An Introduction to vSphere APIs

21 SEP 2023

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

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

Contents

Revision History

1

This book, *An Introduction to vSphere APIs and SDKs*, is updated with each release of the product or when necessary.

This table provides the update history of *An Introduction to vSphere APIs and SDKs*.

Revision	Description
21 SEP 2023	Updated and renamed, formerly *Getting Started with vSphere APIs*.
01 MAY 2023	Corrected copy-paste error in ESX Agent Manager section.
14 APR 2023	New publication with vSphere 8.0 Update 1 release.

Introduction to vSphere APIs

2

VMware vSphere® is VMware's virtualization platform, which transforms data centers into aggregated computing infrastructures that include CPU, storage, and networking resources. vSphere manages these infrastructures as a unified operating environment, and provides you with the tools to administer the data centers that participate in that environment. VMware offers a number of powerful APIs and SDKs to help you manage your virtualization infrastructure.

VMware provides APIs for managing different aspects of VMware vCenter Server® and the data center. This publication describes APIs in vSphere release 8.0 and later.

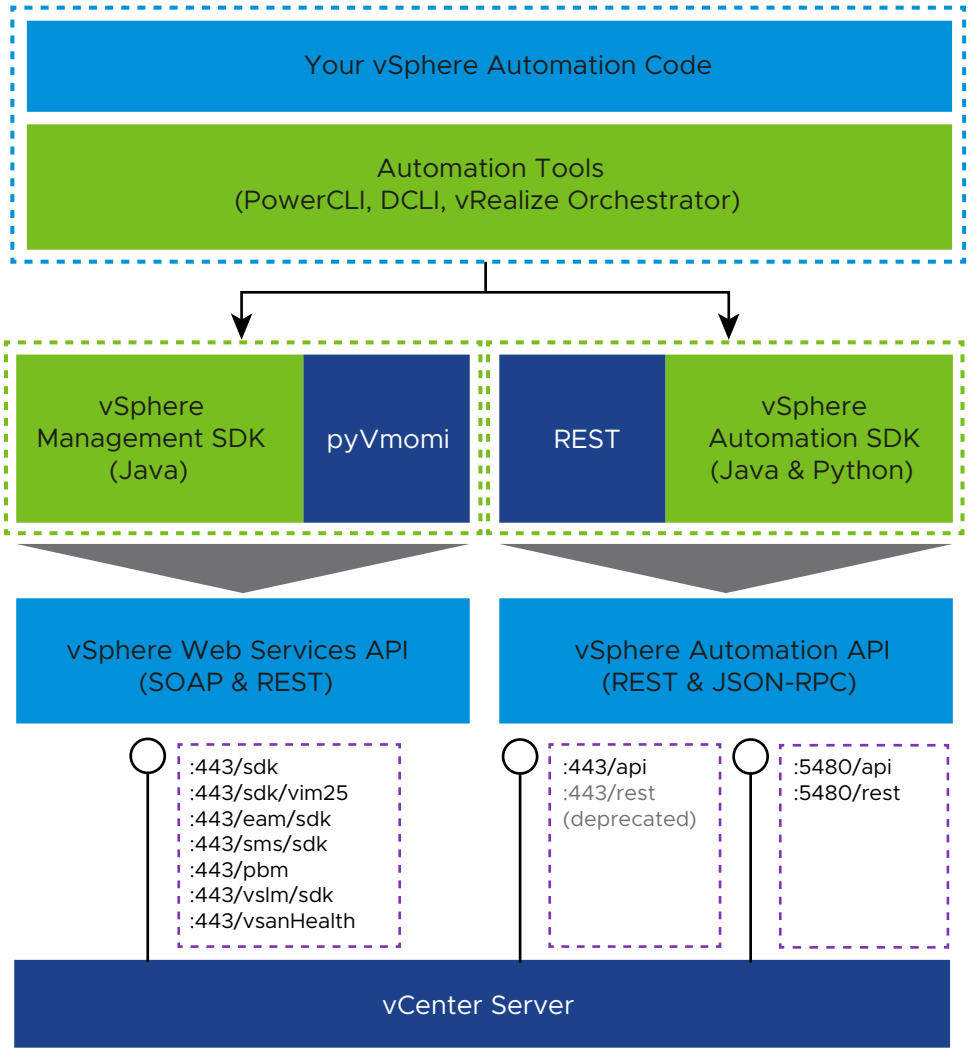
The following table provides a quick comparison of vSphere APIs with respect to functional capabilities and language bindings. Each row of the table indicates protocol and language support for an API. Language support can be a full SDK, an add-on library, or in some cases partial library support for the API.

Table 2-1. vSphere APIs Compared

API	function	SOAP	JSON	Java	Python	Golang
vSphere Automation API	datacenter management advanced features		Yes	SDK	SDK	govmomi (partial)
VIM API	datacenter management basic features	Yes	Yes	SDK	pyvmomi	govmomi
vCenter Server Management API	vCenter Server Appliance admin		Yes	SDK	SDK	govmomi
Storage Policy API	storage affinity profiles	Yes		SDK	pyvmomi	govmomi
Storage Monitoring API	manage storage arrays	Yes		SDK	pyvmomi	govmomi
vSAN API	pooling ESXi attached storage	Yes		SDK	SDK + pyvmomi	SDK
Virtual Storage Lifecycle Management (VSLM)	manage virtual disks standalone	Yes		SDK		build from WSDL
ESX Agent Manager (EAM)	vSphere extensions & services	Yes		SDK	pyvmomi	govmomi
Virtual Disk API	operate on virtual disks, C language	local	local	DDK		

For reference, the following diagram shows key service endpoints used by different vSphere APIs.

Figure 2-1. vCenter Server endpoints



Core vSphere APIs

3

VMware vSphere virtual infrastructure supports a user interface for automated management of remote vSphere assets. Many vSphere APIs address different functional areas. Two vSphere APIs support core functions of vSphere management.

The core functional areas can be viewed as configuration and management of hosts, virtual machines, and networks, as well as storage at an abstract level. The two core APIs that address these functional areas complement each other both in function and in design principles. Use the two APIs together to achieve complete functionality.

The vSphere Automation API applies REST design principles using JSON-based wire protocols, while the VIM API applies RPC principles using a SOAP protocol. In general, the Automation API offers a simpler design, while the VIM API offers better compatibility with older vSphere releases.

Read the following topics next:

- vSphere Automation API
- vSphere Web Services (VIM) API

vSphere Automation API

The VMware vSphere Automation SDK™ is optimized for developing applications that use major vSphere features such as virtual machine management, content libraries and resource deployment, tagging, and managing internal and external security certificates.

The vSphere Automation API follows a resource-based REST architecture with native JavaScript Object Notation (JSON) requests and responses. Clients can use the following technologies to send requests to the vCenter Server Automation API endpoint:

- Java SDK
- Python SDK
- REST
- PowerCLI™

Other language bindings are available from open-source projects.

The vSphere Automation API currently includes core features such as these:

- Federated authentication with support for AD FS and Okta.

- The VMware vSphere Lifecycle Manager™ feature supports desired-state management of hosts in a cluster.
- The content library feature manages templating, storage, and deployment of virtual machines and vApps.
- Tagging of vSphere objects with metadata that facilitates filtering and sorting of objects when managing large data centers.
- Certificate management handles certificates issued by the VMware Certificate Authority (VMCA) and third-party or custom-made certificates to your environment.
- VMware vSphere Trust Authority™ is a foundational technology that enhances workload security.

VMware implements all major new vSphere features in the Automation API.

The Automation API exposes two protocols: JSON-RPC, which is used by the Automation SDKs, and a simplified JSON protocol used with REST clients. The SDKs are available for download in the vSphere Automation SDKs.

The Automation API service endpoint is

```
https://{domain}/api/{service}
```

where `{service}` is the name of a specific service interface.

For example, you can reach the Session service through the following URL:

```
https://<vcenter_server_ip_address_or_fqdn>/api/session
```

For more information about using the vSphere Automation API, see the following publications:

- *VMware vSphere Automation API REST Programming Guide*
- *VMware vSphere Automation API Java Programming Guide*
- *VMware vSphere Automation API Python Programming Guide*
- *VMware PowerCLI User's Guide*
- *vSphere Automation REST API Reference 8.0*
- *vSphere Automation API Java 8.0*
- *vSphere Automation API Python 8.0*

vSphere Web Services (VIM) API

The vSphere Web Services (VIM) API gives programmatic access to core virtualized resources that support robust, fault-tolerant virtualized applications comprising compute, networking, and storage resources. Resorcure types include virtual machines, ESXi hosts, clusters, datastores, networks, and system abstractions such as events, alarms, authorization, and plug-in extensions.

The VIM API is a mature and comprehensive management API. The API works against both ESXi and vCenter Server systems, providing access to a hierarchy of managed objects that you use to configure, monitor, and manage datacenter resources.

Developers have several ways to access the VIM API:

- The vSphere Management SDK contains Java bindings and samples that use SOAP access to the server.
- The pyvmomi SDK is a user-friendly Python package containing bindings and utilities for working with the APIs.
- The govmomi SDK is a comprehensive set of bindings, utilities, examples, and a CLI for Go developers.
- The vSphere Management SDK provides language-neutral WSDL code for the SOAP protocol, from which you can generate bindings for your chocie of language.
- The vSphere Management SDK provides a language-neutral OpenAPI specification to which you can write code that uses a REST-like JSON protocol.

The vSphere Web Services API includes these core features:

- Data center inventory management, using a hierarchy of folders and resource objects
- A powerful server-side data retrieval feature, the Property Collector
- Host system configuration and management
- Datastore configuration and management
- Network configuration and management
- Virtual machine configuration and management, including guest OS operations
- Virtual machine encryption
- Cluster and resource pool configuration and management
- Management of events and alarms

The Web Services API uses a SOAP-XML RPC protocol. It is bundled in the vSphere Management SDK. The Management SDK provides RPC stubs for clients with Java language bindings.

The Web Services server endpoint is

```
https://{domain}/sdk
```

The VIM JSON protocol service endpoint is

```
http://{domain}/sdk/vim25/8.0.1.0
```

For more information about using the VIM API, see the following documentation:

- *vSphere Web Services SDK Developer Setup Guide (8.0)*
- *vSphere Web Services SDK Programming Guide 8.0*

- *vSphere Web Services API 8.0*
- http://developers.eng.vmware.com/apis/vi-json/latest/

vSphere Functional Extension APIs

4

VMware supplements the core vSphere APIs with several extensions that provide specialized capabilities for managing specific areas, such as management of different types of storage or the vCenter Server Appliance™ itself.

Read the following topics next:

- VMware vSAN
- Virtual Storage Lifecycle Management
- Storage Monitoring Service
- Storage Policy API
- ESX Agent Manager
- vCenter Server Management API
- vCenter Single Sign-On
- Virtual Disk API

VMware vSAN

VMware vSAN™ is a distributed layer of software that aggregates local or direct-attached capacity devices of a host cluster and creates a single storage pool shared across all hosts in the vSAN cluster.

vSAN runs natively as a part of the VMware ESXi™ hypervisor. It provides an object-oriented, distributed storage area network platform that can use either SSD or spinning disks in an ESXi host as physical media. vSAN uses dedicated VMkernel network interfaces for node-to-node communication.

vSAN supports features that require shared storage, such as HA, vMotion, and DRS, but vSAN eliminates the need for external shared storage and simplifies storage configuration and virtual machine provisioning activities.

VMware vSAN uses a software-defined approach that creates shared storage for virtual machines. It virtualizes the local physical storage resources of ESXi hosts and turns them into pools of storage that can be divided and assigned to virtual machines and applications according to their quality-of-service requirements.

The API can be used for solution integration. You can use the vSAN API to configure and monitor vSAN disk clusters and related services on ESXi hosts and vCenter Server instances. The vSAN API also handles virtual disk functions such as mounting, partitioning, secure erasing, and snapshots.

Functional features of vSAN include:

- Cluster configuration and query methods.
- Stretched cluster configuration and query methods.
- Performance, health, and diagnostics.
- Virtual disk formatting and space usage queries.
- Software upgrade handling.
- Data Protection Service (DPS) is an optional data backup and recovery feature for VMware vCloud Air that enables self-service, policy-based protection of business-critical data by backing up vApps and their associated virtual machines within dedicated or VMware vCloud® Air™ Virtual Private Cloud service types.
- Cloud Native Storage (CNS) is a VMware vSphere with VMware Tanzu™ feature that makes Kubernetes aware of how to provision storage on vSphere on-demand, in a fully automated, scalable fashion. CNS also provides visibility for the administrator into container volumes through the CNS UI within vCenter.

The vSAN API uses a SOAP-XML RPC protocol, patterned after the vSphere Web Services (VIM) API. It is bundled in the vSphere Management SDK.

The vSAN service endpoint is

```
https://{domain}/vsan
```

For more information about using the vSAN API, see the following publications:

- *vSAN Management SDK for Java*
- *vSAN Management SDK for Python*
- *vSAN Management SDK for .NET*
- *vSAN Management SDK for Perl*
- *vSAN Management SDK for Ruby*

Virtual Storage Lifecycle Management

Virtual Storage Lifecycle Management (VSLM) is a Web service that runs on vCenter Server as part of storage policy services (SPS). You use this API to connect to the VSLM endpoint and retrieve information about first class disks (FCD). SPS also includes storage policy based management (SPBM) and storage management service (SMS).

First class disk, also called improved virtual disk (IVD), refers to disks not associated with a virtual machine. Previously, disk operations on unattached virtual disks required you to create a dummy VM and attach the virtual disks before you could work with the disks.

For example, snapshots work on a per-VM basis, and to snapshot a single virtual disk rather than all virtual disks attached to a VM required configuration overhead. The FCD feature offers full lifecycle management of virtual storage objects, independent of any VM. VSLM APIs are able to create, delete, snapshot, backup, restore, and do other lifecycle tasks on VMDK objects without requiring VM attachment.

An FCD is identified by UUID and human readable name. The UUID is globally unique and the primary identifier. The UUID remains valid even if a snapshot is taken or its FCD relocated by XvMotion.

Use cases for FCD include App Volumes, just-in-time virtual desktops for VDI, OpenStack Cinder, Docker persistent volumes, Kubernetes nodes, and VMware Cloud Native Storage.

The VSLM API uses a SOAP-XML RPC protocol, patterned after the vSphere Web Services (VIM) API. It is bundled in the vSphere Management SDK.

The VSLM service endpoint is

```
https://{domain}/vslm
```

For more information about using the VSLM API, see the following API reference documentation:

- *Virtual Storage Lifecycle Management API 8.0*

Storage Monitoring Service

The Storage Monitoring Service (SMS) provides methods to retrieve information about available storage topology, capabilities, and state.

The vSphere API for Storage Awareness (VASA) permits storage arrays to integrate with vCenter for management functionality. VASA providers expose features of the physical storage devices, such as storage health status, configuration info, and storage capacity.

SMS establishes and maintains connections with VASA providers. SMS retrieves information about storage availability from the providers, and clients can use the SMS API to perform the following operations.

- Identify VASA providers.
- Retrieve information about storage arrays.
- Identify vSphere inventory entities (hosts and datastores) which are associated with external storage entities on the storage arrays.

The SMS API uses a SOAP-XML RPC protocol, similar to the vSphere Web Services (VIM) API. It is bundled in the vSphere Management SDK.

The SMS service endpoint is

```
https://{domain}/sms
```

For more information about using the Storage Monitoring Service, see the following API reference documentation:

- *vCenter Storage Monitoring Service API 8.0*

Storage Policy API

The Storage Policy (SPBM) API, available in the vSphere Management SDK, is an extension of the Web Services (VIM) API. Storage policies simplify the task of matching available storage to virtual machines.

A vSphere storage profile defines storage policy information that describes storage requirements for virtual machines and storage capabilities of storage providers. You use the VMware Storage Policy API to manage the association between virtual machines and datastores.

With appropriate policies, SPBM governs virtual machine placement into appropriate datastore, accounting for RAID levels, pre-zeroed allocated disk, and space efficient thin provisioned disk. Storage policies may also request specific data services for virtual disks, such as caching, replication, and deduplication.

The Storage Policy API supports the following capabilities:

- Access to vSphere storage profiles and to entities associated with profiles
- Compliance checking of profiles and entities
- Placement compatibility checking

The Storage Policy API uses a SOAP-XML RPC protocol, similar to the vSphere Web Services (VIM) API. It is bundled in the vSphere Management SDK. The Management SDK provides RPC stubs for clients with Java language bindings.

The Storage Policy service endpoint is

```
https://{domain}/pbm
```

For more information about using the Storage Policy API, see the following API reference documentation:

- *VMware Storage Policy API 8.0*

ESX Agent Manager

You can add functions to vSphere by developing software applications that you register as vCenter Server extensions. A vSphere Solution is an extension that registers with vCenter Server and implements some or all of the extension features of the vSphere API.

You can deploy a solution as an Open Virtualization Format (OVF) package, with optional VMware vSphere Installation Bundles (VIB). You can also install solutions by using an installer, such as Windows Installer (MSI) or RPM Package Manager. Most of the extension functions in the vSphere API are independent of the technology that you use to deploy a solution. If you deploy a solution by using OVF, you can use the vCenter Extension vService to simplify the registration of the solution with vCenter Server.

You can create two types of vSphere solutions:

- A vService is a service that a solution provides to specific applications that run inside virtual machines and vApps. A solution can provide several types of vServices. Virtual machines or vApps can have dependencies on several types of vServices.
- A vSphere ESX agent is a virtual machine and an optional vSphere Installation Bundle (VIB) that extends the functions of an ESXi host to provide additional services that a vSphere solution requires.

The vService Manager and ESX Agent Manager are pre-built solutions that are part of the vCenter Server Extensions functionality within the vCenter Server.

vSphere ESX Agent Manager is an intermediary between vCenter Server and a solution with two key aspects:

- provisioning agent virtual machines and VIB modules for the solution onto a scope in vCenter Server
- monitoring changes to the agent virtual machines and scope in vCenter Server, and reporting them back to the solution

vSphere ESX Agent Manager exposes a SOAP-based API similar to the VIM API, that solutions use to register and monitor agents. The SDK includes a Model-View-Controller framework and library functions for developing vSphere extensions that interface with the vSphere Web Services (VIM) API. It is bundled in the vSphere Management SDK.

The EAM service endpoint is

```
https://{domain}/eam/sdk
```

For more information about using the ESX Agent Manager, see the following publication:

- *Developing and Deploying vSphere Solutions, vServices, and ESX Agents*

vCenter Server Management API

You can use the vCenter Server Management API to automate management tasks for a vCenter Server Appliance. Rather than manage the data center infrastructure, this API manages the server instance itself.

Features of the vCenter Server Management API include:

- Installation and Setup

- Configuration
- Patching and Upgrade
- Backup and Restore

The vCenter Server Management API uses a REST JSON RPC protocol, associated with the Automation API. It is available for download in the vSphere Automation SDKs.

For most operations, the vCenter Server Management API uses the service endpoint

```
https://{domain}/api/{service}
```

where `{service}` is the name of a specific service interface.

During setup, the Management API uses the service endpoint

```
https://{domain}:5480/api/{service}
```

until networking is configured and the service is available on port 443.

For more information about using the vCenter Server Management API, see the following publications:

- *VMware vSphere Automation API REST Programming Guide*
- *VMware vSphere Automation API Java Programming Guide*
- *VMware vSphere Automation API Python Programming Guide*
- *vSphere Automation REST API Reference 8.0*
- *vSphere Automation API Java 8.0*
- *vSphere Automation API Python 8.0*

vCenter Single Sign-On

The VMware vCenter Single Sign-On API simplifies secure communications with the vCenter Single Sign-On service.

The vCenter Single Sign-On server provides a Security Token Service (STS). A token uses the Security Assertion Markup Language (SAML), which is an XML encoding of authentication data.

vCenter Single Sign-On authentication can use the following identity store technologies:

- Windows Active Directory
- OpenLDAP (Lightweight Directory Access Protocol)
- Local user accounts (vCenter Single Sign-On server resident on the vCenter Server machine)
- vCenter Single Sign-On user accounts

This API defines a set of request operations that correspond to the WS-Trust 1.4 bindings:

- Issue – Obtains a token from a vCenter Single Sign-On server.

- Renew – Renews an existing token.
- Validate – Validates an existing token.
- Challenge – Part of a negotiation with a vCenter Single Sign-On server to obtain a token.

The vCenter Single Sign-On SDK includes Java bindings for the vCenter Single Sign-On WSDL. The SDK also contains sample code that demonstrates client-side support for the WSSecurityPolicy standard. Security policies specify the elements that provide SOAP message security. To secure SOAP messages, a client inserts digital signatures, certificates, and SAML tokens into the SOAP headers for vCenter Single Sign-On requests. The Java sample includes a JAX-WS implementation of SOAP header methods that support the vCenter Single Sign-On security policies.

The vCenter Single Sign-On SDK is bundled in the vSphere Management SDK.

The Single Sign-On service endpoint is

```
https://{domain}/STS/STSService
```

For more information about using the vCenter Single Sign-On API, see the following documentation:

- *vCenter Single Sign-On Programming Guide*

Virtual Disk API

The Virtual Disk API provides methods for backup and recovery, cloning, and other operations on virtual disks.

The Virtual Disk API can connect to local or network virtual disks, either for ESXi or hosted applications.

The Virtual Disk API supports the following functions:

- Create, open, close, and delete virtual disks
- Read and write both data and metadata
- Grow, shrink, and defragment virtual disks
- Manage virtual disk snapshot chains
- Quiesce virtual machine access and flush buffers
- Clone virtual disks

The Virtuas Disk API is provided as a C language library, running in-process as part of user-defined utility software.

The Virtual Disk API is available as part of the Virtual Disk Development Kit (VDDK).

For more information about the Virtual Disk API, see the *Virtual Disk Development Kit Programming Guide*.

vSphere Add-On SDKs

5

Add-on API layers are available for some popular vSphere APIs. Open-source communities can contribute new language bindings or usability improvements.

Read the following topics next:

- pyVmomi
- govmomi

pyVmomi

pyVmomi is an open-source Python binding for several vSphere SOAP-based APIs.

The pyVmomi package provides a convenience layer on top of the core VIM API and the following related extension APIs that use the SOAP/XML protocol:

- ESX Agent Manager
- Storage Policy API
- Storage Monitoring Service (SMS)

pyVmomi is notable for:

- Simplified approach to connection, authentication, and retrieval of the ServiceContent object
- Module to handle client-server synchronization and asynchronous task tracking
- Convenient retrieval of managed object properties
- Facility to interface with vCenter Single Sign-On

For more information about using the pyVmomi, see the following documentation:

- http://vmware.github.io/pyvmomi-community-samples/#getting-started

govmomi

govmomi is a Go language library for interacting with the VMware Web Services API for ESXi and vCenter Server.

The code in the govmomi package is a wrapper for the code that is generated from the vSphere API description. It primarily provides convenience functions for working with the vSphere API:

- Generated types, managed objects, and methods for the complete VIM API
- Client structure that abstracts a session and its immutable state
- Session Manager abstraction that allows a user to login and logout
- Wrappers for a selection of managed objects
- Generated coverage for some other SOAP-based APIs, such as SMS, SPBM, and vSAN
- Partial coverage for the vSphere Automation API

For more information about using govmomi, see the following documentation:

- https://pkg.go.dev/github.com/vmware/govmomi

Use Cases for vSphere APIs

6

Read the following topics next:

- Virtual Disk Development Kit

Virtual Disk Development Kit

This SDK is most frequently used to implement backup and restore applications. This section is a walk-though of code to restore a virtual machine.

The Virtual Disk Development Kit (VDDK) is a set of libraries and utilities that enable software developers to create applications that can access and manipulate virtual disks used for storage by products such as vSphere, Workstation, and Fusion. VDDK is a valuable tool for developers of storage and backup applications.

VDDK provides APIs for backup and recovery, cloning, and other operations on virtual disks. With VDDK, developers can create applications that perform backup and restore operations for virtual machines, as well as applications that create clones or snapshots of virtual disks. VDDK also includes utilities for converting virtual disks between different formats and for creating virtual disks from scratch.

Safekeeping – a VDDK Use Case

Based on VDDK, Safekeeping is a "proof of concept" application to demonstrate backup and restore of vSphere VMs, vApps, and first-class disk. You can find its source code on Github, as linked from the developer.vmware.com SDK landing page for VDDK. This section focuses on `RestoreVm.java`, a source code module that recovers a VM previously saved by the `BackupVm.java` module.

```
safekeeping backup vm:myTestVM
safekeeping restore vm:myTestVM
```

After package and interface imports, class `RestoreVm` extends method `AbstractRestoreFcoWithDisk`, which in turn extends method `AbstractRestoreFco` for restoring a first class object (FCO) but with disk, as in a vSphere VM. (A vApp is an FCO that includes VMs but not disk.) `RestoreVm` must be passed a VIM connection, user-specified command options, and a log file.

The `checkOptions` method parses passed-in options, including (if specified) VM name, parent vApp, data center, resource pool, datastore, and storage profile. If the destination VM folder cannot be restored to, `RestoreVM` uses a different location.

Snapshot is a typical operation for both backup and restore. The `createSnapshot` method takes a vSphere snapshot so the VM can continue running while being restored, and also to quiesce the virtual disk while it is being replaced. If a snapshot is unnecessary because the VM does not exist, snapshot is skipped.

If the VM is being restored because it no longer runs, the `destroy` method deletes the old VM so it does not conflict with the VM to be restored.

The `disksMapping` method is one of the most complicated in `RestoreVM`. It needs to know the number of disks to restore, the storage profile of disks, and the location of disk data in the backup. This information comes from the `managedInfo` object, combined with the generated `profile` object.

Once virtual disks are correctly mapped, they are restored by the `disksRestore` method. This could take a lot of time, so the `final` loop blocks until disks are restored.

If a failed VM was not deleted, the `manageExistingFco` method deals with naming conflicts between the pre-existing VM and the VM to be restored. The VM might be part of a vApp, which allows duplicate names. If the pre-existing VM is powered on, the backed-up VM cannot be restored. With duplicate names, the user can be given a chance to rename VM to be restored.

The `networksMapping` method determines what networks the backed-up VM used for each virtual NIC, and attempts to reformulate them using existing networks.

Finally the `powerOn` method starts up the restored VM so it can be configured. A VM can be powered on before the Guest OS is ready to run.

The `reconfig` method configures the VM based on profiles of its virtual disks and networks determined above. Configuration may be interrupted by a fault. Similar `reconfigDisk` and `reconfigNetwork` methods take care of disk backing and network backing for the VM to be restored. The `removeSnapshot` methods cleans up the snapshot previously created.

Finally the `restore` method, the main restore function, puts this all together. It retrieves the restore profile and calls snapshot, start-access, `disksRestore`, and end-access in an if statement. This is where the C++ functions in VixDiskLib get called (start and end access prevent vMotion during restore).

```
if (createSnapshot(rar) && startVddkAccess(rar, jvddk) && disksRestore(rar, jvddk)
                        && endVddkAccess(rar, jvddk)) {
   final String st = rar.getFcoToString() + " Restore Success - start cleaning";
}
```

The restored VM could either be newly created with the `restoreByCreate` method, or recovered with import of virtual disk data by the `restoreByImport` method. It is the `restoreMetadata` method that decides which. The `restoreManagedInfo` method takes care of items managed by vCenter Server, such as folder groupings and resource pools.