Deployment Guide

VMware:
Enabling vCenter Dynamic Provisioning with the AX Series

Table of Contents

DEPLOYMENT GUIDE

VMware: Enabling vCenter Dynamic Provisioning with the AX Series

Introduction

This deployment guide contains AX Series configuration steps to enable a dynamic application delivery infrastructure combined with load balancing support for VMware deployments. AX Series and VMware have flexible and powerful APIs that can be integrated and applied to create dynamic deployments. The intent of this guide is to provide a base architecture deployment example that can be used as a reference.

The deployment configuration guide is based on the VMware vCenter threshold exception where a new VM (virtual machine) is instantiated when additional resources are required. By using this guide example, the initiation of VM will dynamically trigger the provisioning of related servers on the AX platform to support the new VM.

For more information on VMware visit:

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

Assumptions

- A10's AX platform should be running software version 2.2.4 or later.
- It is assumed that users have some basic configuration familiarity with both AX as well as VMware's vSphere administration.

The software versions used for this deployment guide are as follows:

AX Software	Release 2.2.4 or later
VMware vCenter	Version 2.5
VMware vSphere	Version 4.0
VMware ESX	Version 3.4

Note: AX Models supported for R.2.2.4 are AX 1000, AX 2000, AX 2100, AX 2200, AX 3100 and AX 3200.
AX Models supported for R.2.4.1 are AX 2500, AX 2600, AX 3000 and AX 5200.

VMware vCenter Integration with AX Application Delivery Controllers for Dynamic Provisioning

Overview

The following sections show the integration of the AX Series Application Delivery Controller and vCenter when new virtual machine instances are provisioned, with the new virtual machine creation triggering the corresponding configuration on the AX Series. This automatically allows them to be placed in service and pass traffic to the new virtual machine(s).

Topology Reference

The following figure depicts logical deployment where virtual machines are managed by vCenter and where two AX Application Delivery Controllers in a high availability configuration are accelerating and directing traffic. The diagram also shows the logical flow sequence for dynamic provisioning.

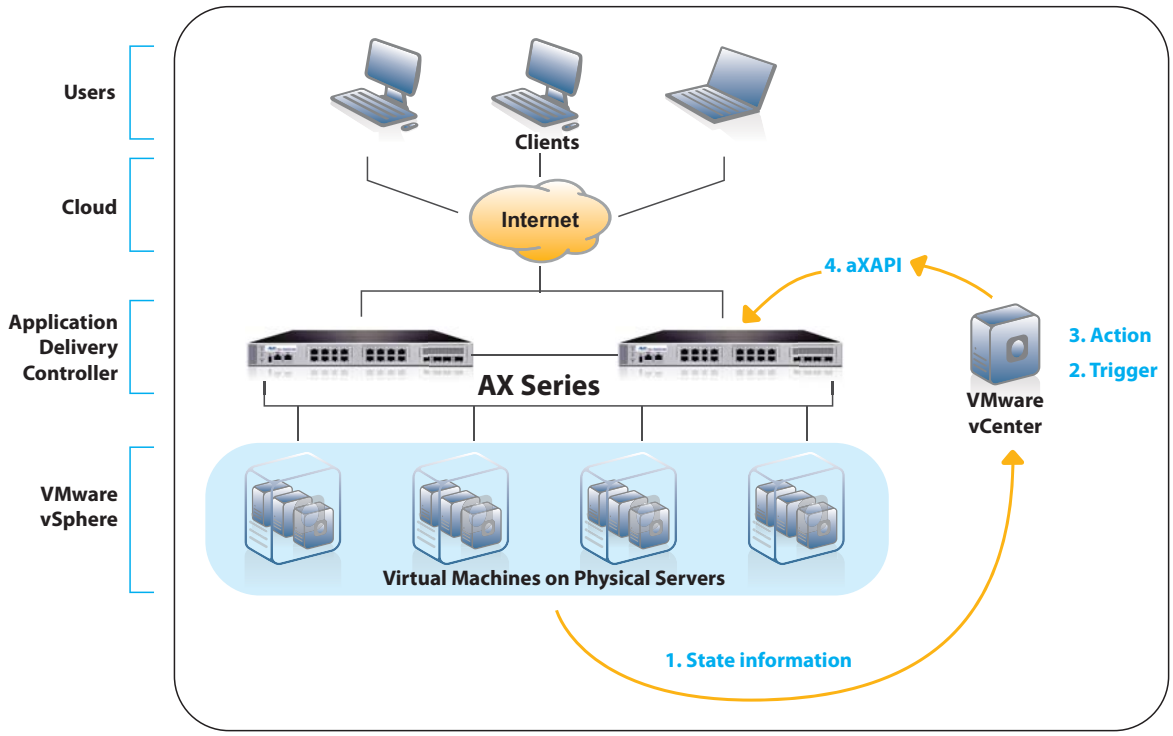


Figure: VMware and AX Series Logical Deployment

Overview of Deployment Steps

The following lists the detailed deployment steps and a brief description of each.

- **Step 1:** In VMware, create web server virtual machine
- **Step 2:** In VMware, setup vCenter and ESX HTTP Proxy
- **Step 3:** In AX, create the AX configuration objects to support dynamic provisioning (which will be the destination for the AX configuration update after the VM addition)
- **Step 4:** Load the aXAPI application “VMPowerOps” on vCenter
- **Step 5:** Configure the aXAPI application “VMPowerOps”
- **Step 6:** In VMware, setup the VM trigger with the application batch program that calls VMPowerOps
- **Step 7:** In VMware, configure the vSphere application trigger for CPU threshold that calls the batch program

Dynamic Provisioning of Application Delivery Services for VMware

The ability to dynamically provision and de-provision virtual machines is a key technology for today's responsive on-demand applications, infrastructure and cloud computing providers. The dynamic support of virtual servers also requires the application delivery infrastructure to pass the traffic to support it. The infrastructure must be flexible and "elastic" enough to expand and contract based on the demands on the infrastructure.

There are seven steps to deploy the solution that creates a joint dynamic framework. Between the virtual machines and the application delivery controllers, they are as follows:

Step 1: Virtual Machine Configuration

Install the required virtual machines for your deployment in the first physical machine or host, and then follow with the second physical machine.

Note: When you have finished the installation, you must install the "VMware Tools." This is important because if you do not have the VMware Tools installed, VMware's process cannot work correctly. To install VMware Tools, right click on the virtual machine and select Guest -> install/upgrade VMware tools. Add all virtual machines to AX's service group as "real server."

Virtual Machine Guests for Web Server

Using the virtual machines setup above, configure each to run a web server on port 80. The exact procedure to do this depends on the operating system and web server software. Alternatively, you can download a pre-configured virtual appliance from the VMware Virtual Appliance Marketplace:

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

Each web server should be configured to have a different IP address and should be accessible from the AX devices.

Step 2: Setup vCenter and ESX HTTP Proxy

The vCenter server and VMware vSphere client 4.0 are required to be installed on your system. By default, the service "VMware VirtualCenter Management Webservices" is not started automatically and therefore you will need to manually launch the service.

To start "VMware VirtualCenter Management Webservices," first navigate to the Start button -> all programs -> control panel -> (switch classic view if needed) -> administrative tools -> services. Find the service which is named "VMware VirtualCenter Management Webservices," then right click on the service and select "start."

Then start the VMware vSphere client 4.0 to check whether the service is working. Login with your windows user account and password and fill in the IP address as localhost if the vSphere is running in the same machine as vCenter, or the address of the vCenter machine if vSphere is running in a different machine.

Note: By default, the vCenter web service only accepts HTTPS connections. Thus the VMAPI may fail to connect to the vCenter. To resolve this, you can either import the certificate for HTTPS or modify the vCenter configuration for HTTP access. We use the second method here (with configuration reference in the section immediately below).

Configure Web Proxy Services on ESX 3.5 for HTTP

1. Log in to the service console as the root user
2. Change directories to /etc/vmware/hostd
3. Use a text editor to open the proxy.xml file
4. Navigate to the list of endpoints in the file (identified by the <EndpointList> tag) that contains settings for the Web service supporting the SDK. The nested tags may look something like this:
   ...
   <e id="1">
   <_type>vim.ProxyService.NamedPipeServiceSpec</_type>
   <accessMode>httpsWithRedirect</accessMode>
   <pipeName>/var/run/vmware/proxy-sdk</pipeName>
   <serverNamespace>/sdk</serverNamespace>
   </e>
   ...
5. Change the access mode to HTTP and HTTPS as follows:
   <accessMode>httpAndHttps</accessMode>
   Or to completely disable HTTPS use the following:
   <accessMode>httpOnly</accessMode>

The Management Network will need to be restarted (but not the host) if changes are made to proxy.xml.

Step 3: AX Configuration

For the AX configuration, the virtual machines are added to the service group. Note down the name of the service group and the service-port as these two parameters will be used later.

Note: Other configurations such as virtual server and real server are needed but not described in this deployment guide.

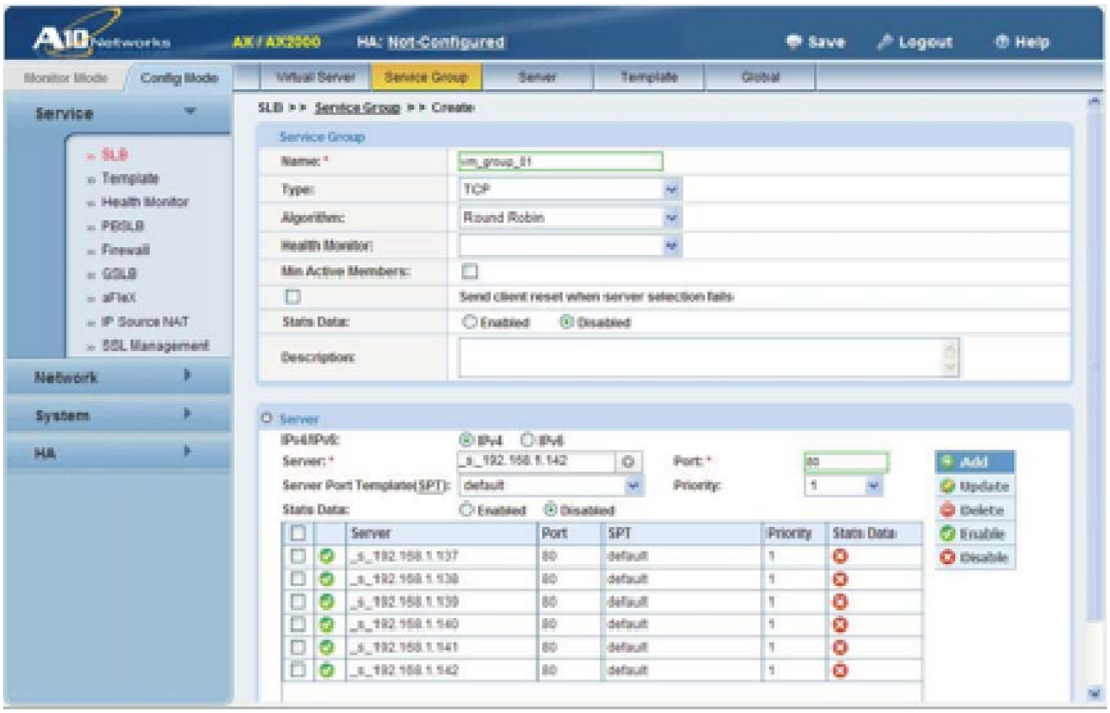


Figure: Service Group for the Virtual Machines

Step 4: Load AX's API Application to vCenter

AX's API, called aXAPI, supports functions such as adding new servers, creating service groups and virtual servers, creating new sessions, and updating service group methods. The aXAPI code for this deployment guide is provided as a separate file called VMPowerOps.cs that is loaded into vCenter. On a host system (with virtual machines), when the resource usage hits a predefined trigger, the vSphere program described will call the application VMPowerOps automatically. This application will do the following:

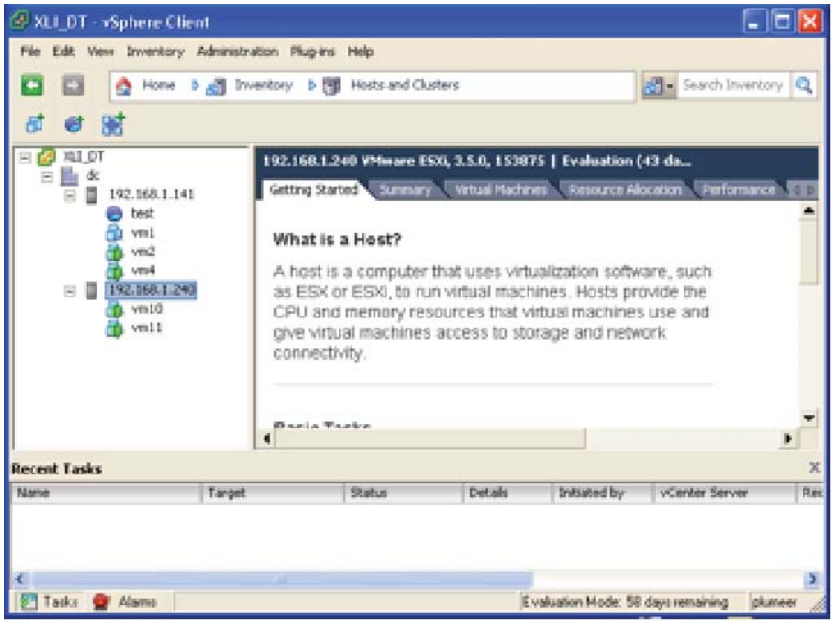
- Finds a powered-off VM in the ESX host, and boots it up
- Adds this VM into the AX service group, so this VM can receive traffic as a "real server" to share the load with the other already operational VMs
- If the application can not find a powered-off VM, then if the total VM number is less than the limit, the application will clone a VM, and then power it on. If the total VM number is more than the limit, the application will do nothing

Note: The application is the VMPowerOps that vSphere calls.

Step 5: Configure aXAPI Application Using vSphere Client

This is the last step in this deployment where the aXAPI application is configured from the vSphere client and associated with the event.

- Bring up the vSphere client and select the correct host.



Step 6: Setup VM Trigger Batch Program that Calls VMPowerOps

Setup the VM Power Trigger in the following format; adjust the variables/arguments to your deployment. ***Note:*** There are a total of 13 parameters and all of them are required; each item is explained after the example:

Example:

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

Variables/arguments definition:

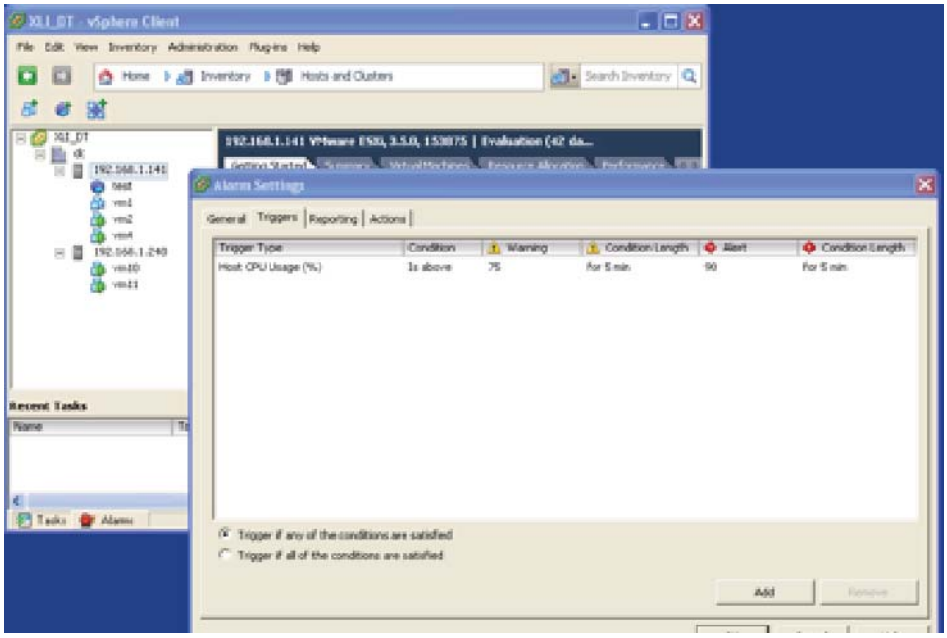
Url < type String >
The URL to login to the DataCenter, you can change the IP address to yours. But keep the /sdk suffix
Username < type String >
The username needed by VMAPI to login to the DataCenter, related to --url
Password < type String >
Relates to –username and –url
DatacenterName < type String >
On which data center the operation will be targeted
MultipleAllow < type String >
Whether we allowed multiple process, 1: Allow, 2: Deny
AXServiceGroup < type String >
The service group name in AX
HostName; < type String >
On which host you want to power on the VM
AXPort < type String >
The port number of the server
AXIP < type String >
The management IP address of the AX device
AXPasswd < type String >
Password for AX login
vmPath < type String >
The path for the clone_template
Threshold < type String >
The threshold for cloning a new virtual machine
AXUsrName < type String >
User name to login AX

When all parameters are entered, write the application name and each parameter to a text file with a .bat extension, which in this case is “Power.bat.”

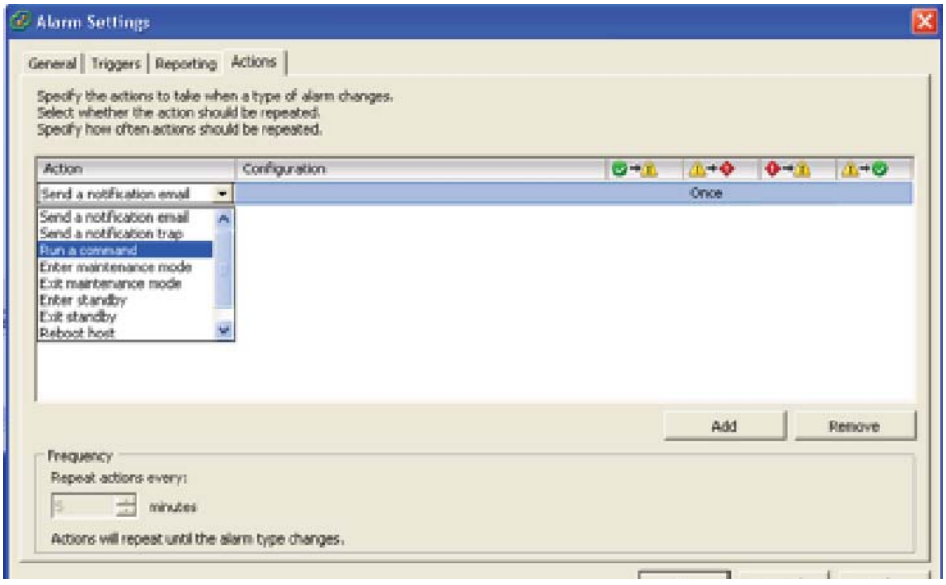
Step 7: Configure the Application and Trigger

Open the vSphere client and login; right click the host that you want to configure in the drop-down list and select Alarm -> Add Alarm.

Give the Alarm a name, click the trigger tab, and click add to add a trigger. Select the proper trigger.



Click the Action tab, click add to add an Action, and in the Action type area, select “Run a command.”



In the configuration area, type the full path of the .bat file, which is “Power.bat.”

For example: C:\windows\system32\cmd.exe /c C:\Debug\Power.bat

When the first host hits the trigger, the system will run the application VMPowerOps and then add the servers dynamically to the AX Series configuration.

Appendix Section: Resource Usage Triggering Program Reference Section

For a host system (with virtual machines), when the resource usage hits a predefined trigger, the vSphere program will call the application automatically. This application will do the following things:

- The program will find a powered-off VM in another host, and boot it up. The corresponding code segment is below. ***Note:*** Refer to the file VMPowerOps.cs for the complete code listing

```
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();
```

This code is provided as an example sample reference; it is not for production operational use.

```
                                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;
        }
    }
```

- Add this virtual machine to the pool of servers in the AX service group, so this VM works as a server to share the services. The corresponding code segment is below

```
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;
      }
```

Note: If there is no virtual machine that can be found that is powered off, then if the total number of virtual machines is less than the virtual machine limit, the application will clone a new virtual machine, and then power it on. If the total virtual machine number is more than the limit, the application will do nothing. Refer to CloneVM VMPowerOn functions in the file VMPowerOps.cs.

Summary and Conclusion

By using the AX Series Application Delivery Controller services, the following key advantages are achieved:

- Ability to adjust the application delivery network capacity in conjunction with VMware virtual machines
- Obtain higher availability with redundant AX Application Delivery Controllers with multiple availability groups to virtual machines
- Obtain higher utilization as the AX can load balance traffic across many virtual machines
- Achieve higher connection throughput and faster end user responsiveness to VMs by offloading compute intensive security processing and employing various other acceleration techniques

By using the AX Series Advanced Traffic Manager, significant benefits are achieved for VMware. For more information about AX Series products, please refer to:

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

About A10 Networks

A10 Networks was founded in 2004 with a mission to provide innovative networking and security solutions. A10 Networks makes high-performance products that help organizations accelerate, optimize and secure their applications. A10 Networks is headquartered in Silicon Valley with offices in the United States, Europe, Japan, China, Korea and Taiwan. For more information, visit www.a10networks.com.

Performance by Design

To learn more about the AX Series Advanced Traffic Manager and how to improve application performance up to 8 times faster while enhancing reliability and security, visit A10 Networks' website at:
www.a10networks.com
Or call and talk to an A10 sales representative:

Corporate Headquarters

A10 Networks, Inc.
2309 Bering Drive
San Jose, CA 95131
Tel: +1 408 325-8668
Fax: +1 408 325-8666

North America Sales:
+1 888 A10-6363
+1 408 325-8616

Europe, Middle East & Africa Sales:
+31 70 799-9143

Asia Pacific Sales:

China, Beijing Office:
+86 10 8515-0698

China, Shanghai Office:
+86 21 6137-7850

Japan Sales:
+81-3-3291-0091

Korea Office:
+82-2-6007-2150
+82-2-6007-2151

Taiwan Office:
+886-2-2657-3198