

Hello World – using vSphere Web Services SDK – By Balaji Parimi

Download Sample Code: <http://blogs.vmware.com/files/getcurrenttime-1.java>

If you know how to write web services clients, you know how to program using the vSphere Web Services SDK. vSphere APIs are exposed as web services, compliant with WS-I BP 1.0. vSphere Web Services SDK consists of the WSDL files (containing the vSphere API, and data object definitions), client side stubs (Java) generated by Axis 1.4, sample programs in Java and C#, and reference documentation. The goal of this is to explain how to write a simple Hello World equivalent program using the vSphere APIs.

A good understanding of the object model is fundamental to writing the client programs. vSphere APIs consist of managed objects, and data objects.

Data object represents information about the managed objects. It is passed by value to the client. It is defined in the WSDL schema.

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, root folder etc.). It is not defined in the WSDL schema. All the API invocations require a managed object reference. Managed object reference is an index for the server to identify the appropriate managed object.

Different ways of getting managed object references:

- Using constructor

Typically the client applications use this method only once, during the creation of *ServiceInstance* managed object. This is necessary since every API invocation requires a managed object reference.

- Accessing the property of a data object

Data object can contain managed object references as its properties. Simply using the accessor method for that property can fetch the managed object reference.

- Invoking an API

When an API is invoked, it can return a managed object reference.

- Using the property collector

Property collector provides a way to retrieve and monitor properties of managed objects and the managed object references.

- Using *SearchIndex* APIs

SearchIndex APIs (*findByDnsName*, *findByIp* etc.) can return a managed object references.

All the clients start with creating *ServiceInstance* managed object reference using the constructor. This is used to obtain *ServiceContent* data object. *ServiceContent* data object contains the managed object references to all the top level controller (service provider) objects. Most of them are singleton, and the reference guide contains more details about these. Some examples:

rootFolder – managed object reference to the top of the inventory

sessionManager – managed object reference for managing client session

propertyCollector – session specific managed object reference for retrieving and monitoring the properties of managed objects

Every client program goes through the following steps in order to perform any function using the vSphere APIs.

- Create an instance of *ServiceInstance* managed object
- Connect to the web service port
- Obtain *ServiceContent* data object using the *ServiceInstance* managed object reference (using *retrieveServiceContent* API)
- Establish session with the vCenter Server / ESX host using the *login* API. The *ServiceContent* data object contains managed object references to all of the controller objects required for invoking any API
- Use the appropriate controller object to invoke the right API
- Terminate the session using the *logout* API

Code snippets in Java illustrating the aforementioned steps:

//Create an instance of the *ServiceInstance* managed object

```
ManagedObjectReference siMoref = new ManagedObjectReference();
```

```
siMoref.setType("ServiceInstance");
```

```
siMoref.set_val("ServiceInstance");
```

//Connect to the web service port

```
VimServiceLocator locator = new VimServiceLocator();
```

```
locator.setMaintainSession(true);
```

```
VimPortType vimPort = locator.getVimPort(new URL(https://<HOST\_NAME or IP>/sdk));
```

//Retrieve *ServiceContent* data object

```
ServiceContent sc = vimPort.retrieveServiceContent(siMoref);
```

//Establish session with the server (vCenter or ESX)

```
vimPort.login(sc.getSessionManager(), user, password, "en");
```

//After establishing session any API can be invoked. For simplicity we invoke the current server time.

//The API required passing a managed object reference and the invocation fails if the session is not established.

```
Calendar cal = vimPort.currentTime(SIMO_REF);
```

//Terminate the session

```
vimPort.logout(sc.getSessionManager());
```

Here is a simple vSphere API client program written in Java to retrieve the current time from the server with all the aforementioned steps. You can replace `getCurrentTime` method in the attached code sample with any other method to invoke any API or you can add any number of methods to invoke any API.

The following code sample is created using vSphere Web Services SDK 4.0. You can use the jar files in the `lib` directory of the SDK distribution to compile and run it.