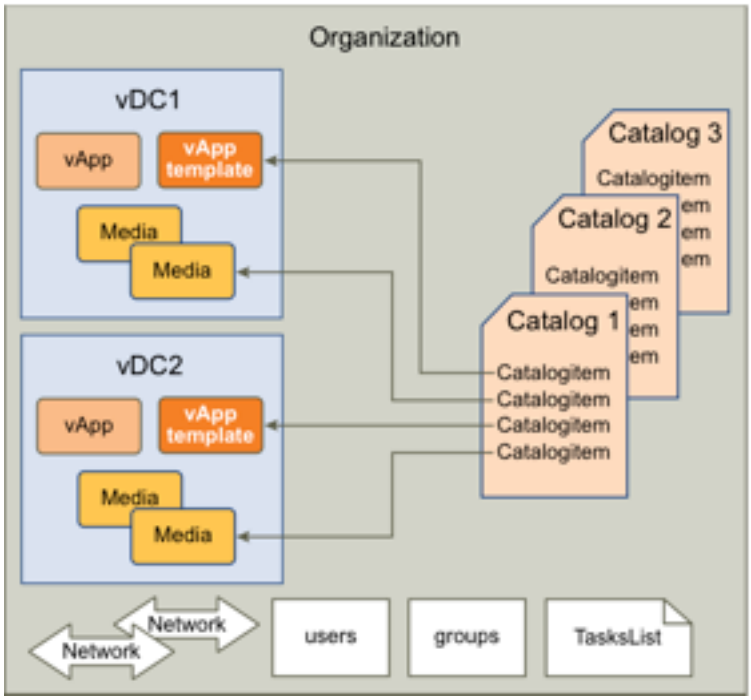
Backup Design for vCloud Tenant vApps

This document introduces software developers to the concepts and procedures necessary to create backup and restore solutions for vCloud Director.

The multi-tenant and self-service capabilities of vCloud Director provide multiple levels of protection for a vApp. A service provider can offer vApp protection at the system level, the tenant level, or the end-user level. These functions are managed by a system administrator, an Organization administrator, and an end user, respectively. This document focuses particularly on the protection provided at the system level, where service providers use backup solutions from data protection software vendors.

Figure 1 shows the objects within a single organization that you can access with the vCloud API. This document describes only how to design software to back up and restore the vApps.

Figure 1. vCloud API Object Taxonomy



A vCloud vApp is a management construct linked to one or more virtual machines in a vSphere environment. To back up or restore a vApp, you need to deal with both the vCloud configuration and the actual virtual machines that belong to the vApp. In vSphere, a virtual machine is represented by configuration files and virtual disk files.

What You Need To Know

You should be familiar with programming concepts, practices, and techniques. You should also be familiar with vCloud, vCloud API, vCloud SDK for .NET, and vSphere concepts. VMware also provides the vCloud SDK for Java and the vCloud SDK for PHP, but this document focuses on .NET for the backup and restore examples.

VMware recommends that you design backup/restore software for the vCloud environment using the following APIs:

Table 1. APIs Used To Back Up vApps

Product	API	Data
vCloud Director	vCloud API or vCloud SDK wrapper	vApp metadata
vSphere	WS API	virtual machine configuration
VDDK	VixDiskLib API or VixMntapi	virtual disk contents

You use the vCloud API or SDK to identify vApp targets for backup and restore operations. The vApp metadata identifies the virtual machines that constitute the vApp. You use the WS API to back up and restore virtual machine configurations. You use the VDDK API to back up and restore virtual disk files.

NOTE This document uses the term "metadata" in a general sense to mean all the vApp configuration data, in addition to user-defined data that the vCloud REST API serializes in the `<Metadata>` element.

You should be familiar with the use of the WS API and the VDDK API for backup and restore of individual virtual machines.

This document does not, in general, duplicate information available in other documents. In particular, this document does not provide details about any storage or data protection API that you need to use for backing up and restoring virtual machines in vSphere. You should consult separate reference documentation for details about specific API calls. See "References" on page 19 for a list of related documentation.

This document emphasizes the use of the vCloud API and SDK for the purpose of managing metadata of the virtual machines and related artifacts in vCloud Director. The vCloud SDK for .NET translates your C# code into REST operations using the vCloud API.

To learn about VMware vCloud and vSphere concepts and usage, refer to the vCloud Director documentation available from the VMware Web site, http://www.vmware.com/support/sdk_pubs.html. You can also visit the VMware SDK community forum at http://communities.vmware.com/community/vmtn.

About This Document

This document is divided into the following main sections:

Conceptual Overview

This section provides background and summarizes the backup and restore processes for vApps managed by vCloud Director.

VMware vCloud Director uses one or more vCenter servers to manage virtualized resources. At the same time, it manages the vCloud feature of multi-tenancy by maintaining metadata related to various tenant artifacts such as vApp, users, networks, storage, and so on.

This document explains how to use VMware APIs to collect the metadata needed to drive backup and restore operations. The actual backup and restore operations are performed at the vSphere layer by using the VMware vStorage APIs for Data Protection (VADP), which are not covered in this document.

When a system administrator chooses to back up a vApp, certain vApp metadata must be retrieved from vCloud Director. The metadata includes general information about the vApp (name, description, virtual machine descriptions), networking information (organization network connectivity, external network connectivity), user information, lease, and quota. This information becomes particularly important when restoring the vApp, in addition to the names of virtual disk files and `.vmx` files typically retrieved from vSphere using the VADP.

The Backup Process

The backup process requires the backup/restore software to collect and store information both from vCloud Director and from vSphere. This process assumes that you use vCloud Director system administrator credentials to connect to vCloud Director. System administrator credentials allow the software to access vApps belonging to any Organization, and to access all the necessary information about a vApp and associated vCloud constructs.

A vApp in vCloud Director can comprise one or more virtual machines. When you work with a single vApp in vCloud Director, you might be working with a number of virtual machines in vSphere.

1. Connect to vCloud Director and access the organization where the vApp or set of vApps is to be backed up.
2. When backing up a vApp for a given Organization or VDC in vCloud Director, access the vCloud Director inventory for a list of all desired vApps.
3. Select maintenance mode for each vApp to prevent updates during the backup process.
4. Collect all the metadata related to the vApp(s), including any user-defined metadata associated with any given vApp.
5. Use the vApp metadata to identify the virtual machines associated with each vApp.
6. Connect to vCenter Server as a user with sufficient permissions to access the virtual machines. Use the vSphere inventory to locate the virtual machine configuration and virtual disk files.
7. Use the VMware APIs for Data Protection to back up the vSphere virtual machine files:
   a. (optional) Save a snapshot of the virtual machine.
   b. Save the virtual machine configuration, using the WS API.
   c. Save the virtual disks using the VDDK API.
   d. (optional) Delete the virtual machine snapshot, if applicable.
8. Store the vApp metadata in an appropriate format along with the associated virtual machine files.
9. Deselect maintenance mode for each vApp.

The Restore Process

The restore process offers some options to the administrator.

When you restore a vApp, you can choose to overwrite an existing vApp. For instance, the restore software might need to overwrite a vApp with data corruption. You can also choose to restore a vApp that no longer exists, for instance, a vApp that was accidentally deleted.

Restoring an Existing vApp

You can choose whether to keep the same vApp name and other vApp attributes, or you can choose to change attributes during the restore process. If the attributes of the restored vApp no longer conform to the environment because of changes since the backup was taken, you can select new values for the non-conforming attributes.

1 Identify the child virtual machines of the vApp, using the metadata stored with the backup.

2 Connect to vCenter Server as a user with sufficient permissions to access the virtual machines and restore the virtual machines in the vSphere environment. This step restores the virtual disk files and virtual machine configuration. If you are overwriting an existing vApp, you generally restore the files to the same data store that vCloud Director currently uses for the vApp.

3 Connect to vCloud Director and authenticate as an administrator, which gives you backup and restore privileges.

4 Locate the corrupted vApp, using the ID retrieved from the metadata in the backup store.

5 Select maintenance mode for the vApp, to prevent changes while restoring metadata.

6 Edit vApp settings such as network, user privileges, lease, and quota as needed. Make sure to include any user-defined metadata from the backup store. If you restored a virtual machine to a different location from the original, you might need to adjust the vApp settings.

7 Deselect maintenance mode for the vApp.

Restoring a Missing vApp

1 Identify the child virtual machines of the vApp, using the metadata stored with the backup.

2 Connect to vCenter Server as a user with sufficient permissions to access the virtual machines and restore the virtual machines in the vSphere environment. This step restores the virtual disk files and virtual machine configuration.

3 Connect to vCloud Director and authenticate as an administrator, which gives you backup and restore privileges.

4 Compose a new vApp or import the virtual machine(s) into vCloud Director to create a new vApp with these characteristics:

   a It has the same name as the lost vApp.

   b It belongs to the same Organization as the lost vApp.

   c It obtains resources from the same provider vDC as the lost vApp.

5 Select maintenance mode for the vApp, to prevent changes while restoring metadata.

6 Edit vApp settings such as network, user privileges, lease, and quota as needed. Make sure to include any user-defined metadata from the backup store.

7 Deselect maintenance mode for the vApp.

NOTE This is a simplified view of the restore process. The exact process you use will depend on the features provided by your software. For instance, if the datastore is full, the software could offer to migrate the vApp to a different datastore.

Use Cases Overview

The following sections give an overview of use cases related to the backup and restore processes.

Managing Credentials

The backup software needs to access vCloud Director to manage vApps at the metadata level, and vCenter Server to manage vApps at the virtual machine and virtual disk level. The backup software must collect and retain authentication credentials for both vCloud Director and vCenter Server.

For more information about authenticating with vCloud Director, see "Getting Access to vCloud Director" on page 6. For more information about authenticating with vSphere, see the *vSphere Web Services SDK Programming Guide*, listed in "References" on page 19.

Finding a vApp

There are different ways to locate a vApp managed by vCloud Director. One way is to traverse the vCloud Director inventory. Another way is to use the query service.

Inventory Traversal

Using the vCloud Director inventory to locate a vApp requires navigating a hierarchy of containers based on organizational and resource divisions. The process is explained in "Inventory Access" on page 7.

Using the Query Service

The vCloud SDK for .NET also supports the query service of the vCloud API for finding vApps. Consult the sample programs in the SDK for more information about how to use the query service in the SDK.

Protecting Specified vApps

Backup systems typically identify vApps to be backed up in a given Organization based on their identity, using vApp attributes such as name and ID or user defined metadata. A set of vApps to be backed up can also be created based on their Organization (for example, all vApps in the Human Resources Organization), the VDC where they are deployed, and so forth.

In all these cases you must traverse the given Organization and its contents to locate and make a list of vApps.

Recovering an Older Version of a vApp

If a vApp has become corrupted, or if users need to revert to an older state of the vApp, the administrator can restore a version of the vApp from backup storage even when the vApp still exists in vCloud Director. The backup/restore application in these cases can access vCloud Director to get vApp identity information and metadata before restoring the backup copy.

The backup/restore application has a choice between overwriting the current vApp instance or deleting it and creating a new vApp. The choice to delete the vApp can be convenient when the vApp configuration has changed since the last backup, especially when a virtual machine has been added to or deleted from the vApp.

Recovering a Deleted vApp

When recovering a deleted vApp, the backup/restore application must identify the vApp from user input to locate the vApp metadata and virtual machine files on the backup storage medium. After the virtual machines have been restored using vSphere APIs, the vApp can be recomposed using the vCloud API. The backup software must first create a vApp from one of the virtual machines, then import the remaining virtual machines into the same vApp.

Recovering a Single Virtual Machine

The process of recovering a single virtual machine from the backup storage medium is a special case of recovering a deleted vApp. In the case of a deleted vApp, the backup software must re-create the vApp in vCloud Director, then import the remaining virtual machines. For a single lost virtual machine, the backup software must only import the one virtual machine into the existing vApp.

Backing Up vCloud Director

The vCloud SDK for .NET does not offer any special features for backing up or restoring the vCloud Director application and its data. Users should follow standard industry advice for protecting Tomcat applications and Oracle or SQL Server databases.

vCloud API Operations

The following sections describe commonly used vCloud API operations using vCloud SDK for .NET. (See "References" on page 19.) The API descriptions in this document do not provide complete backup/restore implementation details, but focus instead on identifying a set of vCloud API methods that facilitate certain operations that use vCloud Director.

You should be familiar with vCloud Director and vCloud API concepts. Every resource in vCloud Director can be accessed using either its unique ID or HREF (the reference URL) in the vCloud API. The .Net SDK provides wrapper utility classes for commonly-used resources to make the programming easier.

The operations described in the following sections are:

- "Getting Access to vCloud Director" on page 6

  Shows how to connect and authenticate with the vCloud API.

- "Inventory Access" on page 7

  Shows how to retrieve data for different Organization types.

- "Retrieving Catalog information" on page 11

  Shows how to retrieve Catalog entries for backup.

- "Retrieving vApp Configuration" on page 12

  Shows how to list virtual machines and vApp configuration data.

- "Preventing Updates to a vApp During Backup or Restore" on page 13

  Shows how to use maintenance mode to quiesce vApp configuration.

- "Associating vCloud Resources with vSphere Entities" on page 14

  Shows how to get Managed Object References of virtual machines and storage resources from vCloud Director.

- "Restoring vApps" on page 17

  Shows how to import virtual machines into vApps.

Getting Access to vCloud Director

The backup/restore software component must use system administrator privileges to connect to vCloud Director, so that it can access any Organization. The system administrator always logs into the System organization. When `Administrator@System` is used as the user name for the API, `Administrator` is the login name and `System` is the System Organization name.

Using system administrator privileges to connect to vCloud Director also allows the backup/restore software to access additional information relating a vApp to the corresponding resources in vSphere. This is described in "Inventory Access" on page 7.

Example 1 shows how to log in using C# with the vCloud SDK for .NET. After logging in, the code shows how to access Organization data.

Example 1. vCloud Director login code sample using Administrator@System/<password>

```
using com.vmware.vcloud.sdk;
using com.vmware.vcloud.api.rest.schema;

public static vCloudClient client = null;
client = new vCloudClient(vCloudURL,com.vmware.vcloud.sdk.constants.Version.V1_5);
client.Login(username, password);
// Get references to all Organizations:
Dictionary<string,ReferenceType> organizationsMap = client.GetOrgRefsByName();
// Get reference to a specific Organization:
string orgName = "Org1";
ReferenceType orgRef = client.GetOrgRefByName(orgName);
// Convert Organization reference to Organization object:
Organization org = Organization.GetOrganizationByReference(client, orgRef);
```

Inventory Access

In general, you locate a desired vApp for backup in the context of a given Organization and vDC. To locate a vApp that you want to back up, you first need a reference to its parent Organization.

You use the Organization reference to get the Organization object, which you use to get a list of references to the vDCs that belong to the Organization. You use a vDC reference to get a vDC object, which you then use to get a list of references to the vApps that belong to the Organization. You convert the desired vApp reference to a vApp object, which you use to list the virtual machines that belong to the vApp.

Example 1 shows how to get a reference to the user view of an Organization. Example 2 shows how to get a reference to the admin view of an Organization and a vDC.

Admin Views

The admin view of resources such as Organization, vDC, and vApp provides extra information that is useful to users with administrative privileges. For example, in the case of a vApp, the admin view provides information about vCenter and the virtual machines that belong to the vApp. The admin view provides information such as Managed Object References that vCenter uses for those entities. See "Associating vCloud Resources with vSphere Entities" on page 14 for more information about getting vCenter Managed Object References.

To access admin views, you use a method of the client connection object to create an admin client proxy. The admin proxy has methods similar to those of the client connection object to get references to Organizations and other vCloud objects. However, the objects you get from the admin proxy have additional properties not present in user objects.

Example 2. Get Admin Org and Admin vDC

```
public static vCloudClient client = null;
// Login
…
// Get admin view of Org
VcloudAdmin admin = client.GetVcloudAdmin();
string orgName = "Org1";
ReferenceType orgRef = admin.GetAdminOrgRefByName(orgName);
AdminOrganization adminOrg = Organization.AdminGetOrgByReference(client, orgRef);

// Get admin vDC
string vdcName = "VDC1";
ReferenceType vdcRef = adminOrg.GetAdminVdcRefByName(vdcName);
…
AdminVdc adminVdc = AdminVdc.GetAdminVdcByReference(client, vdcRef);
```

Admin Extensions

Similar to the admin views, you can use a different method of the client connection object to create an admin extension client proxy. You use the admin extension proxy to find provider vDCs. A provider vDC includes one or more resource pools and allocates resources from those pools to the Org vDCs that it supports.

Example 3 shows how to get a Provider vDC.

Example 3. Get Provider vDC

```
// Login
…
// Get dictionary of Provider vDCs:
AdminExtension.VcloudAdminExtension adminExt = client.GetVcloudAdminExtension();
string pvdcName = "ProvVDC1";
Dictionary<string, ReferenceType> refs = adminExt.GetVMWProviderVdcRefsByName();
…
// Get reference for pvdcName -> pvdcRef
ReferenceType pvdcRef = refs[pvdcName];
VMWProviderVdc vmwPvdc = VMWProviderVdc.GetVMWProviderVdcByReference(client, pvdcRef);
```

Using the vCloud SDK for .NET allows you to access vCloud Director from a C# development environment. These examples show how to use .NET methods. The vCloud SDK for .NET simplifies access to the vCloud API. For more information about using the SDK, see the *vCloud SDK for .NET Developer's Guide*.

The vCloud API is REST-based. For more information about the vCloud API, see the *vCloud API Programming Guide*. Example 4 shows the REST API calls that accomplish the tasks shown in Example 1, Example 2, and Example 3, after logging in.

Example 4. REST API Calls To Get Provider vDC

```
vCloud API Ref:

GET  https://<vCloud>/api/admin
GET  https://<vCloud>/api/admin/org/id
GET https://<vCloud>/api/admin/vdc/id

GET https://<vCloud>/api/admin/extension
GET https://<vCloud>/api/admin/extension/providervdc/id
```

In general, if you do not need admin views or provider views, you can use an Organization reference to get a vDC reference, and you can use the vDC reference to get a list of vApps belonging to the vDC. Example 5 shows how to list the hierarchy of Organizations, vDCs, and vApps known to vCloud Director. This example assumes you have already logged in to vCloud Director.

Example 5. List vApps in a vDC for a Given Organization

```
Dictionary<string, ReferenceType> organizationsMap = client.GetOrgRefsByName();
if (organizationsMap != null)
{
   foreach (string organizationName in organizationsMap.Keys)
   {
      ReferenceType organizationReference = organizationsMap[organizationName];
      Organization org = Organization.GetOrganizationByReference(client, organizationReference);
      string OrgID = org.Resource.id;
      Console.WriteLine("Organization Name:" + organizationName);
      Console.WriteLine("Organization Id :" + OrgID);
   }
   foreach (ReferenceType orgRef in organizationsMap.Values)
   {
      Organization org = Organization.GetOrganizationByReference(client, orgRef);

      foreach (ReferenceType vdcRef in org.GetVdcRefs())
      {
         Vdc vdc = Vdc.GetVdcByReference(client, vdcRef);
         string vdcId = vdc.Resource.id;
         Console.WriteLine("Org vDC Id:" + vdcId);
         Console.WriteLine("Org vDC Name:" + vdc.Reference.name);
         foreach (ReferenceType vAppRef in Vdc.GetVdcByReference(client, vdcRef).GetVappRefs())
         {
            Vapp vapp = Vapp.GetVappByReference(client, vAppRef);
            Console.WriteLine("vApp Id:" + vapp.Resource.id);
            Console.WriteLine("vApp Name:" + vapp.Resource.name);
            List<VM> vms = new List<VM>();
            try
            {
               vms = vapp.GetChildrenVms();
            }
            catch
            {
               // Handle exception here
            }
            foreach (VM vm in vms)
            {
               Console.WriteLine("VM Id : " + vm.Resource.id);
               Console.WriteLine("VM Name : " + vm.Resource.name);
            }
         }
      }
   }
}
```

The .NET SDK code in Example 5 translates to the API calls shown in Example 6

Example 6. REST API Calls To List vApps in a vDC for a Given Organization

```
GET  https://<vCloud>/api/admin
GET  https://<vCloud/api/admin/org/id
GET https://<vCloud>/api/admin/vdc/id

GET https://<vCloud>/api/admin/extension
GET https://<vCloud>/api/admin/extension/providervdc/id
```

You can use a provider vDC reference to enumerate its associated datastores, as shown in Example 7. This example assumes you have already logged in to vCloud Director.

Example 7. List Datastores

```
/// <summary>
/// Returns list of Provider vDCs.
/// </summary>
/// <returns>ReferenceType</returns>
public static List<ReferenceType> GetProviderVdc()
{
   List<ReferenceType> vdcRefList = new List<ReferenceType>();
   foreach (ReferenceType vdcRef1 in
      client.GetVcloudAdminExtension().GetVMWProviderVdcRefsByName().Values)
   {
      vdcRefList.Add(vdcRef1);
   }
   return vdcRefList;
}

/// <summary>
/// Returns the list of DataStores
/// </summary>
/// <returns>ReferenceType</returns>
public static List<ReferenceType> GetDataStore()
{
   extension = client.GetVcloudAdminExtension();
   List<ReferenceType> vmDatastorelist = new List<ReferenceType>();
   foreach (ReferenceType datastoreRef in extension.GetVMWDatastoreRefs())
   {
      vmDatastorelist.Add(datastoreRef);
   }
   return vmDatastorelist;
}

// Get the datastores for the list of Provider vDCs.

foreach (ReferenceType providerVdcRef in GetProviderVdc())
{
   string providerVdcId = GetId(providerVdcRef.href);
   Console.WriteLine("Provider vDC Id:" + providerVdcId);
   Console.WriteLine("Provider vDC Name:" + providerVdcRef.name);
   foreach (string morefitem in
      VMWProviderVdc.GetResourcePoolsByMoref(client, providerVdcRef).Keys)
   {
      Console.WriteLine("Moref :" + morefitem);
   }
   foreach (VMWProviderVdcResourcePoolType VcResourcePool in
      VMWProviderVdc.GetResourcePoolsByMoref(client, providerVdcRef).Values)
   {
      string VcResourcePoolId = GetId(VcResourcePool.ResourcePoolVimObjectRef.VimServerRef.href);
      Console.WriteLine("VcResourcePoolId :" + VcResourcePoolId);
   }
}
foreach (ReferenceType item in GetDataStore())
{
   string DatastoreId = GetId(item.href);
   Console.WriteLine("Data Store ID:" + DatastoreId);
   Console.WriteLine("DataStore:" + item.name);
}
```

Retrieving Catalog information

Catalogs on vCloud Director store vApp templates and ISO images as Catalog items. Backup solutions can be asked to back up the items in the Catalog for a given Organization. Catalogs can be shared or private. A user can choose to back up all items or only selected items in the given catalog. For this it is necessary to traverse the given Catalog in an Organization to access the contents and extract the various metadata associated with the vApp.

The following example shows inventory traversal to access the Catalog items in a given Organization. This example assumes you have already logged in to vCloud Director and obtained a map of Organizations, as shown in Example 1.

Example 8. List Catalogs and Catalog Items for a Given Organization

```
Console.WriteLine();
if (organizationsMap != null && organizationsMap.Count > 0)
{
   foreach (string organizationName in organizationsMap.Keys)
   {
      ReferenceType organizationReference = organizationsMap[organizationName];
      Console.WriteLine(organizationName);
      Console.WriteLine(organizationReference.href);
      Organization organization = Organization.GetOrganizationByReference(client,
               organizationReference);
      List<ReferenceType> catalogLinks = organization.GetCatalogRefs();
      if (catalogLinks != null && catalogLinks.Count > 0)
      {
         foreach (ReferenceType catalogLink in catalogLinks)
         {
            Catalog catalog = Catalog.GetCatalogByReference(client, catalogLink);
            CatalogType catalogType = catalog.Resource;
            Console.WriteLine("   " + catalogType.name);
            Console.WriteLine("   " + catalogLink.href);
            List<ReferenceType> catalogItemReferences = catalog.GetCatalogItemReferences();
            if (catalogItemReferences != null && catalogItemReferences.Count > 0)
            {
               foreach (ReferenceType catalogItemReference in catalogItemReferences)
               {
                  Console.WriteLine("   " + catalogItemReference.name);
                  Console.WriteLine("   " + catalogItemReference.href);
               }
               Console.WriteLine();
            }
            else
            {
               Console.WriteLine("No CatalogItems Found");
            }
         }
         Console.WriteLine();
      }
      else
      {
         Console.WriteLine("No Catalogs Found");
      }
   }
}
else
{
   Console.WriteLine("No Organizations");
}
```

Example 9 shows the REST API calls that accomplish some of the tasks shown in Example 8.

Example 9. REST API Calls To List Catalog Items

```
GET https://<vCloud>/api/catalog/id
GET https://<vCloud>/api/catalog/id/catalogItems
GET https://<vCloud>/api/catalogitem/id
```

Retrieving vApp Configuration

For a typical user, a vApp is the basic unit of backup specified in vCloud Director. The current generation of backup software maps vApps to their associated virtual machines in vSphere, and thus the virtual machine becomes an actual artifact. Virtual disk and virtual machine configuration files need to be stored in a backup. Along with the associated virtual machine artifacts, the user needs to back up the metadata and properties associated with every vApp to successfully restore it in vCloud Director when needed.

When a vApp is lost or deleted from vCloud Director, backup software can restore the vApp by composing a new vApp using virtual machines restored in vSphere. In such a case it becomes imperative to restore the properties and metadata associated with the vApp in vCloud Director.

The SDK includes a number of methods that you can use to get vApp configuration information. Although some of this information is included in the OVF used to upload the vApp to vCloud Director, the information might have subsequently been modified either by using the vCloud API or through the user interface.

All of these methods apply to an object of type `Vapp`.

Methods To Retrieve vApp Configuration

- `GetChildrenVms()`

  Gets a list of all child virtual machines that constitute a given vApp. Returns `List<VM>`.

- `GetStartupSection()`

  Get virtual machine startup information. Returns `StartupSectionType`.

- `GetNetworksByName()`

  Get mapping of all the network sections using their name. Returns `Dictionary<string, NetworkSection_TypeNetwork>`.

- `GetNetworkConfigSection()`

  Get network configuration details for a vApp. The information typically contains IP scope (Gateway, Netmask, DNS settings, IP range), Parent network, Fence Mode settings, and so on. Returns `NetworkConfigSectionType`.

- `GetLeaseSettingSection()`

  Get lease settings information. It includes deployment and storage lease settings for the vApp. Returns `LeaseSettingsSectionType`.

- `GetOwner()`

  Get owner information for the vApp. Returns `ReferenceType`.

- `GetMetadata()`

  Every resource in vCloud API can be associated with user-defined metadata. This method returns user-defined metadata associated with a vApp. Returns `MetadataType`.

Example 10 shows the REST API calls used to get vApp configuration data.

Example 10. REST API Calls To Get vApp Configuration

```
GET https://<vCloud>/api/vapp/id
GET https://<vCloud>/api/vapp/id/startupSection
GET https://<vCloud>/api/vapp/id/networkConnectionSection
GET https://<vCloud>/api/vapp/id/networkConfigSection
GET https://<vCloud>/api/vapp/id/leaseSettingsSection
GET https://<vCloud>/api/vapp/id/owner
GET https://<vCloud>/api/vapp/id/metadata
```

Virtual Machine Information

vCloud Director also stores virtual machine configuration information uploaded from an OVF file into a vApp template. If you have not modified a virtual machine configuration since uploading, you can use this information to verify the configuration of the virtual machine before restoring it.

The following methods, applied to an object of type `VM`, retrieve configuration data structures from vCloud Director.

Configuration Data for a Virtual Machine

- `GetVirtualHardwareSection()`

  Get hardware requirements of the virtual machine. Returns `VirtualhardwareSection_Type`.

- `GetOperatingSystemSection()`

  Get information about the guest operating system installed on this virtual machine. Returns `OperatingSystemSectionType`.

- `GetNetworkConnectionSection()`

  Get information about virtual network devices used by this virtual machine. Returns `NetworkConnectionSectionType`.

- `GetRuntimeInfoSection()`

  Get version of VMware Tools installed on the virtual machine. Returns `RuntimeInfoSectionType`.

Example 11 shows the REST API calls corresponding to the virtual machine configuration sections available from the SDK for .NET.

Example 11. REST API Calls To Get Virtual Machine Configuration Data

```
GET https://<vCloud>/api/vapp/id/virtualhardwaresection
GET https://<vCloud>/api/vapp/id/operatingSystemSection
GET https://<vCloud>/api/vapp/id/networkConnectionSection
GET https://<vCloud>/api/vapp/id/runtimeInfoSection
```

Preventing Updates to a vApp During Backup or Restore

While you are backing up or restoring a vApp, you need to prevent updates to the vApp configuration and metadata so that the vApp remains internally consistent. To prevent updates during the backup/restore process, the vCloud API allows the vApp to be placed in maintenance mode, which rejects any new updates to the configuration and metadata. The backup software must select maintenance mode for the vApp before starting backup or restore operations, and deselect maintenance mode for the vApp after the operations are completed. Example 12 shows how to select and deselect maintenance mode for a vApp.

Example 12. Protecting a vApp with Maintenance Mode

```
using com.vmware.vcloud.sdk;
using com.vmware.vcloud.api.rest.schema;
…
VApp vapp; // VApp utility class from vCloud SDK
// Identify vApp

vapp.EnableMaintenance();   // Enter maintenance mode

// Perform backup/restore here

vapp.DisableMaintenance(); // Exit maintenance mode
```

Example 13 shows the corresponding REST API calls used to select and deselect maintenance mode for a vApp.

Example 13. REST API Calls To Protect a vApp with Maintenance Mode

```
POST https://<vCloud>//api/vapp/id/action/enterMaintenancemode
POST https://<vCloud>/api/vapp/id/action/exitMaintenanceMode
```

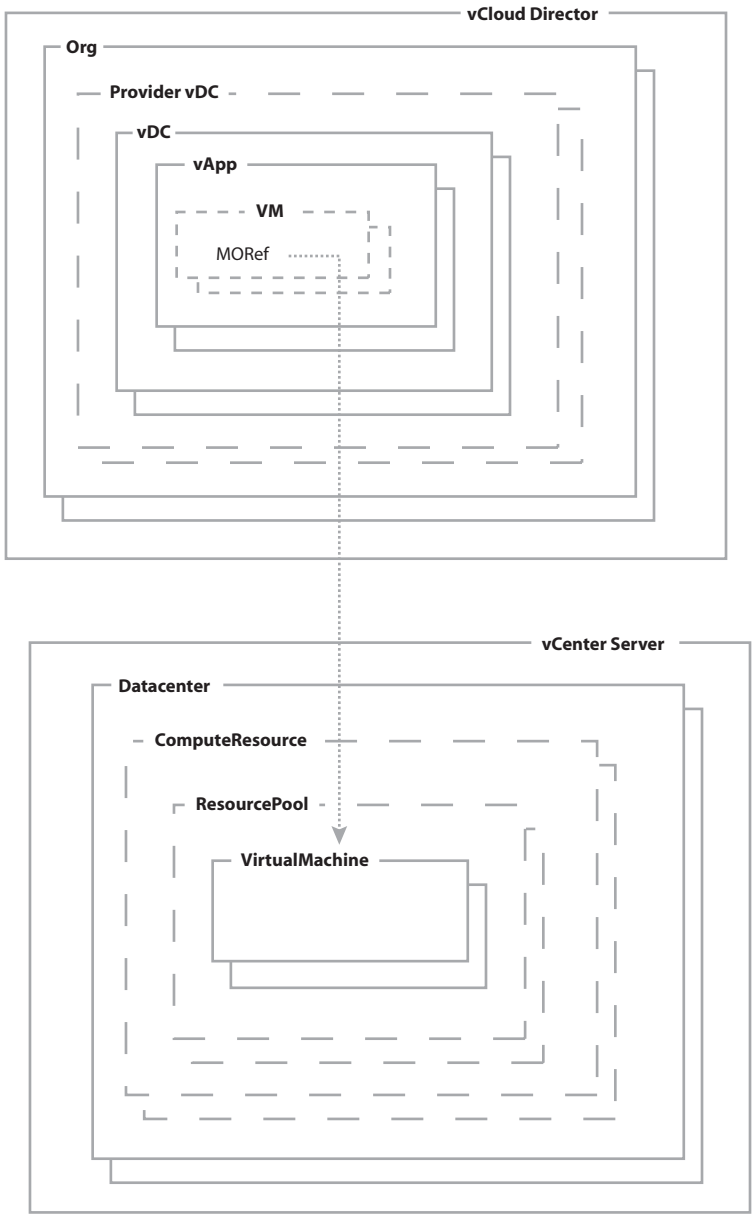
NOTE Selecting maintenance mode does not affect current or pending tasks associated with the vApp. Current or pending tasks will run to completion concurrent with the backup or restore operation. If these tasks involve configuration changes, they could result in an inconsistent vApp configuration. The backup system must ensure that such tasks are complete before storing the vApp properties and metadata.

Associating vCloud Resources with vSphere Entities

The admin view of vCloud Director resources provides additional information about the corresponding entities relevant to the vSphere platform. This information is available only when administrative credentials are used to log in to vCloud Director. The additional information does not replace the use of the vSphere API to provide comprehensive information about the entities. It merely provides the bridge between the vCloud and vSphere by mapping the IDs known to the respective systems.

For example, any given virtual machine is known in vCloud Director by a URN that contains the UUID and resource type. The same resource is identified in vSphere using its native identification, a MORef (Managed object reference). Additional information provided in the vCloud API makes the necessary link between the two entities by mapping their ID in the two systems. The mapping context is shown in Figure 2.

Figure 2. Mapping to a Virtual Machine from a vApp



The vCloud API describes the mapping in terms of XML elements, shown in Example 14. The box in Example 14 highlights XML data that maps a virtual machine from vCloud Director to vSphere. The MORef of the virtual machine is in bold type.The object type is shown as `VIRTUAL_MACHINE`.

Example 14. XML Mapping a Virtual Machine URL to a MORef

```
<Vm needsCustomization="false" deployed="false" status="3" name="RedHat6"
               id="urn:vcloud:vm:f487ba71-058a-47a9-9e9a-def458c63fd5"
               type="application/vnd.vmware.vcloud.vm+xml"
               href="https://10.20.140.167/api/vApp/vm-f487ba71-058a-47a9-9e9a-def458c63fd5">
   <VCloudExtension required="false">
      <vmext:VmVimInfo>

         <vmext:VmVimObjectRef>
            <vmext:VimServerRef type="application/vnd.vmware.admin.vmwvirtualcenter+xml"
               name="dao_w2k8_vc"
               href="https://10.20.140.167/api/admin/extension/vimServer/e7026985-19f6-4b9a
               -9d0d-588629e63347"/>
               <vmext:MoRef>vm-63</vmext:MoRef>
               <vmext:VimObjectType>VIRTUAL_MACHINE</vmext:VimObjectType>
         </vmext:VmVimObjectRef>

         <vmext:DatastoreVimObjectRef>
            <vmext:VimServerRef type="application/vnd.vmware.admin.vmwvirtualcenter+xml"
               name="dao_w2k8_vc"
               href="https://10.20.140.167/api/admin/extension/vimServer/e7026985-19f6-4b9a-9d0d-
               588629e63347"/>
            <vmext:MoRef>datastore-29</vmext:MoRef>
            <vmext:VimObjectType>DATASTORE</vmext:VimObjectType>
         </vmext:DatastoreVimObjectRef>
         <vmext:HostVimObjectRef>
            <vmext:VimServerRef type="application/vnd.vmware.admin.vmwvirtualcenter+xml"
               name="dao_w2k8_vc"
               href="https://10.20.140.167/api/admin/extension/vimServer/e7026985-19f6-4b9a-9d0d-
               588629e63347"/>
            <vmext:MoRef>host-28</vmext:MoRef>
            <vmext:VimObjectType>HOST</vmext:VimObjectType>
         </vmext:HostVimObjectRef>
         <vmext:VirtualDisksMaxChainLength>1</vmext:VirtualDisksMaxChainLength>
      </vmext:VmVimInfo>
   </VCloudExtension>
…
</Vm>
```

Besides the virtual machine object itself, the `VmVIMInfo` element encapsulated in the `VCloudExtension` element of Example 14 lists a datastore object and a host object. Each section provides the vSphere entity reference (MORef) for the corresponding entity, along with its type. The types are `DATASTORE` and `HOST`, respectively. In vCloud Director, the virtual machine can be described as virtual machine `vm-63` stored in datastore `datastore-29` and managed by vCenter Server `dao_w2k8_vc`.

In a similar way, Example 15 shows the administrative view of a vDC wherein the `VCloudExtension` element provides additional information about the corresponding entities in vSphere. In this particular case, the vDC in the example is based on a resource pool configured in vCenter Server, named `dao_w2k8_vc`. More information on this server can be obtained by using the vCloud API and its reference URL, which is available as the `href` property. The `MoRef` element provides the ID of the resource pool that backs the given vDC, as known to vSphere. Since a MORef is treated as an opaque value, the `VimObjectType` element specifies the type of object that the MORef points to. Combining these elements enables you to use the vSphere API and to locate the Resource Pool served by the specified vCenter Server.

Example 15. XML Mapping a Datacenter URL to a MORef

```
<AdminVdc … >
    <VCloudExtension required="false">
        <vmext:VimObjectRef>
            <vmext:VimServerRef type="application/vnd.vmware.admin.vmwvirtualcenter+xml"
               name="dao_w2k8_vc"
               href="https://10.20.140.167/api/admin/extension/vimServer/e7026985-19f6-4b9a-9d0d-
               588629e63347"/>
            <vmext:MoRef>resgroup-52</vmext:MoRef>
            <vmext:VimObjectType>RESOURCE_POOL</vmext:VimObjectType>
        </vmext:VimObjectRef>
    </VCloudExtension>
…
</AdminVdc … >
```

Example 16 shows how to use SDK helper methods to access the vSphere specific information for the virtual machines of a given vApp.

The return value of the methods has type `VimObjectRefType`, which provides a reference to a vCenter Server, a MORef to the vSphere entity, and the type of the entity it is referring to.

Example 16. Using the SDK for .NET To Access MORefs

```
using com.vmware.vcloud.sdk;
using com.vmware.vcloud.api.rest.schema;
…
// Log in with admin privileges and get admin view of vDC containing the vApp.
…

VApp vapp; // VApp utility class from vCloud SDK
// Identify vApp.
…

List<VM> Vms;
// Get list of children VM(s)
Vms = vapp.GetChildrenVms();
foreach (VM vm in Vms)
{
  Console.WriteLine();
  // Access vSphere information for VM
  …
  // VM Info from vSphere
  VimObjectRefType vmRef = vm.GetVMVimRef();
  Console.WriteLine("VirtualMachine: " + vmRef.moRefField);

  // Datastore Info from vSphere for VM
  VimObjectRefType datastoreRef = vm.GetVMDatastoreVimRef();
  Console.WriteLine("Datastore: " + datastoreRef.moRefField);

  // Host info form vSphere for VM
  VimObjectRefType hostRef = vm.GetVMHostVimRef();
  Console.WriteLine("Host: " + hostRef.moRefField);
}
```

Restoring vApps

During the restore process, the backup software typically restores a virtual machine in vSphere using the virtual machine configuration and disk files. In situations where the vApp has been lost from the vCloud Director inventory, the backup software needs to first restore the virtual machine in vSphere, and then import the virtual machine into vCloud Director.

Although the vApp may contain multiple virtual machines in the view of vCloud Director, the virtual machines are known individually to vSphere. To complete the restore operation, the backup software needs to re-create the restored vApp so that all the member virtual machines are created as child virtual machines of the vApp.

To re-create the vApp using the vCloud SDK for .NET, the backup software must use two method calls: `ImportVmAsVapp` and `ImportVmIntoVapp`. Use the `ImportVmAsVapp` method to create a vApp from any one of the child virtual machines. Then call the `ImportVmIntoVapp` method once for each remaining child virtual machine.

Example 17 shows how to use both methods to create a vApp using the vCloud SDK.

Example 17. Importing Virtual Machines into vApps

```
/// <summary>
/// Reference to hold the vCloud Client reference
/// </summary>
private static VcloudAdminExtension extension = null;

vcloudClient.login(user,  password);
extension = vcloudClient.getVcloudAdminExtension();

// Get references for known VIM Servers
Dictionary<string, ReferenceType> vimServerRefsByName = extension.GetVMWVimServerRefsByName();

// Select VIM Server Reference
VMWVimServer vimServer = VMWVimServer.GetVMWVimServerByReference(
                            vcloudClient, vimServerRefsByName[vimServerName]);
…
// Import first VM from VIM server as vApp:
ImportVmIntoVAppParamsType importVmIntoVAppParamsType = new ImportVmIntoVAppParamsType();
importVmIntoVAppParamsType.vmMoRefField = moref; // vSphere ID from backup data.
importVmIntoVAppParamsType.vdcField = vdcRef; // vDC where the new vApp will be created.
Vapp vapp = vimServer.ImportVmAsVApp(importVmAsVAppParamsType); // Task is embedded in vapp.
...
foreach (VM vm in vms)
{
   // Import remaining VMs from VIM Server into existing vApp:
   …
   importVmIntoVAppParamsType.vmMoRefField = moref; // vSphere ID from backup data.
   importVmIntoVAppParamsType.vAppField = vapp; // vApp to hold restored VMs.
   Task task = vimServer.ImportVmIntoVApp(importVmIntoVAppParamsType);
   …
};
…
```

Conclusion

This document provides an overview of how to use the vCloud SDK for .NET to back up and restore vApps in vCloud Director. This information is provided as a guide to using the vCloud SDK while designing backup/restore software. This document supplements a number of other documents, which you should consult for more detailed information about the operations described here. Those documents are listed in "References" on page 19.

The examples in this document are not intended to be complete. They are intended only to illustrate the method calls you would use during backup and restore operations using vCloud Director and vCenter Server. For more detail about the SDK methods and for more examples of their use, see the *vCloud SDK for .NET Developer's Guide*.

References

1 vCloud Director Administrator's Guide (https://www.vmware.com/pdf/vcd_15_admin_guide.pdf)

2 vCloud API Programming Guide (https://www.vmware.com/pdf/vcd_15_api_guide.pdf)

3 vCloud SDK for .NET Developer's Guide (https://www.vmware.com/pdf/vcd_15_sdk_dotnet_dg.pdf)

4 VMware vSphere Basics Guide (http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-basics-guide.pdf)

5 vSphere Web Services SDK Programming Guide (http://pubs.vmware.com/vsphere-50/topic/com.vmware.wssdk.pg.doc_50/PG_Preface.html)

6 vSphere API Reference (http://pubs.vmware.com/vsphere-50/index.jsp?topic=/com.vmware.wssdk.apiref.doc_50/index.html&single=true)

7 Designing Backup Solutions for VMware vSphere (https://www.vmware.com/support/developer/vddk/vcb_vsphere_backup.pdf)

8 Virtual Disk API Programming Guide (http://pubs.vmware.com/vsphere-50/index.jsp?topic=/com.vmware.vddk.pg.doc_50/vddkPreface.html)

If you have comments about this documentation, submit your feedback to: docfeedback@vmware.com

VMware, Inc. 3401 Hillview Ave., Palo Alto, CA 94304 www.vmware.com

Item: EN-000804-00