Getting Started with vSphere Web Services SDK

ESX 4.0
vCenter Server 4.0

Contents

Getting Started with vSphere Web Services SDK

Get started with vSphere Web Services SDK with the help of this guide which will facilitate quick development of applications using the SDK. This guide will take you through the basics of creating your first sample application using vSphere SDK, useful scenarios and some troubleshooting tips. You can then further explore the SDK to control your Virtual Infrastructure.

This information here is for developers who have prior experience with Web services and want to write applications to manage, monitor, and maintain VMware vSphere components using Java or C#.

Introduction

What are vSphere APIs?

The VMware® vSphere APIs provide a way of interfacing with the Virtual Infrastructure. They are exposed as language-neutral web services, hosted on vCenter Server and ESX Server, and are WS-I Basic Profile 1.0 compliant.

The Web service provides all the operations necessary, including life-cycle operations, to monitor and manage virtual infrastructure components like compute resources, virtual machines, networks, storage, and the like.

Why do I need vSphere Web Services SDK?

The vSphere Web Services SDK facilitates the development of the client applications that target the VMware vSphere API. It allows building SOAP-based applications to control ESX Servers / vCenter Servers and virtual machines running on them.

The vSphere Web Services SDK comprises of WSDL files, sample code in Java and C#, various libraries, and all necessary components that are required to work with the vSphere APIs.

What do I need to know?

To get started with vSphere Web Services SDK, you need to have basic knowledge of:

1. A general overview of Virtual Infrastructure,
2. Basic concepts of Web services,
3. vSphere Object Model,
4. And last but not the least, programming and debugging techniques.

What are the various SDKs available?

Since vSphere APIs are exposed as web services, the client application can be written in any programming language that provides mechanism to consume web services.

Other than vSphere Web Services SDK, VMware provides several different SDK products, each intended for different developer communities and target platforms. VMware provides following SDKs:

1. vSphere SDK for Perl
2. VMware vSphere PowerCLI
3. vSphere CLI
4. VMware CIM APIs

SDK Versions and Backward Compatibility

ESX 4.0 and vCenter Server4.0 expose vSphere API version 4.0. However, the vSphere Web Services SDK 4.0 can be used with prior versions of the servers as well (ESX versions 3.5, 3i, 3.0.x,

VirtualCenter versions 2.5, 2.0.x). This is done with the help of appropriate WSDL files. The WSDLs that are shipped with SDK are:

vim.wsdl - This wsdl is to be used in conjunction with ESX 3.0.x and VirtualCenter 2.0.x

vim25.wsdl - This wsdl is to be used in conjunction with ESX 3.5, ESXi 3.5, VirtualCenter 2.5, ESX 4.0, ESXi 4.0 and vCenter Server 4.0

Also, since VI API 2.5, the backward compatibility has been maintained. With backward compatibility, it is meant that the vim25 wsdl will be used for all the future releases of vSphere Servers. As can be seen with release of VI 4, no new wsdl has been introduced. vim25 wsdl caters to older versions (ESX 3.5 or later) as well as the newer ESX Versions released.

The new features introduced in vSphere API 4.0 will not be available for ESX 3.5 and ESXi 3.5, but former features (as available in VI API 2.5) will continue working for ESX 3.5 and ESXi 3.5, without any change in the namespace. For example, “hostProfileManager” is a new property of ServiceContent that is introduced in 4.0. This property will not be available on ESX 3.x servers.

In case your client application requires to support mixed server environments (for example, ESX 3.x and ESX 4.x), then you need to retrieve the API version that is supported on the server and then accordingly design the client-side logic. Section "Supporting Multiple API Versions" of Chapter 6, Client Application Pattern, of vSphere Web Services SDK Programming Guide, provides further details on how this can be achieved.

http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/sdk40programmingguide.pdf

Setting up vSphere Web Services SDK

Where to download?

The SDK is readily available for download at:
http://www.vmware.com/support/developer/vc-sdk/

How to setup the SDK?

After you have downloaded the SDK, the next step is to setup the development environment so that one can start writing the client applications. Here, we only provide reference to the documents that include instructions on setting up development environment for using Java or C# to develop client-side applications.

To setup development environment for Java, refer to Chapter 2, Setting Up for Java Development, of Developer's Setup Guide.

To setup development environment for C#, refer to Chapter 3, Setting Up for Microsoft C# Development, of Developer's Setup Guide.

http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/sdk40setupguide.pdf

Writing your first program

This section explains how to write your first connect application using vSphere SDK. It involves different components and steps that you need to follow while writing your first application.

Connect Application

This section lists the complete standalone connect application both in Java and C#.

In Java

```
package com.vmware.samples.general;

import com.vmware.vim25.*;
import java.net.URL;
import java.util.*;

public class Connect {

    private VimServiceLocator _locator;
      private VimPortType _service;
      private ServiceContent _sic;
      private ManagedObjectReference _svcRef;
      private String  strUrl;
      private String  strUser;
      private String  strPassword;


      Connect(String url, String user , String pwd) throws
Exception
      {
            strUrl = url;
            strUser = user;
            strPassword = pwd;

            _svcRef = new ManagedObjectReference();
            _svcRef.setType("ServiceInstance");
            _svcRef.set_value("ServiceInstance");

            _locator = new VimServiceLocator();
            _locator.setMaintainSession(true);
            _service = _locator.getVimPort(new URL(strUrl));

            _sic = _service.retrieveServiceContent(_svcRef );
      }

   private void Login() throws Exception
   {
            _service.login(_sic.getSessionManager(), strUser,
strPassword, null);
      }
```

```
public static void main(String[] args) throws Exception {
      try {
             System.setProperty("axis.socketSecureFactory",

"org.apache.axis.components.net.SunFakeTrustSocketFactory");
         Connect connObj = new Connect (args[0],args[1],args[2]);
         System.out.println("Connecting...");
         connObj.Login();
         System.out.println("Done!!!");
             System.out.println("Press any key to Exit");
         System.in.read();
      }
      catch(Exception e) {
         System.out.println("Connection UnSuccesful.");
         throw e;
      }
   }
}
```

In C#

```
using System;
using System.Net;
using System.Net.Security;
using System.Security.Cryptography.X509Certificates;
using System.Web.Services.Protocols;
using VimApi;

namespace ConnectSample
{
    class Connect
    {
        private ManagedObjectReference svcRef;
        private VimService service;
        private ServiceContent sic;
        private UserSession session;
        private string strUrl;
        private string strUser;
        private string strPassword;

        public Connect(string url, string user , string pwd )
        {
            strUrl = url;
            strUser = user;
            strPassword = pwd;


ServicePointManager.ServerCertificateValidationCallback =
                    new
RemoteCertificateValidationCallback(ValidateServerCertificate);
            svcRef = new ManagedObjectReference();
            svcRef.type = "ServiceInstance";
            svcRef.Value = "ServiceInstance";

            service = new VimApi.VimService();
            service.Url = strUrl;
            try
            {
                sic = service.RetrieveServiceContent(svcRef);
            }

            catch (Exception ex)
            {
                Console.WriteLine("Caught Exception: " +
ex.Message);
            }
        }
      public static bool ValidateServerCertificate(Object sender,
            X509Certificate certificate,
            X509Chain chain,
            SslPolicyErrors sslPolicyErrors)
        {
            return true;
        }
```

```
public void Login()
        {
            try
            {
                session = service.Login(sic.sessionManager,
strUser, strPassword, null);
            }

            catch (Exception ex)
            {
                Console.WriteLine("Caught Exception: " +
ex.Message);
            }
        }

public static void Main(string[] args)
        {
            Connect connObj = new Connect(args[0], args[1],
args[2]);
            Console.WriteLine("Connecting...");
            connObj.Login();
            Console.WriteLine("Done!!!");
            Console.WriteLine("Press any key to Exit");
            Console.ReadLine();
        }
    }
}
```

A Closer Look at the "Connect" Application

The general concept of API is the method written in a class which is being exposed through a web server to the external world. The external world can consume these APIs in our client application to perform tasks. If you are calling an API you need an object to consume it. At the same time these APIs may or may not return some data/object. Handling the primitive data as a return value is easy in client application (from where you are calling these API's). But, when an API is returning an object it is necessary that the client application understands it and works accordingly on those objects in the code. To achieve this, we need something at the client side which interprets the API's returning object. So, here comes the concept of stubs. These stubs are generated using the WSDL files and are shipped with VI SDK.

When you are writing your application you need to include the jar file (in Java) and DLL (in C#) in your application to get the reference of the complex objects returned by APIs.

There are two types of objects in VMware Object Model

1. Managed objects
2. Data objects

Managed object represents a server side object. It represents either an entity (data center, host system, virtual machine etc.) or a service (session manager, property collector etc.).

The first thought which comes in your mind is that if API can be invoked through a managed object, then how can we get the managed object reference at client side which can be used to call vSphere API's

Here is the answer to your query!

1. The ServiceContent data object contains the managed object references to all the top level controller (service provider) objects.
2. As, ServiceContent data object is the return type of API RetrieveServiceContent, we need to call this API to get the reference of all available managed objects.
3. Again, for this we need an object to call RetrieveServiceContent API and for this we should use the ServiceInstance managed object reference.
4. So, the ServiceInstance managed object reference is the entry point in vSphere API's.

We can create a client side reference of ServiceInstance and use it further to get the all other managed object reference. RetrieveServiceContent method is available on ServiceInstance managed object and returns the ServiceContent data object leading to the entry in VI SDK.

To explore vSphere API's and the usage of its API one needs to perform the steps in the following order:

- Ensure that you included the vimService jar/dll into the application.
- An instance of ServiceInstance managed object is created
- Web service port connection is created using the url.
- Retrieval of ServiceContent data object using the ServiceInstance managed object reference (using retrieveServiceContent API)
- Use the Login API to establish a session with the vCenter Server / ESX host. (SessionManager is the Managed Object that includes methods for logging on and logging off clients and maintains the session with ESX/VC to execute the API's to get the correct information and execute the tasks.)
- Invoke the right API with the use of appropriate controller object
- Session termination using the logout API

We have gone through the steps required to create the connect application.
Now it's time to go a little deep inside the code and elaborate on different sections of the code. The "Connect" application consists of three primary components: libraries, definition of Connect class and main method. The following explanation will provide you with a basic understanding of the code.

Libraries

Include the dll/jar into the project. This reference needs to be included in the code as this contains stubs of all the objects and methods associated with them.

In Java:

```
import com.vmware.vim.*;
```

In C#:

```
using VimApi;
```

Connect class

This class involves different objects and invoking methods to make a connection with ESX/VC based on the parameter passed at the time of execution of the program.

Variable declaration:

In Java:

```
private VimServiceLocator _locator;
private VimPortType _service;
private ServiceContent _sic;
private ManagedObjectReference _svcRef;
```

In C#:

```
private ManagedObjectReference svcRef;
private VimService service;
private ServiceContent sic;
private UserSession session;
```

The constructor of the class is created as follows:

1. Initializing the variable passed as a command line argument
   ```
   strUrl = url;
   strUser = user;
   strPassword = pwd;
   ```

2. A Managed Object Reference of ServiceInstance. This reference, that is, svcRef ManagedObjectReference will be used with API RetrieveServiceContent to get the ServiceContent data object.

   In Java:

   ```
   _svcRef = new ManagedObjectReference();
   _svcRef.setType("ServiceInstance");
   _svcRef.set_value(“ServiceInstance”);
   ```

   In C#:

   ```
   svcRef = new ManagedObjectReference();
   svcRef.type = "ServiceInstance";
   svcRef.Value = "ServiceInstance";
   ```

3. VimService class has the client side implementation of all the APIs. We need to create an object of VimService class to call the API RetrieveServiceContent.

   In Java:

   ```
   _locator = new VimServiceLocator();
   _locator.setMaintainSession(true);
   _service = _locator.getVimPort(new URL(strUrl));
   ```

In C#:

```
service = new VimApi.VimService();
service.Url = strUrl;
```

4. Calling the RetrieveServiceContent API to get the ServiceContent data object. As all the client side implementation is defined in VimService call, we should use this class object to call the RetrieveServiceContent API.

   In Java:

   ```
   _sic = _service.retrieveServiceContent(_svcRef );
   ```

   In C#:

   ```
   sic = service.RetrieveServiceContent(svcRef);
   ```

   Now, we get the ServiceContent data object that contains the managed object references to all the top level controllers (service providers)

5. Finally, call the login method using the service object (object of VimService class)

   In Java:

   ```
   _service.login(_sic.getSessionManager(), strUser, strPassword,
   null);
   ```

   In C#:

   ```
   session = service.Login(sic.sessionManager, strUser, strPassword,
   null);
   ```

Main Method

In Java/C# programming language, every application must contain a main method whose signature is:

```
public static void main(String[] args)
```

The main method is the entry point for your application and will subsequently invoke all the other methods required by your program.

The main method accepts a single argument: an array of elements of type String.

This main method is creating an object of Connect class.

In Java:

```
Connect connObj = new Connect (args[0], args[1], args[2]);
```

In C#:

```
Connect connObj = new Connect(args[0], args[1], args[2]);
```

args[0] – url of the ESX/VC
args[1] – userid of ESX/VC
args[2] – password of ESX/VC

Once the Object of Connect Class is created we can call the Login method.

In Java:

```
connObj.Login();
```

In C#:

```
connObj.Login();
```

Use Cases

This section describes usage of some of the APIs that might be useful for certain scenarios. They describe steps required in order to successfully use the APIs.

Note: The sample scripts provided in this section uses the client-side libraries shipped along the vSphere SDK (apputils.jar for Java and AppUtil.dll for C#)

1. Creating a Virtual Machine

Virtual Machine is created under a folder using the “Folder” object. It needs to be associated with a resource pool from which it will obtain the required resources. For unmanaged ESX/ESXi systems, you need not specify the host. If the host is a member of a cluster, then a host also needs to be associated.

The CreateVM_Task API is used to create the virtual machine. Along with the above parameters, this API also takes VirtualMachineConfigSpec as a parameter. This spec encapsulates the configuration settings of the virtual machine that you want to create. For example, you can provide memory size, number of CPUs, add various virtual devices and allocate shares.

In Java:

```
ManagedObjectReference resourcePool
   = cb.getServiceUtil().getMoRefProp(crmor, "resourcePool");
ManagedObjectReference vmFolderMor
   = cb.getServiceUtil().getMoRefProp(dcmor, "vmFolder");

VirtualMachineConfigSpec vmConfigSpec =
      vmUtils.createVmConfigSpec(cb.get_option("vmname"),
                                 cb.get_option("datastorename"),

Integer.parseInt(cb.get_option("disksize")),
                                 crmor, hostmor);

vmConfigSpec.setName(cb.get_option("vmname"));
vmConfigSpec.setAnnotation("VirtualMachine Annotation");
vmConfigSpec.setMemoryMB(new
Long(Integer.parseInt(cb.get_option("memorysize"))));
vmConfigSpec.setNumCPUs(Integer.parseInt(cb.get_option("cpucount")));
vmConfigSpec.setGuestId(cb.get_option("guestosid"));

ManagedObjectReference taskmor =
cb.getConnection().getService().createVM_Task(
        vmFolderMor, vmConfigSpec, resourcePool, hostmor
);
```

In C#:

```
ManagedObjectReference resourcePool
 = cb.getServiceUtil().GetMoRefProp(crmor, "resourcePool");
ManagedObjectReference vmFolderMor
 = cb.getServiceUtil().GetMoRefProp(dcmor, "vmFolder");

  VirtualMachineConfigSpec vmConfigSpec =
       vmUtils.createVmConfigSpec(cb.get_option("vmname"),
                                  cb.get_option("datastorename"),
                                  int.Parse(cb.get_option("disksize")),
                                  crmor, hostmor);


vmConfigSpec.name=cb.get_option("vmname");
vmConfigSpec.annotation="VirtualMachine Annotation";
vmConfigSpec.memoryMB= (long)(int.Parse(cb.get_option("memorysize")));
vmConfigSpec.memoryMBSpecified = true;
vmConfigSpec.numCPUs = int.Parse(cb.get_option("cpucount"));
vmConfigSpec.numCPUsSpecified = true;
vmConfigSpec.guestId= (cb.get_option("guestosid"));

ManagedObjectReference taskmor = cb.getConnection()._service(
                     vmFolderMor, vmConfigSpec, resourcePool, hostmor);
```

2. Power operations on a Virtual Machine

After you are done with the creation of the Virtual Machine, you will like to perform various power operations on the VM. For instance, you will be required to power on your virtual machine to install applications. You can perform following operations on a virtual machine:

- Power On
- Power Off
- Suspend
- Resume
- Reset
- Shut down guest
- Restart guest

Please note, for “Shut down guest” and “Restart guest” operations, you need to have VMware Tools installed.

Here, only powering on a virtual machine has been covered. You can refer to VMPowerOps.java for Java sample under *SDK\samples\Axis\java\com\vmware\samples\vm* and VMPowerOps for C# sample under *SDK\samples\DotNet\cs*, for more power operations.

In Java:

```
String vmName
  = (String)cb.getServiceUtil().getDynamicProperty((
          ManagedObjectReference)vmMOR, "name");
ManagedObjectReference taskmor = null;
System.out.println("Powering on virtualmachine '"+vmName+"'");
taskmor = cb.getConnection().getService().powerOnVM_Task(vmMOR, null);
```

In C#:

```
ManagedObjectReference vmMOR = cb.getServiceUtil().GetDecendentMoRef(null,
                                                "VirtualMachine",
                                                vmName);
ManagedObjectReference taskmor = null;
taskmor = cb.getConnection()._service.PowerOnVM_Task(vmMOR, null);
```

3. Reconfigure Virtual Machine

After a virtual machine has been created, sometimes it is required to change its configurations. For example, allocate more shares to the VM or change the allocated memory or add another virtual disk. ReconfigVM_Task API is used to modify these settings. Similar to CreateVM_Task API, this API also takes VirtualMachineConfigSpec as a parameter.

The code snippet provided here demonstrates how to change the number of CPUs assigned to a VM. You can refer to VMReconfig.java for Java sample under *SDK\samples\Axis\java\com\vmware\samples\vm* and VMReconfig for C# sample under *SDK\samples\DotNet\cs*, to understand various options to reconfigure a VM.

In Java:

```
VirtualMachineConfigSpec vmConfigSpec = new VirtualMachineConfigSpec();
vmConfigSpec.setNumCPUs(numCpus);
ManagedObjectReference tmor
   = cb.getConnection().getService().reconfigVM_Task(vmMOR, vmConfigSpec);
```

In C#:

```
VirtualMachineConfigSpec vmConfigSpec = new VirtualMachineConfigSpec();
vmConfigSpec.numCPUs = numCpus;
vmConfigSpec.numCPUsSpecified = true;
ManagedObjectReference tmor
   = cb.getConnection()._service.reconfigVM_Task(vmMOR, vmConfigSpec);
```

4. Searching files related to a Virtual Machine

SearchDatastoreSubFolders_Task API can be used to retrieve all the files information for a Virtual Machine. SearchDatastoreSubFolders returns the information for the files that match the given search criteria specified in the searchSpec as a SearchResults[] object. Searches the folder specified by the datastore path and all subfolders.

One must not specify searchSpec to retrieve all the files in the specified folder.

In Java

```
ManagedObjectReference dsBrowser =
(ManagedObjectReference)cb.getServiceUtil().getDynamicProperty(hostMOR,
"datastoreBrowser");
String dsPath = "[" + dsName + "]";
ManagedObjectReference task =
cb.getConnection().getService().searchDatastoreSubFolders_Task(dsBrowser,
dsPath, null);
String taskRes = cb.getServiceUtil().waitForTask(task);

    if(taskRes.equals("sucess")) {
       HostDatastoreBrowserSearchResults[] searchResults =
(HostDatastoreBrowserSearchResults[])cb.getServiceUtil().getDynamicProperty(
task,   "info.result");

        if(searchResults != null) {
           for(int a=0;a <searchResults.length; a++){
                System.out.println("Files in path " +
searchResults[a].getFolderPath() + " are:");
                FileInfo[] files = searchResults[a].getFile();
                for(int i=0; i < files.length; i++) {
                    System.out.println(i+1 + ": " + files[i].getPath());
                }
           }
        }
    }
```

In C#

```
ManagedObjectReference dsBrowser =
(ManagedObjectReference)cb.getServiceUtil().getDynamicProperty(hostMOR,
"datastoreBrowser");
String dsPath = "[" + dsName + "]";
ManagedObjectReference task =
cb.getConnection().getService().searchDatastoreSubFolders_Task(dsBrowser,
dsPath, null);
String taskRes = cb.getServiceUtil().waitForTask(task);

    if(taskRes.equals("sucess")) {
       HostDatastoreBrowserSearchResults[] searchResults =
(HostDatastoreBrowserSearchResults[])cb.getServiceUtil().getDynamicProperty(
task,   "info.result");

        if(searchResults != null) {
           for(int a=0;a <searchResults.length; a++){
                System.out.println("Files in path " +
searchResults[a].getFolderPath() + " are:");
                FileInfo[] files = searchResults[a].getFile();
                for(int i=0; i < files.length; i++) {
                    System.out.println(i+1 + ": " + files[i].getPath());
                }
           }
        }
    }
```

Note: Incase your virtual machine disks reside on multiple datastores then one has to call SearchDatastoreSubFolders_Task API for all those datastores.

5. Selection of a LUN while creating a RDM disk

To Filter ScsiLun on which RDM can be created, one must use the following approach:

a) Get the EnvironmentBrowser of the ComputeResource object for the HostSystem under which you are adding Virtual Machine
b) Get ConfigTarget by calling QueryConfigTarget() on EnvironmentBrowser.
c) Get configTarget.ScsiDisk which is a VirtualMachineScsiDiskDeviceInfo[].
d) Each of these VirtualMachineScsiDiskDeviceInfo will contain the LUN information which is RDM capable.
e) VirtualMachineScsiDiskDeviceInfo.Disk will give HostScsiDisk type object.
f) Use SCSIDiskDeviceInfo.Disk.CanonicalName to get the devicename for the LUN

In Java

```
Object parObj = cb.getServiceUtil().getDynamicProperty(hostmor, "parent");

ManagedObjectReference comRes = (ManagedObjectReference)parObj;
if(comRes.getType().equalsIgnoreCase("ComputeResource")){
   System.out.println("ComputeResource is " +
comRes.get_value().toString());
   Object envBrObj = cb.getServiceUtil().getDynamicProperty(comRes,
"environmentBrowser");
   ManagedObjectReference envBrow = (ManagedObjectReference)envBrObj;
   ConfigTarget ctar =
cb.getConnection().getService().queryConfigTarget(envBrow,hostmor);
   VirtualMachineScsiDiskDeviceInfo[] arrSCSILun =
(VirtualMachineScsiDiskDeviceInfo[])ctar.getScsiDisk();
   if(arrSCSILun != null){
      for(int i=0; i<arrSCSILun.length; i++) {
          deviceName = arrSCSILun[i].getDisk().getCanonicalName();
      }
      System.out.println("HostScsiDisk deviceName is " + deviceName);
   }
   else{
      System.out.println("No HostScsiDisk supporting RDM is found ");
      return;
   }
}
```

In C#

```
Object parObj = cb.getServiceUtil().GetDynamicProperty(hostmor, "parent");
ManagedObjectReference comRes = (ManagedObjectReference)parObj;

if (comRes.GetType().Equals("ComputeResource"))
{
    Console.WriteLine("ComputeResource is " + comRes.Value);
    Object envBrObj = cb.getServiceUtil().GetDynamicProperty(comRes,
"environmentBrowser");
    ManagedObjectReference envBrow = (ManagedObjectReference)envBrObj;
    ConfigTarget ctar =
cb.getConnection()._service.QueryConfigTarget(envBrow, hostmor);
    VirtualMachineScsiDiskDeviceInfo[] arrSCSILun =
(VirtualMachineScsiDiskDeviceInfo[])ctar.scsiDisk;
    if (arrSCSILun != null)
    {
        for (int i = 0; i < arrSCSILun.Length; i++)
        {
            deviceName = arrSCSILun[i].disk.canonicalName;
        }
        Console.WriteLine("HostScsiDisk deviceName is " + deviceName);
    }
    else
    {
        Console.WriteLine("No HostScsiDisk supporting RDM is found ");
        return;
    }
}
```

There are some pre-requisites for the LUN to be RDM capable, which are as follows:

1. The ScsiLun is attached to the system through a shared HostBusAdapter. At least one of the HostBusAdapters that expose the ScsiLun should be shared [FibreChannel or iSCSI]
2. The ScsiLun should not be a part of VMFS volume.
3. The ScsiLun should not already be used as a RDM target by any virtual machine.

Apart from this, RDM uses serial number to identify the mapped device. And if the disk LUN doesn't expose serial number then that cant be used for RDM.

6. Searching for a specific virtual machine configuration option

QueryConfigOption API can be used to know the supported virtual machine configuration. It returns VirtualMachineConfigOption data object which contains the execution environment for a virtual machine. This includes information about the features that are supported, such as:

1. Which guest operating systems are supported?
   VirtualMachineConfigOption data object has a property GuestOsDescriptor, which provides an array of GuestOsDescriptors holding the supported Guest Operating Systems.
2. Which adapters are supported for a Virtual machine
   GuestOsDescriptor also contains the recommendedEthernetCard and supportedEthernetCard.

   In Java

```
Object envBrObj = cb.getServiceUtil().getDynamicProperty(comRes,
"environmentBrowser");
ManagedObjectReference envBrow = (ManagedObjectReference)envBrObj;
VirtualMachineConfigOption vmConOpt =
cb.getConnection().getService().queryConfigOption(envBrow,null,
hostmor);
GuestOsDescriptor[] gOSDes = vmConOpt.getGuestOSDescriptor();
    for(int i = 0; i < gOSDes.length; i++){
        System.out.println("Guest OS " + i + "is: " +
gOSDes[i].getFullName());
        System.out.println("\t List of Supported EthernetCard are:
");
        String[] arrSupEC = gOSDes[i].getSupportedEthernetCard();
          for (int s = 0; s < arrSupEC.length; s++){
              System.out.println("\t \t" + arrSupEC[s]);
          }
    }
```

   In C#

```
Object envBrObj = cb.getServiceUtil().GetDynamicProperty(comRes,
"environmentBrowser");
    ManagedObjectReference envBrow =
(ManagedObjectReference)envBrObj;
    VirtualMachineConfigOption vmConOpt =
cb.getConnection()._service.QueryConfigOption(envBrow, null,
hostmor);
    GuestOsDescriptor[] gOSDes = vmConOpt.guestOSDescriptor;

    for(int i = 0; i < gOSDes.Length; i++){
        Console.WriteLine("Guest OS " + i + "is: " +
gOSDes[i].fullName);
        Console.WriteLine("\t List of Supported EthernetCard are: ");
        String[] arrSupEC = gOSDes[i].supportedEthernetCard;

        for(int s = 0; s < arrSupEC.Length; s++){
            Console.WriteLine("\t \t" + arrSupEC[s]);
        }
    }
```

7. To Create a VMFS Datastore

1. First QueryAvailableDisksForVmfs API must be called to get the list of disks that can be used to contain VMFS datastore extents.
2. We must then select a hostScsiDisk and pass it devicePath as a parameter to QueryVmfsDatastoreCreateOption. QueryVmfsDatastoreCreateOptions returns the options for creating a new VMFS datastore for a disk. There are often multiple ways in which extents can be allocated on a disk. Each instance of this structure represents one of the possible options that can be applied to provisioning VMFS datastore storage. Select one of them
3. Then pass it as spec for CreateVmfsDatastore API call.

In Java

```
   ManagedObjectReference HostDatastoreSystemRef =
(ManagedObjectReference)cb.getServiceUtil().getDynamicProperty(hostmor,
"configManager.datastoreSystem");
    HostScsiDisk[] scsiDisks =
cb.getConnection().getService().queryAvailableDisksForVmfs(HostDatastoreS
ystemRef, null);
    for(int j = 0; j < scsiDisks.length; j++){
        VmfsDatastoreOption[] dsOptions =
cb.getConnection().getService().queryVmfsDatastoreCreateOptions(HostDatas
toreSystemRef, scsiDisks[j].getDevicePath());
        if (dsOptions != null && dsOptions.length > 0){
            VmfsDatastoreCreateSpec vmfsSpec =
(VmfsDatastoreCreateSpec)dsOptions[0].getSpec();
vmfsSpec.getVmfs().setVolumeName(dsName);
            ManagedObjectReference newDS =
cb.getConnection().getService().createVmfsDatastore(HostDatastoreSystemRe
f, vmfsSpec);
        }
    }
```

In C#

```
    ManagedObjectReference HostDatastoreSystemRef =
(ManagedObjectReference)cb.getServiceUtil().GetDynamicProperty(hostmor,
"configManager.datastoreSystem");
    HostScsiDisk[] scsiDisks =
cb.getConnection()._service.QueryAvailableDisksForVmfs(HostDatastoreSyste
mRef, null);
    for (int j = 0; j < scsiDisks.Length; j++){
        VmfsDatastoreOption[] dsOptions =
cb.getConnection()._service.QueryVmfsDatastoreCreateOptions(HostDatastore
SystemRef, scsiDisks[j].devicePath);
        if (dsOptions != null && dsOptions.Length > 0){
            VmfsDatastoreCreateSpec vmfsSpec =
(VmfsDatastoreCreateSpec)dsOptions[0].spec;
            vmfsSpec.vmfs.volumeName = dsName;
            ManagedObjectReference newDS =
cb.getConnection()._service.CreateVmfsDatastore(HostDatastoreSystemRef,
vmfsSpec);
        }
    }
```

Troubleshooting Tips

Below are some common issues that are faced. This section provides steps that can be taken to resolve them.

A) Are you seeing frequent connection timeouts?

The default connection timeout is 30 minutes and requires you to re-establish a connection. Ideally if you are not using the session for long time, you should logout and re-login when needed as an active session costs resources on the server side, and will not be good for the overall performance.

If you are still interested in avoiding the timeout, you may optionally use a thread to run a command before 30 minutes to avoid expiry.

B) Server rejects the login requests

The hostd on the ESX server side can take a limited number of user sessions at a time. On exceeding this limit, the server will reject the remaining requests. You may see the SOAP session count limit reached error message. The session limit is set to 100.

C) Building stubs in Microsoft C# environment throws following error "No Visual Studio 2005 environment settings found".

Ensure that you have properly set the environment variables. After setting the environment variables, make sure that you close the current window on which you had been executing the command and open a new command prompt in the start menu under the "Microsoft Visual Studio 8" program group.

D) Do you notice any SocketTimeout Exception on the client side when using waitforUpdates?

For asynchronous tasks, such as CloneVM_Task, one needs to have to block their thread themselves which may be achieved through WaitForUpdates method.

Now, if there is a thread blocked on 'waitForUpdates' and there are no updates for given time out period (this time out period is set on the client stubs - default is 10 minutes), the call throws Remote Exception with SocketTimeOut exception as the cause.

One needs to handle this error on client side by invoking waitforupdates call again.

E) How can I collect logs in order to troubleshoot the error?

You can refer to the below KB article:
http://kb.vmware.com/kb/1001457
This article provides procedures for obtaining diagnostic information for vSphere SDK.

F) “NotSupported” fault is thrown when executing an operation on the server.

All methods available in vSphere API might not be supported on the target server (ESX server / vCenter Server). Section A, Managed Object Privileges Reference, in vSphere Web Services SDK Programming Guide provides the list of operations that are supported on vCenter Server and ESX Server.

G) Unable to find a type, method or property in vSphere API that was available earlier

As of vSphere 4.0, many APIs have been deprecated. You can refer to the Deprecations Summary to identify whether the API has been deprecated and the equivalent available in vSphere API 4.0

http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/vsdk400deprecations.html

Support Offerings

One can search the Knowledge Base, use Community Forums or review Documentation resources to find ready answers to some of the queries.

VMware also offers different types of support offerings targeting different business needs. Please visit following site for more details:

http://www.vmware.com/support/services/

Quick References

Developer's Setup Guide
http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/sdk40setupguide.pdf

vSphere Web Services SDK Programming Guide
http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/sdk40programmingguide.pdf

vSphere API Reference
http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/index.html

Deprecations List
http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/vsdk400deprecations.html

VMware Knowledge Base
http://kb.vmware.com

VMware APIs and SDKs Documentation
http://www.vmware.com/support/pubs/sdk_pubs.html

VMware Technical Resources
http://www.vmware.com/resources/techresources/

VMware Developer Community
http://www.vmware.com/developer

Existing sample codes posted by community users
http://communities.vmware.com/community/developer/utilities