] ocary;
        ManagedObjectReference vmMOR = null;
        DynamicProperty[] pcary = null;
        cb.log.LogLine("Get the object reference of the given host
        system: " + hostSys);
        //retrieve the host system by name
        ManagedObjectReference host =
cb.getServiceUtil().GetDecendentMoRef(null, "HostSystem", hostSys);
        if (host == null)
        {
                cb.log.LogLine("Cannot find the host system with hostname: " +   hostSys);
                return null;
        }
        else
        {
                cb.log.LogLine("To find a power-off VM under the host    system");
                //get the VM under the given host system
                ArrayList vms = cb.getServiceUtil().GetDecendentMoRefs(host,  "VirtualMachine");
                //find a VM with the status of power off
                for (int i = 0; i < vms.Count; i++)
                {
                        objary = (Object[])vms[i];
                        vmMOR = (ManagedObjectReference)objary[0];
                        ocary = cb.getServiceUtil().GetObjectProperties(null, vmMOR, new string[] {
```

```
“runtime.powerState” });
                        pcary = ocary[0].propSet;
                        if (pcary[0].val.ToString().Equals(“poweredOff”))
                        {
                                ObjectContent[] oc =
cb.getServiceUtil().GetObjectProperties(null, vmMOR, new string[]
{ “name” });
                                DynamicProperty[] dp = oc[0].propSet;
                                string vname = dp[0].val.ToString();
```

————————————————————

こちらは、例としてご提供しているサンプルコードです。実際のオペレーション用ではありませんのでご注意ください。

```
                                if (vname.StartsWith(“vm”))
                                {
                                        cb.log.LogLine(“Found a proper VM, successful! VM name: “ +
vname);
                                        return vname;
                                }
                                else
                                {
                                        continue;
                                }
                        }
                        else
                        {
                                        continue;
                        }
                }
                cb.log.LogLine(“Cannot find VM under “ + hostSys);
                return null;
        }
    }
```

- 仮想マシンをAXのサービスグループ内にあるサーバプールに追加します。これにより、この仮想マシンはサービスを共有するサーバとして動作します。対応するコードセグメントを以下に示します。

```
public bool addSerToSergroup(string IP,string usrname,string passwd,string srvgroupname, string
serverIP,string port)
    {
        Session se = new Session(usrname, passwd, IP);
        cb.log.LogLine(“Session ID successful”);

        A10_WebServices_AX.SLB.ServiceGroup.ServiceGroup sgg = A10_WebServices_AX.SLB.
```

```
ServiceGroup.ServiceGroup.getByName(se, “vm_group_01”);
        if (sgg != null)
        {
            cb.log.LogLine(“Get service group successful.”);
            A10_WebServices_AX.SLB.Member.Member mb = new A10_WebServices_AX.SLB.Member.
Member(serverIP, “10000”, “10000”, port, “10”, “1”, “100”);
            sgg.members = new A10_WebServices_AX.SLB.Member.Member[] { mb};
            try
            {
                A10_WebServices_AX.SLB.ServiceGroup.ServiceGroup.upDate(se, sgg);
            }
            catch
            { return false; }
            return true;
        }
        cb.log.LogLine(“Can not find the specified service group”);
        return false;
    }
```

ご注意: アプリケーションが、パワーオフ状態の仮想マシンを発見できなかった場合で、仮想マシン数の合計が制限より少ない場合、アプリケーションは、仮想マシンのクローンを一つ作成して起動します。仮想マシン数の合計が制限を超えていた場合、アプリケーションはなにも処理を加えません。VMPowerOps.cs内のCloneVM VMPowerOn機能を参照してください。

まとめ

アプリケーション配信プラットフォーム AXシリーズ を使用することで、次のようなメリットがあります。

- VMwareの仮想マシンと連携してアプリケーション配信の容量を調整することができます。
- 冗長化されたAXシリーズと複数の仮想マシングループによって高い可用性を実現することができます。
- AXを使用して多くの仮想マシンを負荷分散することにより、より高い稼働率を実現することができます。
- AXのアクセラレーション機能を用いて、負荷の高い処理をオフロードすることにより、顧客体感速度の向上と、より高いコネクションスループットを実現することができます。

AXシリーズ　アドバンスドアプリケーション配信プラットフォームを使用することで、VMware環境に高い付加価値を加えることできます。AXシリーズに関する詳しい情報については、以下のサイトをご参照ください。

http://www.a10networks.com/products/axseries.php
http://www.a10networks.com/resources/solutionsheets.php
http://www.a10networks.com/resources/casestudies.php

A10ネットワークスについて

A10ネットワークスは、ネットワーキングと、セキュリティ分野における革新的なソリューションの提供を目指して2004年に設立されました。当社は、あらゆるお客様のアプリケーションを高速化、最適化するとともに、そのセキュリティの確保をも支援することができる高性能な製品群を開発しています。当社は、米国シリコンバレーに本拠地を置く、株式非公開のテクノロジカンパニーであり、イギリス、フランス、オランダ、ドイツ、ブラジル、日本、中国、韓国、台湾にも拠点を置いています。 詳しくはホームページをご覧ください。www.a10networks.co.jp

お問い合わせ

A10ネットワークス株式会社
営業部
03-3291-0091
jinfo@a10networks.com