VMware vCenter Server Appliance Management Programming Guide for Python

Modified on 05 JUN 2018

vCenter Server 6.5

VMware ESXi 6.5

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

Contents

About the vCenter Server Appliance Management Programming Guide for Python

The *vCenter Server Appliance Management Programming Guide for Python* provides information about using APIs to work with the vCenter Server Appliance, a turnkey solution for managing data centers featuring VMware®vCenter Server and VMware ESXi.

Intended Audience

This information is intended for anyone who wants to develop software to configure, monitor, and manage the vCenter Server Appliance. The information is written for developers who have some experience with Python.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Updated Information

This *VMware vCenter Server Appliance Management Programming Guide for Python* is updated with each release of the product or when necessary.

This table provides the update history of the *VMware vCenter Server Appliance Management Programming Guide for Python*.

Revision	Description
05 JUN 2018	Updated some code snippets.
30 NOV 2017	Refined multiple topics.
15 NOV 2016	Initial release.

Introduction to the vCenter Server Appliance and vCenter Server Appliance APIs

1

The vCenter Server Appliance provides a fully packaged solution for data center management in a vSphere environment. You can use the APIs to configure, monitor, and maintain the vCenter Server Appliance.

This chapter includes the following topics:

About vSphere

vSphere is the VMware software stack that implements private-cloud data center management and the on-premises component of hybrid-cloud deployments.

A vSphere installation includes one or more instances of vCenter Server configured to manage one or more virtual data centers. Each virtual data center includes one or more instances of VMware ESXi.

The vCenter Server Appliance contains an instance of vCenter Server. The appliance Management API gives you programmatic access to manage the management elements of your data center.

About ESXi

Each instance of ESXi includes management agents and the VMware hypervisor layer, which runs several virtual machines. Each virtual machine contains a guest operating system, such as Windows or Linux, capable of running IT or user applications.

The vCenter Server Appliance runs as a virtual machine on an ESXi host. The appliance provides an independent endpoint capable of handling API requests both for vCenter Server and for the appliance Management API.

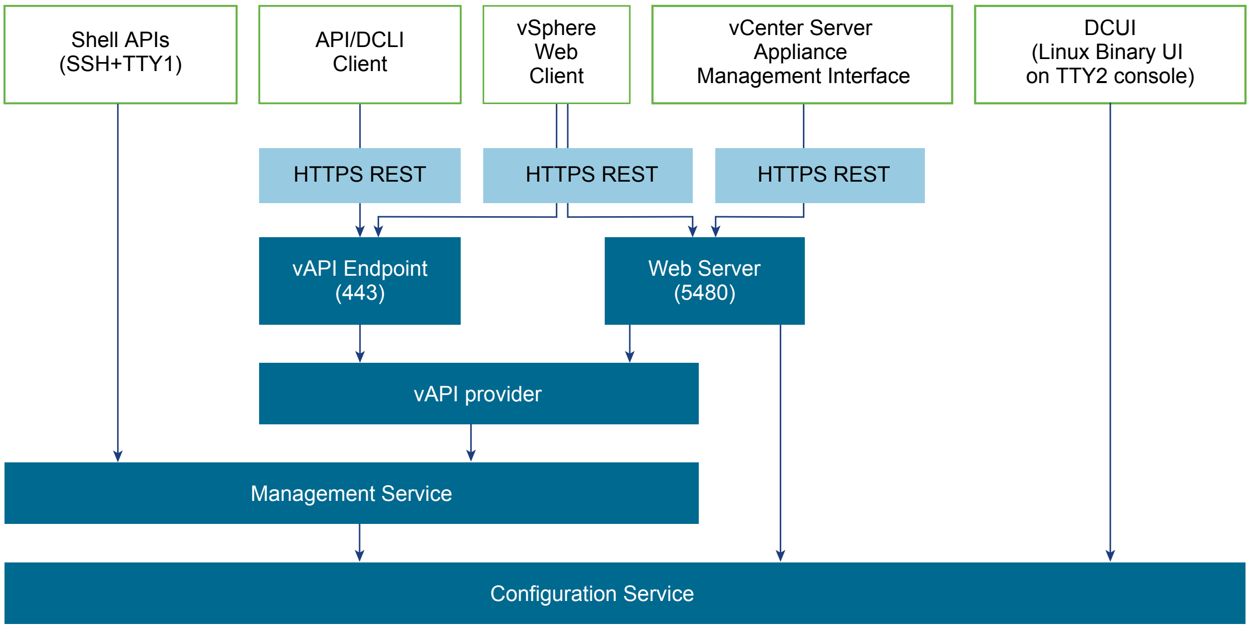
vCenter Server Appliance Management Overview

The vCenter Server Appliance is an instance of vCenter Server running in a Photon OS™ guest operating system.

vCenter Server is a collection of services designed for managing and monitoring vSphere installations. The vCenter Server Appliance responds to CLI commands, requests from the vSphere Web Client, and API requests from custom clients. API clients can be written in a choice of several software languages.

The vCenter Server Appliance is managed by CLI, Web interfaces, or API requests. These requests help you manage appliance configuration, monitor resource usage, or back up and restore the vCenter Server instance. You can also use API requests to check the health of the vCenter Server installed in the appliance. This programming guide explains how to use the APIs that are available to manage the appliance.

Figure 1-1. vCenter Server Appliance Management Connections



For more information about the capabilities of the vCenter Server Appliance, see *vCenter Server Appliance Configuration*.

Limitations of Programming for the vCenter Server Appliance

The vCenter Server Appliance supports several programming interfaces for monitoring appliance health and performance, managing network configuration, security hardening, and other functionalities. The vCenter Server Appliance also supports several user interfaces, which offer overlapping sets of functionality.

You can use vSphere Web Client to perform common operations. By using the API, you have access to more specific settings and operations.

However, the API cannot access all the capabilities. A few special features require direct shell access or special user interfaces. See Supplementing the vCenter Server Appliance API.

API Endpoints for Managing the vCenter Server Appliance

The vCenter Server Appliance integrates with the vSphere Automation API endpoint that provides a common surface for exposing several vSphere services. When you use the vSphere Automation API endpoint, you establish a single session that provides access to virtual machine management, search and filter, Content Library, and other services for working with vSphere objects.

Other endpoints associated with the vCenter Server Appliance include the Lookup Service and the vCenter Single Sign-On Service. For more information about using the Lookup Service, see Chapter 3 Retrieving Service Endpoints. For more information about using the vCenter Single Sign-On Service, see vCenter Single Sign-On Token Authentication for the vCenter Server Appliance.

Supplementing the vCenter Server Appliance API

Some less common features of the vCenter Server Appliance are not accessible by API. These features require direct shell access or specific user interfaces.

Direct Console User Interface to the vCenter Server Appliance

The Direct Console User Interface provides access to basic operations for appliance management and set up.

The DCUI provides access to a subset of management functions. It provides direct access to the appliance if the vSphere Web Client and the vCenter Server Appliance Management Interface become unavailable.

For an illustration showing appliance connections, see the block diagram Figure 1-1.

After the vCenter Server Appliance startup is complete, the DCUI displays basic CPU, memory, and network information on the operator console. The root user can use the DCUI screen to configure network interfaces, DNS, and super administrator password.

vCenter Server Appliance Management Interface

The vCenter Server Appliance Management Interface is an interface for configuring, monitoring, and patching of the vCenter Server Appliance.

The vCenter Server Appliance Management Interface runs in a browser that connects to port 5480 of the appliance. The vCenter Server Appliance Management Interface provides access to all the service APIs of the appliance.

For an illustration showing Appliance connections, see the block diagram Figure 1-1.

Appliance Shell and the vCenter Server Appliance

You can use the appliance shell to access all of the vCenter Server Appliance commands and plug-ins that you use for monitoring, troubleshooting, and configuring the appliance through the API.

For an illustration showing Appliance connections, see the block diagram Figure 1-1.

For more information about the appliance shell, see *vCenter Server Appliance Configuration*.

vSphere Web Client and the vCenter Server Appliance

The vSphere Web Client is a user interface for general management tasks.

For an illustration showing Appliance connections, see the block diagram Figure 1-1.

DCLI and the vCenter Server Appliance

The Data Center CLI (DCLI) is a CLI client of the VMware vSphere® Automation™ SDK. Almost all methods that are available in the vSphere Automation SDKs are available as DCLI commands.

You can run the DCLI from the VMware vSphere® Command-Line Interface (vCLI) package or from the appliance shell.

For an illustration showing appliance connections, see the block diagram Figure 1-1.

For more information about the DCLI, see *Getting Started with vSphere Command-Line Interfaces*.

Quick Start with vCenter Server Appliance APIs

You can start using the vCenter Server Appliance APIs without accessing the Lookup Service Endpoint or the vCenter Single Sign-On Endpoint. In a production environment, you might instead use centralized service registration and token authentication.

To use the vCenter Server Appliance APIs without the Lookup Service or token authentication, see vCenter Single Sign-On User Name and Password Authentication for the vCenter Server Appliance.

vCenter Server Appliance Programming Environment 2

The vCenter Server Appliance is a key component in your vSphere environment, providing several services for data center management, as well as its own management.

This chapter includes the following topics:

Platform Services Controller Services

With Platform Services Controller, all VMware products within the same environment can share the authentication domain and other services. Services include certificate management, authentication, and licensing.

Platform Services Controller includes the following core infrastructure services.

Table 2-1. Platform Services Controller Services

Service	Description
`applmgmt` (VMware Appliance Management Service)	Handles appliance configuration and provides public API endpoints for appliance lifecycle management. Included on the Platform Services Controller appliance.
`vmware-cis-license` (VMware License Service)	Each Platform Services Controller includes VMware License Service, which delivers centralized license management and reporting functionality to VMware products in your environment. The license service inventory replicates across all Platform Services Controller in the domain at 30-second intervals.
`vmware-cm` (VMware Component Manager)	Component manager provides service registration and lookup functionalities.
`vmware-psc-client` (VMware Platform Services Controller Client)	Backend to the Platform Services Controller Web interface.

Table 2‑1. Platform Services Controller Services (Continued)

Service	Description

`vmware-sts-idmd`
(VMware Identity Management Service)
`vmware-stsd`
(VMware Security Token Service)
	Services behind the vCenter Single Sign-On feature, which provide secure authentication services to VMware software components and users.
	By using vCenter Single Sign-On, the VMware components communicate using a secure SAML token exchange mechanism. vCenter Single Sign-On constructs an internal security domain (vsphere.local by default) where the VMware software components are registered during installation or upgrade.

`vmware-rhttpproxy` (VMware HTTP Reverse Proxy)	The reverse proxy runs on each Platform Services Controller node and each vCenter Server system. It is a single entry point into the node and enables services that run on the node to communicate securely.
`vmware-sca` (VMware Service Control Agent)	Manages service configurations. You can use the `service-control` CLI to manage individual service configurations.
`vmware-statsmonitor` (VMware Appliance Monitoring Service)	Monitor the vCenter Server Appliance guest operating system resource consumption.
`vmware-vapi-endpoint` (VMware vAPI Endpoint)	The vSphere Automation API endpoint provides a single point of access to vAPI services. You can change the properties of the vAPI Endpoint service from the vSphere Web Client. See the *vSphere Automation SDKs Programming Guide* for details on vAPI endpoints.
`vmafdd` VMware Authentication Framework	Service that provides a client-side framework for vmdir authentication and serves the VMware Endpoint Certificate Store (VECS).
`vmcad` VMware Certificate Service	Provisions each VMware software component that has the vmafd client libraries and each ESXi host with a signed certificate that has VMCA as the root certificate authority. You can change the default certificates by using the Certificate Manager utility or Platform Services Controller Web interface. VMware Certificate Service uses the VMware Endpoint Certificate Store (VECS) to serve as a local repository for certificates on every Platform Services Controller instance. Although you can decide not to use VMCA and instead can use custom certificates, you must add the certificates to VECS.
`vmdird` VMware Directory Service	Provides a multitenant, multimastered LDAP directory service that stores authentication, certificate, lookup, and license information. Do not update data in `vmdird` by using an LDAP browser. If your domain contains more than one Platform Services Controller instance, an update of vmdir content in one vmdir instance is propagated to all other instances of vmdir.
`vmdnsd` VMware Domain Name Service	Not used in vSphere 6.x.

Table 2-1. Platform Services Controller Services (Continued)

Service	Description

`vmonapi`
VMware Lifecycle Manager API
`vmware-vmon`
VMware Service Lifecycle Manager
	Start and stop vCenter Server services and monitor service API health. The `vmware-vmon` service is a centralized platform-independent service that manages the lifecycle of Platform Services Controller and vCenter Server. Exposes APIs and CLIs to third-party applications.

`lwsmd` Likewise Service Manager	Likewise facilitates joining the host to an Active Directory domain and subsequent user authentication.

Platform Services in the vCenter Server Appliance Environment

The vCenter Server Appliance registers its services with the Platform Services Controller, but in some situations you might be unable to use the Lookup Service registration. In those situations, you must access an appliance endpoint directly.

A direct connection to the vCenter Server Appliance can become necessary in configurations where the Platform Services Controller is embedded in the appliance. When the appliance vCenter Server is being restored from a backup, or when the embedded Lookup Service is restarting, you might be unable to look up the appliance's service registration.

For more information about embedded or external Platform Services Controller configurations, see vSphere Deployment Configurations.

For more information about backup and restore operations, see Chapter 8 Maintenance of the vCenter Server Appliance.

vSphere Deployment Configurations

vSphere Automation client applications communicate with services on the Platform Services Controller and vCenter Server components of the virtual environment. vCenter Server can be deployed with an embedded or external Platform Services Controller.

vCenter Server with an Embedded Platform Services Controller

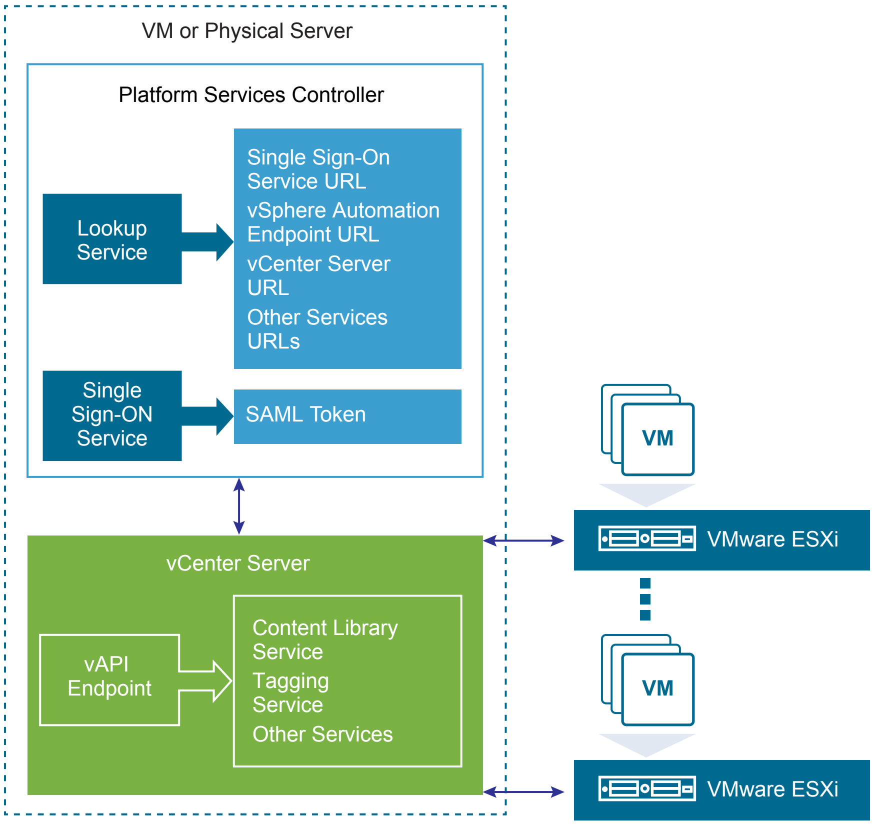
vCenter Server and Platform Services Controller reside on the same virtual machine or physical server. This deployment is most suitable for small environments such as development or test beds.

You can use the Platform Services Controller in two ways to establish secure, authenticated sessions for your client application, by making requests to the Lookup Service and the vCenter Single Sign-On Service.

One way to use the Platform Services Controller is to request an authentication token that can be used to authenticate requests across services. The client connects to the Lookup Service and retrieves the vCenter Single Sign-On Service endpoint and the vSphere Automation API endpoint. The client then uses the vCenter Single Sign-On endpoint to authenticate with user credentials and receive a token that securely verifies the client's credentials. This allows the client to authenticate with a number of service endpoints without sending user credentials over the network repeatedly.

Alternatively, if the client connects directly to the vSphere Automation API endpoint, there is no need for the client to interact with the vCenter Single Sign-On Service. The client sends user credentials to the vSphere Automation API endpoint, which creates a session identifier that persists across requests.

Figure 2-1. vCenter Server with Embedded Platform Services Controller

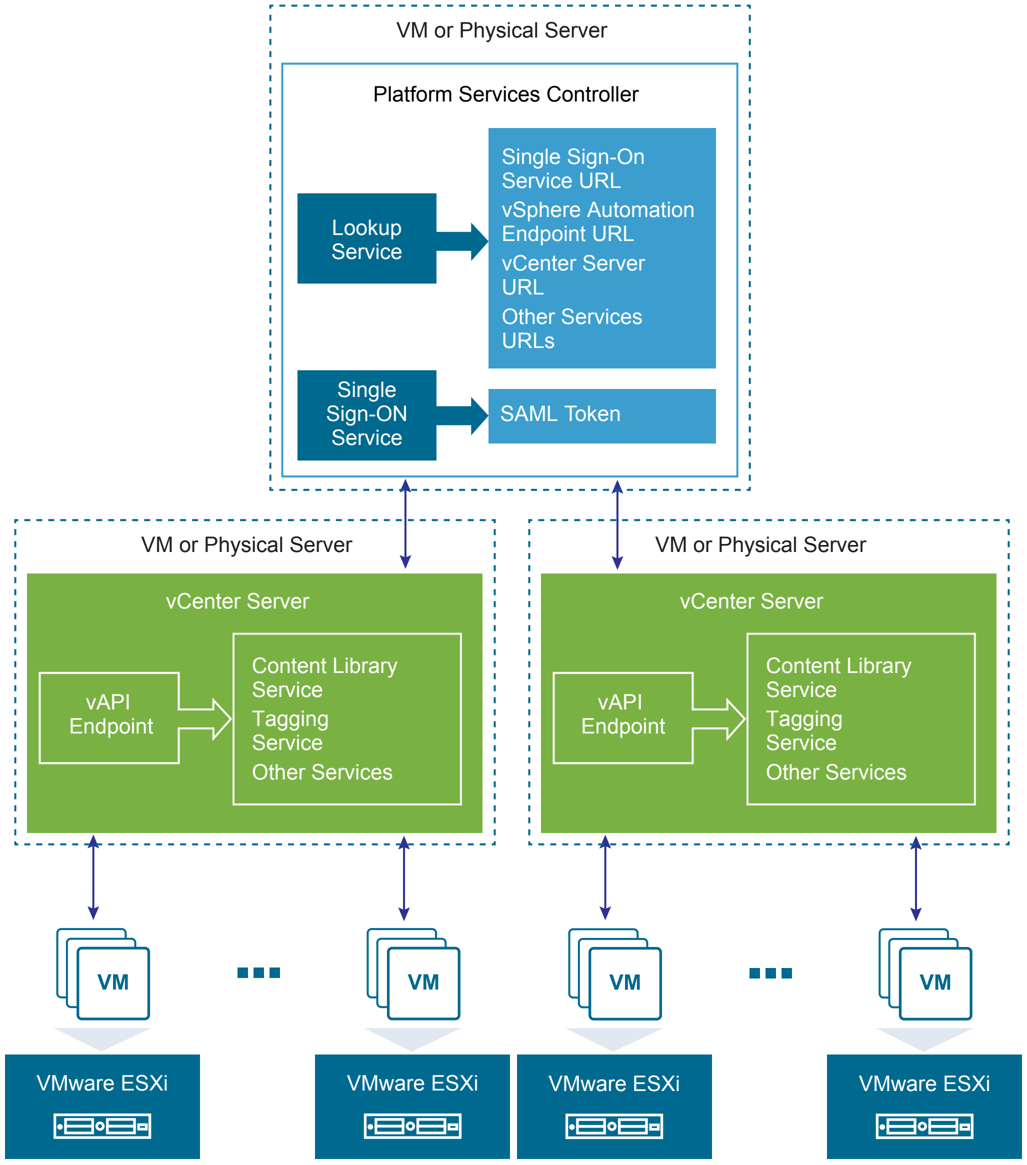


vCenter Server with an External Platform Services Controller

In the case of an external Platform Services Controller, the vCenter Server and the Platform Services Controller are deployed on separate virtual machines or physical servers. The Platform Services Controller can be shared across several vCenter Server instances. For larger deployments or to provide better availability, more than one Platform Services Controller can be deployed. When configured as replication partners within a single vCenter Single Sign-On domain, Platform Services Controller instances replicate all user and system data within the cluster.

A client application functions in a similar way as in a Platform Services Controller with embedded vCenter Server deployment. The only difference is that the client application can access services on multiple vCenter Server instances, or services only on a particular vCenter Server instance.

Figure 2-2. vCenter Server with External Platform Services Controller



Retrieving Service Endpoints 3

To access services and resources in the virtual environment, vSphere Automation API client applications must know the endpoints of vSphere Automation and vSphere services. Client applications retrieve service endpoints from the Lookup Service that runs on the Platform Services Controller.

The Lookup Service provides service registration and discovery by using a Web services API. By using the Lookup Service, you can retrieve endpoints of services on the Platform Services Controller and vCenter Server. The following endpoints are available from the Lookup Service.

- The vCenter Single Sign-On endpoint on the Platform Services Controller. You use the vCenter Single Sign-On service to get a SAML token and establish an authenticated session with a vSphere Automation endpoint or a vCenter Server endpoint.
- The vSphere Automation Endpoint on vCenter Server. Through the vSphere Automation Endpoint, you can make requests to vSphere Automation API services such as virtual machine management, Content Library, and Tagging.
- The vCenter Server endpoint. In an environment with external Platform Services Controller instances, you can use the vCenter Server endpoint to get the node ID of a particular vCenter Server instance. By using the node ID , you can retrieve service endpoints on that vCenter Server instance.
- The vSphere API endpoint and endpoints of other vSphere services that run on vCenter Server.

Workflow for Retrieving Service Endpoints

The workflow that you use to retrieve service endpoints from the Lookup Service might vary depending on the endpoints that you need and their number. Follow this general workflow for retrieving service endpoints.

1. Connect to the Lookup Service on the Platform Services Controller and service registration object so that you can query for registered services.
2. Create a service registration filter for the endpoints that you want to retrieve.
3. Use the filter to retrieve registration information for services from the Lookup Service.
4. Extract one or more endpoint URLs from the array of registration information that you receive from the Lookup Service.

This chapter includes the following topics:

Filtering for Predefined Service Endpoints

The Lookup Service maintains a registration list of vSphere services. You can use the Lookup Service to retrieve registration information for any service by setting a registration filter that you pass to the `List()` function on the Lookup Service. The functions and objects that you can use with the Lookup Service are defined in the `lookup.wsdl` file that is part of the SDK.

Lookup Service Registration Filters

You can query for service endpoints through a service registration object that you obtain from the Lookup Service. You invoke the `List()` function on the Lookup Service to list the endpoints that you need by passing `LookupServiceRegistrationFilter`. `LookupServiceRegistrationFilter` identifies the service and the endpoint type that you can retrieve.

Optionally, you can include the node ID parameter in the filter to identify the vCenter Server instance where the endpoint is hosted. When the node ID is omitted, the `List()` function returns the set of endpoint URLs for all instances of the service that are hosted on different vCenter Server instances in the environment.

For example, a `LookupServiceRegistrationFilter` for querying the vSphere Automation service has these service endpoint elements.

Table 3-1. Service Registration Filter Parameters

Filter Types	Value	Description
`LookupServiceRegistrationServiceType`	`product= "com.vmware.cis"`	vSphere Automation namespace.
	`type="cs.vapi"`	Identifies the vSphere Automation service.

Table 3-1. Service Registration Filter Parameters (Continued)

Filter Types	Value	Description
`LookupServiceRegistrationEndpointType`	`type="com.vmware.vapi.endpoint"`	Specifies the endpoint path for the service.
	`protocol= "vapi.json.https.public"`	Identifies the protocol that will be used for communication with the endpoint .

For information about the filter parameter of the available predefined service endpoints, see Filter Parameters for Predefined Service Endpoints.

Filter Parameters for Predefined Service Endpoints

Depending on the service endpoint that you want to retrieve, you provide different parameters to the `LookupServiceRegistrationFilter` that you pass to the `List()` function on the Lookup Service. To search for services on a particular vCenter Server instance, set the node ID parameter to the filter.

Table 3-2. Input Data for URL Retrieval for the Lookup Service Registration Filter

Service	Input Data	Value
vCenter Single Sign-On	product namespace	`com.vmware.cis`
	service type	`cs.identity`
	protocol	`wsTrust`
	endpoint type	`com.vmware.cis.cs.identity.sso`
vSphere Automation Endpoint	product namespace	`com.vmware.cis`
	service type	`cs.vapi`
	protocol	`vapi.json.https.public`
	endpoint type	`com.vmware.vapi.endpoint`
vCenter Server	product namespace	`com.vmware.cis`
	service type	`vcenterserver`
	protocol	`vmomi`
	endpoint type	`com.vmware.vim`
vCenter Storage Monitoring Service	product namespace	`com.vmware.vim.sms`
	service type	`sms`
	protocol	`https`
	endpoint type	`com.vmware.vim.sms`
vCenter Storage Policy-Based Management	product namespace	`com.vmware.vim.sms`
	service type	`sms`
	protocol	`https`
	endpoint type	`com.vmware.vim.pbm`
vSphere ESX Agent Manager	product namespace	`com.vmware.vim.sms`

Table 3-2. Input Data for URL Retrieval for the Lookup Service Registration Filter (Continued)

Service	Input Data	Value
	service type	`cs.eam`
	protocol	`vmomi`
	endpoint type	`com.vmware.cis.cs.eam.sdk`

Connect to the Lookup Service and Retrieve the Service Registration Object

You must connect to the Lookup Service to gain access to its operations. After you connect to the Lookup Service, you must retrieve the service registration object to make registration queries.

Procedure

1 Connect to the Lookup Service.

   a Configure a connection stub for the Lookup Service endpoint, which uses SOAP bindings, by using the HTTPS protocol.

   b Create a connection object to communicate with the Lookup Service.

2 Retrieve the Service Registration Object.

   a Create a managed object reference to the Service Instance.

   b Invoke the `RetrieveServiceContent()` method to retrieve the `ServiceContent` data object.

   c Save the managed object reference to the service registration object.

   With the service registration object, you can make registration queries.

Python Example of Connecting to the Lookup Service and Retrieving a Service Registration Object

The example is based on the code from the `lookup_service_helper.py` sample file.

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
...

# 1 - Create SOAP client object to communicate with the Lookup Service.
my_ls_stub = Client(url=wsdl_url, location=ls_url)

# 2 - Configure service & port type for client transaction.
my_ls_stub.set_options(service='LsService', port='LsPort')

# 3 - Manufacture a managed object reference.
managed_object_ref = \
   my_ls_stub.factory.create('ns0:ManagedObjectReference')
managed_object_ref._type = 'LookupServiceInstance'
managed_object_ref.value = 'ServiceInstance'

# 4 - Retrieve the ServiceContent object.
ls_service_content = \
my_ls_stub.service.RetrieveServiceContent(managed_object_ref)

# 5 - Retrieve the service registration object.
service_registration = ls_service_content.serviceRegistration
```

Retrieve Service Endpoints on vCenter Server Instances

You can create a function that obtains the endpoint URLs of a service on all vCenter Server instances in the environment. You can modify that function to obtain the endpoint URL of a service on a particular vCenter Server instance.

Prerequisites

- Establish a connection with the Lookup Service.
- Retrieve a service registration object.

Procedure

1 Create a registration filter object, which contains the following parts:

   - A filter criterion for service information
   - A filter criterion for endpoint information

Option	Description
Omit the node ID parameter	Retrieves the endpoint URLs of the service on all vCenter Server instances.
Include the node ID parameter	Retrieves the endpoint URL of the service on a particular vCenter Server instance.

2 Retrieve the specified service information by using the `List()` function.

Depending on whether you included the node ID parameter, the `List()` function returns one of the following results:

- A list of endpoint URLs for a service that is hosted on all vCenter Server instances in the environment.
- An endpoint URL of a service that runs on a particular vCenter Server instance.

What to do next

Call the function that you implemented to retrieve service endpoints. You can pass different filter parameters depending on the service endpoints that you need. For more information, see Filter Parameters for Predefined Service Endpoints.

To retrieve a service endpoint on a particular vCenter Server instance, you must retrieve the node ID of that instance and pass it to the function. For information about how to retrieve the ID of a vCenter Server instance, see Retrieve a vCenter Server ID by Using the Lookup Service.

Python Example of Retrieving Service Endpoints on vCenter Server Instances

This example provides a common pattern for filtering Lookup Service registration data. This example is based on the code in the `lookup_service_helper.py` sample file.

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
def lookup_service_infos(prod, svc_type, proto, ep_type, node_id='*') :

   # 1 - Create a filter criterion for service info.
   filter_service_type = \
     my_ls_stub.factory.create('ns0:LookupServiceRegistrationServiceType')
   filter_service_type.product = prod
   filter_service_type.type = svc_type

   # 2 - Create a filter criterion for endpoint info.
   filter_endpoint_type = \
     my_ls_stub.factory.create('ns0:LookupServiceRegistrationEndpointType')
   filter_endpoint_type.protocol = proto
   filter_endpoint_type.type = ep_type

   # 3 - Create the registration filter object.
   filter_criteria = \
     my_ls_stub.factory.create('ns0:LookupServiceRegistrationFilter')
   filter_criteria.serviceType = filter_service_type
   filter_criteria.endpointType = filter_endpoint_type
   if (node_id != '*') :
     filter_criteria.nodeId = node_id

   # 4 - Retrieve specified service info with the List() method.
   service_infos = my_ls_stub.service.List(service_registration,
                                           filter_criteria)
   return service_infos
```

Retrieve a vCenter Server ID by Using the Lookup Service

You use the node ID of a vCenter Server instance to retrieve the endpoint URL of a service on that vCenter Server instance. You specify the node ID in the service registration filter that you pass to the `List()` function on the Lookup Service.

Managed services are registered with the instance name of the vCenter Server instance where they run. The instance name maps to a unique vCenter Server ID. The instance name of a vCenter Server system is specified during installation and might be an FQDN or an IP address.

Prerequisites

- Establish a connection with the Lookup Service.
- Retrieve a service registration object.

Procedure

1 List the vCenter Server instances.

2 Find the matching node name of the vCenter Server instance and save the ID.

Use the node ID of the vCenter Server instance to filter subsequent endpoint requests. You can use the node ID in a function that retrieves the endpoint URL of a service on a vCenter Server instance. For information about implementing such a function, see Retrieve Service Endpoints on vCenter Server Instances.

Python Example of Retrieving a vCenter Server ID by Using the Lookup Service

This example provides a common pattern for filtering Lookup Service registration data. This example is based on the code in the `lookup_service_helper.py` sample file.

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
def get_mgmt_node_id(node_instance_name) :

   # 1 - List the vCenter Server instances.
   mgmt_node_infos = lookup_service_infos(prod='com.vmware.cis',
                                          svc_type='vcenterserver',
                                          proto='vmomi', ep_type='com.vmware.vim',
                                          node_id='*')

   # 2 - Find the matching node name and save the ID.
   for node in mgmt_node_infos :
     for attribute in node.serviceAttributes :
       if attribute.key == 'com.vmware.vim.vcenter.instanceName' :
         if attribute.value == node_instance_name :
           return node.nodeId
```

Retrieve a vSphere Automation Endpoint on a vCenter Server Instance

Through the vSphere Automation Endpoint, you can access other vSphere Automation services that run on vCenter Server, such as Content Library and Tagging. To use a vSphere Automation service, you must retrieve the vSphere Automation Endpoint.

Prerequisites

- Establish a connection with the Lookup Service.
- Retrieve a service registration object.
- Determine the node ID of the vCenter Server instance where the vSphere Automation service runs.
- Implement a function that retrieves the endpoint URL of a service on a vCenter Server instance.

Procedure

1. Invoke the function for retrieving the endpoint URL of a service on a vCenter Server instance by passing filter strings that are specific to the vSphere Automation endpoint.
2. Save the URL from the resulting single-element list.

Python Example of Retrieving a vSphere Automation Endpoint on a vCenter Server Instance

This example provides a common pattern for filtering Lookup Service registration data. This example is based on the code in the `lookup_service_helper.py` sample file.

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
service_infos = lookup_service_infos(prod='com.vmware.cis',
                                             svc_type='cs.vapi',
                                             proto='vapi.json.https.public',
                                             ep_type='com.vmware.vapi.endpoint',
                                             node_id=my_mgmt_node_id)
my_vapi_url = service_infos[0].serviceEndpoints[0].url
```

Authentication Mechanisms 4

The vCenter Server Appliance accepts several authentication methods. The authentication method that you choose depends on whether you choose token authentication, and on the state of the appliance.

During normal operation, the vCenter Server Appliance enables you to authenticate with vCenter Single Sign-On credentials. You have the option to use either token authentication or user name and password authentication. The user name and password must be recognized within the vCenter Single Sign-On domain.

However, during the process of restoring the Appliance from a backup image, you must use a different authentication protocol. For more information, see Restoring the vCenter Server Appliance.

This chapter includes the following topics:

- vCenter Single Sign-On User Name and Password Authentication for the vCenter Server Appliance
- vCenter Single Sign-On Token Authentication for the vCenter Server Appliance

vCenter Single Sign-On User Name and Password Authentication for the vCenter Server Appliance

You can authenticate with the vCenter Server Appliance by using a user name and password known to the vCenter Single Sign-On Service.

If you prefer to delegate the process of requesting a SAML token for your API client, you can present your vCenter Single Sign-On domain credentials to the vSphere Automation API endpoint and request a session ID. The endpoint process forwards your credentials to the vCenter Single Sign-On Service and requests a SAML token on your behalf. In this case, you never deal with the token.

Authenticate with vCenter Single Sign-On Credentials and Create a Session

To establish a session with the vSphere Automation API Endpoint in the vCenter Server Appliance, you create a connection to the endpoint and authenticate with vCenter Single Sign-On credentials to receive a session ID.

Prerequisites

To perform this task, you must have the following items in place:

- The DNS name or IP address of the vCenter Server Appliance
- A vCenter Single Sign-On domain account that has the requisite permissions for the operation that you intend to invoke

Procedure

1. Create a connection context by specifying the vSphere Automation API Endpoint URL and the message protocol to be used for the connection.

2. Create the request options or stub configuration and set the specific security context to be used.

   The security context contains the vCenter Single Sign-On user name and password that are used for authenticating to the vSphere Automation API Endpoint.

3. Create an interface stub or a REST path that uses the stub configuration.

   The interface stub corresponds to the interface containing the method to be invoked.

4. Invoke the session `create` method.

   The service creates an authenticated session and returns a session identification cookie to the client.

5. Add the cookie to your request headers or to a security context for your client stub configuration.

6. Remove the basic authentication from your request headers or the security context of your client stub configuration.

Subsequent method calls authenticate with the session cookie instead of the user name and password.

What to do next

Use the updated stub configuration with the session ID to create a stub for the interface that you want to use. Method calls on the new stub use the session ID to authenticate.

Python Example of Creating a vSphere Automation API Session with SSO Credentials

This example is based on code in the `vapiconnect.py` sample file.

This example uses the following global variables.

- *my_vapi_hostname*
- *my_sso_username*
- *my_sso_password*

- *my_stub_config*

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
import requests
from com.vmware.cis_client import Session
from vmware.vapi.lib.connect import get_requests_connector
from vmware.vapi.security.session import create_session_security_context
from vmware.vapi.security.user_password import create_user_password_security_context
from vmware.vapi.stdlib.client.factories import StubConfigurationFactory

# Create a session object in the client.
session = requests.Session()

# For development environment only, suppress server certificate checking.
session.verify = False

# Create a connection for the session.
vapi_url = 'https://' + my_vapi_hostname + '/api'
connector = get_requests_connector(session=session, url=vapi_url)

# Add username/password security context to the connector.
basic_context = create_user_password_security_context(my_sso_username,
                                                      my_sso_password)
connector.set_security_context(basic_context)

# Create a stub configuration by using the username-password security context.
my_stub_config = StubConfigurationFactory.new_std_configuration(connector)

# Create a Session stub with username-password security context.
session_stub = Session(my_stub_config)

# Use the create operation to create an authenticated session.
session_id = session_stub.create()

# Create a session ID security context.
session_id_context = create_session_security_context(session_id)

# Update the stub configuration with the session ID security context.
my_stub_config.connector.set_security_context(session_id_context)
```

vCenter Single Sign-On Token Authentication for the vCenter Server Appliance

You can authenticate with the vCenter Server Appliance by using a SAML token from the vCenter Single Sign-On Service. The token can be either a bearer token or a holder-of-key token.

To use SAML token authentication, you issue a request to the vCenter Single Sign-On Service, specifying the token type (bearer or holder-of-key), expected token lifetime, renewability, and other parameters. You also supply a user name and password combination that is valid in the vCenter Single Sign-On domain. These credentials must have an associated role with sufficient privilege for the operations that you intend to invoke with the Management API.

If the vCenter Single Sign-On Service accepts your credentials, it responds with an XML message. The message contains a SAML assertion that your client can extract and present as an `Authorization` header in an HTTP request to the vSphere Automation API endpoint.

Retrieve a SAML Token

The vCenter Single Sign-On service provides authentication mechanisms for securing the operations that your client application performs in the virtual environment. Client applications use SAML security tokens for authentication.

Client applications use the vCenter Single Sign-On service to retrieve SAML tokens. For more information about how to acquire a SAML security token, see the *vCenter Single Sign-On Programming Guide* documentation.

The vSphere Automation SDK for Python provides a utility class to simplify the task of requesting a SAML token from the vCenter Single Sign-On service. The utility provides a wrapper around the complexity of handling token requests. For more information about the utility, see the `sso.py` sample file. The source file is in the vSphere Automation SDK for Python directory: `client/samples/src/com/vmware/vcloud/suite/sample/common/sso.py`.

Prerequisites

Verify that you have the vCenter Single Sign-On URL. You can use the Lookup Service on the Platform Services Controller to obtain the endpoint URL. For information about retrieving service endpoints, see Chapter 3 Retrieving Service Endpoints.

Procedure

1 Create a connection object to communicate with the vCenter Single Sign-On service.

  Pass the vCenter Single Sign-On endpoint URL, which you can get from the Lookup Service.

2 Issue a security token request by sending valid user credentials to the vCenter Single Sign-On service on the Platform Services Controller.

The vCenter Single Sign-On service returns a SAML token.

What to do next

You can present the SAML token to the vSphere Automation API Endpoint or other endpoints, such as the vSphere Web Services Endpoint. The endpoint returns a session ID and establishes a persistent session with that endpoint. Each endpoint that you connect to uses your SAML token to create its own session.

Python Example of Retrieving a SAML Token

This example is based on the code in the `external_psc_sso_workflow.py` sample file.

This example uses the following global variables.

- *my_vapi_hostname*
- *my_sso_username*
- *my_sso_password*

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
from vsphere.samples.common import sso

# Use the SsoAuthenticator utility class to retrieve
# a bearer SAML token from the vCenter Single Sign-On service.
sso_url = 'https://' + my_vapi_hostname + ':7444/ims/STSService'
authenticator = sso.SsoAuthenticator(sso_url)
saml_token = authenticator.get_bearer_saml_assertion(my_sso_username,
                                                     my_sso_password,
                                                     delegatable=True)
```

Create a vSphere Automation Session with a SAML Token

To establish a vSphere Automation session, you create a connection to the vSphere Automation API Endpoint and then you authenticate with a SAML token to create a session for the connection.

Prerequisites

- Retrieve the vSphere Automation Endpoint URL from the Lookup Service.
- Obtain a SAML token from the vCenter Single Sign-On service.

Procedure

1. Create a connection by specifying the vSphere Automation API Endpoint URL and the message protocol to be used for the connection.

   Note To transmit your requests securely, use **`https`** for the vSphere Automation API Endpoint URL.

2. Create the request options or stub configuration and set the security context to be used.

   The security context object contains the SAML token retrieved from the vCenter Single Sign-On service. Optionally, the security context might contain the private key of the client application.

3. Create an interface stub or a REST path that uses the stub configuration instance.

   The interface stub corresponds to the interface containing the method to be invoked.

4 Invoke the session `create` method.

   The service creates an authenticated session and returns a session identification cookie to the client.

5 Create a security context instance and add the session ID to it.

6 Update the stub configuration instance with the session security context.

What to do next

Use the updated stub configuration with the session ID to create a stub for the interface that you want to use. Method calls on the new stub use the session ID to authenticate.

Python Example of Creating a vSphere Automation API Session with a SAML Token

This example is based on code in the `external_psc_sso_workflow.py` sample file.

This example uses the following global variables.

- *my_vapi_hostname*
- *my_stub_config*
- *saml_token*

The example assumes that you previously obtained a vSphere Automation API URL from the Lookup Service, and a SAML token from the vCenter Single Sign-On Service.

Note For a complete and up-to-date version of the sample code, see the vSphere Automation SDK Python samples at GitHub.

```
...

# Create a session object in the client.
session = requests.Session()

# For development environment only, suppress server certificate checking.
session.verify = False
from requests.packages.urllib3 import disable_warnings
from requests.packages.urllib3.exceptions import InsecureRequestWarning
disable_warnings(InsecureRequestWarning)

# Create a connection for the session.
vapi_url = 'https://' + my_vapi_hostname + '/api'
connector = get_requests_connector(session=session, url=vapi_url)

# Add SAML token security context to the connector.
saml_token_context = create_saml_bearer_security_context(saml_token)
connector.set_security_context(saml_token_context)

# Create a stub configuration by using the SAML token security context.
my_stub_config = StubConfigurationFactory.new_std_configuration(connector)

# Create a Session stub with SAML token security context.
```

```
session_stub = Session(my_stub_config)

# Use the create operation to create an authenticated session.
session_id = session_stub.create()

# Create a session ID security context.
session_id_context = create_session_security_context(session_id)

# Update the stub configuration with the session ID security context.
my_stub_config.connector.set_security_context(session_id_context)
```

Authorization Model for Administration of the vCenter Server Appliance

5

There are three types of authorization levels in the vCenter Server Appliance.

Table 5-1. Authorization Levels

Authorization Level	Description
operator	A user has read access to appliance configuration settings.
administrator	A user has read and write access to the appliance configuration settings, but cannot manage user accounts.
super administrator	A user has all the capabilities of the other roles, and has the additional capabilities of creating local user accounts and accessing the local Bash shell.

This model applies to the API and all other interfaces to the vCenter Server Appliance except when you use SSH and log in using a local account.

This chapter includes the following topics:

Authorization Model Mapping to the vCenter Single Sign-On Domain

The three-level authorization model of the vCenter Server Appliance maps to local roles and to vCenter Single Sign-On groups, depending on how the user authenticated. This model allows consistent security control regardless of operational context.

The authorization levels map to group and role.

Table 5-2. Authorization Mapping

Authorization Level	vCenter Single Sign-On Group	Appliance Local Role
operator	SystemConfiguration.Administrators	operator
administrator	SystemConfiguration.Administrators	admin
superAdministrator	SystemConfiguration.BashShellAdministrators	superAdmin

When a super administrator adds user accounts, the options available include a choice of the role to assign to the new user.

Using the Appliance Operator Role

The **operator** role is the most restricted of the authorization levels available to users who work with the vCenter Server Appliance.

Appliance operators are allowed to view information about the appliance. They are not allowed to alter its configuration. The **operator** role is suited for monitoring and reporting functions. For example, the **operator** role allows access to these methods:

- resources.system.health.get
- resources.storage.stats.list
- services.status.get

Using the Appliance Admin Role

The **administrator** role provides an intermediate authorization level for users who manage the vCenter Server Appliance.

An **administrator** role is required for users who alter the appliance configuration, exercise control functions, or other operations that can affect appliance users.

For example, use the **administrator** role for these methods:

- networking.ip4v.renew
- networking.firewall.addr.inbound.add
- services.control
- shutdown.reboot

Using the Appliance SuperAdmin Role

The **superAdmin** role is the most expansive authorization level for users who manage the vCenter Server Appliance.

The **superAdmin** role allows unrestricted access to the appliance. This role is required for adding or altering user accounts and for using the Bash shell.

Overview of vCenter Server Appliance Interfaces

6

The vCenter Server Appliance supports multiple interface areas that you can use to accomplish various management tasks. For details about all the interfaces, including all method calls and parameters, see the API reference document in the SDK.

This chapter includes the following topics:

appliance access Interface

Use this interface to secure access to the vCenter Server Appliance.

The appliance access interface enables you to selectively enable or disable the following paths for administrative access to the appliance.

- consolecli
- Direct Console User Interface
- shell
- ssh

These access paths can be used by system administrators to access many system management functions. Most of these functions are available by using the API. There are a few exceptions for special features, such as debugging.

appliance health Interface

Use this interface to check the health of vCenter Server Appliance services.

The appliance health interface provides several queries that you can use to monitor the condition of different appliance components. The interface reports summary status of the following services.

- Overall health indicator
- Appliance services
- Database storage
- Guest operating system storage
- Operational workload
- Memory usage
- Software update status

To use these reports, issue periodic queries to spot trouble areas. You can use the information to guide corrective actions before the condition compromises the function of the appliance.

For instance, if the appliance reports that its database health is yellow, consider increasing the virtual storage available to the appliance. Or, if the software packages health condition is red, initiate prompt installation of security updates.

To minimize the amount of request traffic needed to check health, query the overall health indicator. If the overall health becomes orange or red, query the individual subsystem health indicators to identify the problem areas that may need corrective actions.

For more information about using the appliance monitoring interface, see Chapter 7 Monitoring the vCenter Server Appliance.

appliance monitoring Interface

Use this interface to track performance of the vCenter Server Appliance.

The appliance monitoring interface provides visibility into individual performance indicators, such as the following:

- Network usage statistics
- Database size
- Memory usage
- CPU usage

To use the appliance monitoring interface, first list the IDs of the available performance indicators. From the list, choose an ID and request detailed information about the indicator. Finally, request the performance data.

For more information about using the appliance monitoring interface, see Chapter 7 Monitoring the vCenter Server Appliance.

appliance networking Interface

Use this interface to examine network endpoints of the vCenter Server Appliance.

The appliance networking interface provides read-only access to the virtual network endpoints attached to the appliance. To use this interface, you can list all the available network endpoint names and status, or you can request information for a specific network endpoint.

appliance networking dns Interface

Use this interface to configure domain name service for the vCenter Server Appliance.

The appliance networking dns interface enables you to control the DNS behavior of the appliance. Using the methods of this interface, you can list, set, or test the following configuration settings.

- Fully qualified domain name of the appliance
- DNS servers used by the appliance
- DNS search domains for the appliance

The DNS configuration applies to all virtual network interfaces of the vCenter Server Appliance.

appliance recovery Interface

Use this interface to back up and restore the vCenter Server Appliance.

The appliance recovery interface simplifies backing up the vCenter Server database and restoring the vCenter Server Appliance to a working state. Backup and restore operations support several different protocols. The interface enables you to estimate in advance the storage needed for file-based backup and restore.

For more information about backup and restore operations, see Chapter 8 Maintenance of the vCenter Server Appliance.

appliance system version Interface

Use this interface to retrieve metadata about the vCenter Server Appliance.

The appliance system version interface retrieves a list of information about the appliance, such as the name, version, deployment type, and update level. You can use this interface to identify the exact software build and patch level.

appliance system time Interface

Use this interface to get the date and time in the guest operating system.

The appliance system time interface of the vCenter Server Appliance retrieves a list of information about the time in the guest operating system. This includes separate data for the time of day and the date, in specified formats, as well as the time zone and the epoch seconds.

appliance system uptime Interface

Use this interface to get the uptime of the guest operating system.

The appliance system uptime interface has a single method which retrieves the length of time that the vCenter Server Appliance has been continuously running. The value is returned in units of seconds.

appliance system storage Interface

Use this interface to manage virtual storage partitions of the vCenter Server Appliance.

The appliance system storage interface provides a method to list the storage partitions used by the vCenter Server Appliance, and a method to resize storage partitions.

The `resize` method is not a general-purpose way to set arbitrary partition sizes. Its unique purpose is to resize all partitions to their maximum size. This is particularly useful for situations where the vCenter Server Appliance database is approaching capacity and you need to expand one or more virtual disks. You can resize virtual disk backing files using the `ExtendVirtualDisk_Task` method of the VMware Web Services API, then use this interface to resize the partitions available to the guest operating system.

Monitoring the vCenter Server Appliance 7

The vCenter Server Appliance provides interfaces to check the current health of its subsystems, and to report the appliance's history of resource consumption. You can use these interfaces to spot potential trouble areas or predict future shortages.

This chapter includes the following topics:

Health Monitoring of the vCenter Server Appliance

The vCenter Server Appliance API offers health status indicators for several key components of the appliance. These indicators can be polled periodically to monitor the components for problems.

The health status indicators report graded values from green to red. The general meanings of the grades are as follows.

green	The component is healthy.
yellow	The component is healthy, but may have some problems.
orange	The component is degraded, and may have serious problems.
red	The component is unavailable, or will stop functioning soon.
gray	No health data is available.

Check Overall System Health of the vCenter Server Appliance

The vCenter Server Appliance provides a composite health indicator that enables you to test a single value that represents the health of all the appliance components. This procedure shows how to test the composite health indicator.

The value of the overall system health indicator reflects the strongest trouble indication among the appliance components. If any component has a `red` indicator, the overall system health indicator is `red`, else if any component has an `orange` indicator, the overall system health indicator is `orange`, and so on.

A `gray` value for any component indicates that data for the subsystem is unknown. If one or more components have a `gray` value, but all other subsystems are `green`, then the overall system health indicator is `gray` rather than `green`. However, if any component has a definite trouble indication, the overall system health indicator reflects the strongest trouble indication among the components.

Prerequisites

Verify that you have an active authenticated session with the vCenter Server Appliance. This procedure assumes that the session ID is present in the security context of a stub configuration.

Procedure

1 Create an interface stub or REST path that uses the stub configuration.

2 Invoke the `health.system` method.

3 Format and display the resulting value.

Python Example of Checking the Overall System Health of the vCenter Server Appliance

This example shows the use of Python with the vSphere Automation SDK for Python to request the overall system health indicator for the vCenter Server Appliance and the overall health indicator for management services. The example assumes a previously existing session with the vSphere Automation API endpoint.

This example depends on the following global variables.

- *my_stub_config*

```
from com.vmware.appliance import health_client

# This example assumes you have previously created a session
# and stored the session ID in my_stub_config.

# Issue request for overall system health.
System_stub = health_client.System( my_stub_config )
health = System_stub.get()
print( 'Overall system health: %s' % health )

# Issue request for applmgmt services health.
Applmgmt_stub = health_client.Applmgmt( my_stub_config )
health = Applmgmt_stub.get()
print( 'Applmgmt services health: %s' % health )
```

Capacity Monitoring of the vCenter Server Appliance

The vCenter Server Appliance keeps a history of statistics that you can use to monitor resources used by the vCenter Server instance.

You can use the statistics to spot peak usage demands or to monitor trends for advance warning of potential resource exhaustion.

Frequency and Retention of Statistics Collection in the vCenter Server Appliance

The vCenter Server Appliance collects statistics from the guest operating system at regular intervals and stores them in a database. Users can query the statistics in the database by selecting a time period and a roll-up function that the appliance applies to the statistics before returning them to the client.

After the appliance monitoring service starts up, it begins requesting statistics from the guest operating system periodically, at a frequency that depends on the type of statistic. The service requests storage statistics once every 10 minutes, while it requests memory, CPU, and networking statistics once per minute. The collection times are fixed relative to the startup time of the monitoring service, rather than to clock time.

The monitoring service retains statistics approximately 13 months, by default. Older statistics are deleted by the service, creating a 13-month moving window within which you can query statistics. You can choose to delete statistics as needed to conserve storage resources.

Nature of Statistics in the vCenter Server Appliance

The vCenter Server Appliance supplies statistics of several types.

The guest operating system computes statistics either as rates, such as CPU cycles per second, or as snapshots of size, such as KB used for storage. Statistics stored as size snapshots are collected at the end of their sample periods. Statistics stored as rates are computed as averages of values sampled frequently during each sample period.

When you query the statistics database, the units are not returned with the data, but you can determine the units for any metric by requesting metadata for the metric with the `get()` method.

Requesting Statistics from the vCenter Server Appliance

To request statistics, you must construct an appropriate request structure to filter statistics from the database.

To request data or metadata for a metric, you must supply the ID of the metric. You can get a list of metric IDs by using the `list()` method, which returns information on all available metrics.

When you query statistics, you provide a list of IDs to specify the metrics in which you are interested. You also supply a start time, an end time, a roll-up interval, and a roll-up function. These values interact as follows to determine the data returned to you.

- The response contains a list of data points for each metric ID you specified in the request.
- The start time and end time control the limits for the data you want in the response. The response contains data points only for statistics that have timestamps between those limits, inclusive of the endpoints. However, the start time is adjusted to a round number, in some cases. For more information, see Statistics Interval Adjustment in the vCenter Server Appliance.

- The roll-up interval enables you to control the granularity of the data points in the response. Rather than a response with a data point for every statistic between the start time and end time, you get a response with a number of data points equal to the number of intervals between the start and end times. Generally, you should specify a time period that is an even multiple of the interval, so that each data point in the response represents the same number of statistics.
- The roll-up function specifies how the response summarizes the statistics that fall within each interval. The resulting data point can be the maximum statistic value within collection interval, or the mean of the statistics values within the interval, and so on.

Statistics Collection Times

The actual time that a statistic was collected is not readily predictable.

The API does not enable you to determine the exact time that a statistic was collected. Furthermore, some statistics, such as those for storage metrics, might take seconds or minutes to collect, so that they are not available immediately at the time a request is made to the guest operating system.

However, because statistics are collected at regular intervals, and roll-up intervals for a request generally all have the same size, each data point in the response represents the same number of statistics as the others. See Statistics Interval Adjustment in the vCenter Server Appliance for more information.

Statistics Interval Adjustment in the vCenter Server Appliance

When you make a request for statistics, the monitoring service might adjust the specified roll-up interval times to improve the appearance of statistics graphs in a graphical interface.

The monitoring service adjusts the start time of a data collection request when it is not an exact multiple of the interval length. In these instances, the start time is rounded downward to the previous UTC time that is a multiple of the interval. All subsequent intervals of the data collection are also adjusted to align with the new start time.

For example, if the start time is 10:31 and the interval length is 1 hour, the monitoring service adjusts the start time to 10:00 and the roll-up intervals have the following continuous pattern.

- 10:00 to 10:59:59.999
- 11:00 to 11:59:59.999
- 12:00 to 12:59:59.999

The monitoring service does not adjust the end time of a data collection. Consequently, the response to a statistics query might contain one more data value than expected, or an incomplete final interval might be lengthened.

Empty Data Values

In some instances, you might encounter a response that reports an empty data value, or even a series of empty data values. This might manifest as a list of data values containing some numeric values alternating with empty values.

- Empty data values can happen when the report time period is too short to be certain of containing any statistics. For instance, a time period of 30 seconds is half the length of the sample period for network metrics, so you have only a 50% chance of finding a network statistic during any 30-second reporting period.
- Empty data values can also happen when the interval is shorter than the sample period for a metric you have requested. In this case, some data points are present in the list, while others are empty because no statistic was collected during those intervals. For instance, an interval of 5 minutes is only half the length of the sample period for storage metrics, so every second data value is empty.
- Empty data values can also happen when the monitoring service has not finished collecting and writing the last sample to the database, even if the nominal sample timestamp falls within the report time period. For example, calculation of storage used can delay writing a storage statistic to the database. A request for the statistic during that delay time produces an empty data point in the response.

When a response contains an empty data value, this indicates that no statistics were collected during a collection interval. An appropriate action for the client in such a case depends on how the client is using the data. For example, if you are graphing a resource usage trend, you might choose to interpolate for the missing value to produce a smooth line.

Check Database Usage in the vCenter Server Appliance

The vCenter Server Appliance contains a database of all objects managed by the vCenter Server instance. In addition to inventory objects, the database includes vCenter Server statistics, events, alarms, and tasks. You can calculate the database storage consumption by adding the sizes of all data categories.

You need to monitor storage consumption in the vCenter Server Appliance.

Prerequisites

This task assumes you have previously authenticated and created a client session.

Procedure

1 Prepare a request for database usage statistics.

  Include metric IDs both for `vcdb_core_inventory` and `vcdb_seat`. The name `vcdb_seat` refers to Statistics, Events, and Tasks in the vCenter Server database.

2 Issue the request to the API endpoint.

3 Process the resulting data points as needed.

4 Format and print the results.

The result of this procedure shows the storage used in the vCenter Server database, which includes storage overhead used for indexes and other optimizations beyond the actual size of the data.

Python Example of Checking Database Usage in the vCenter Server Appliance

This example shows the use of Python with the vSphere Automation SDK for Python to view recent statistics for the vCenter database usage in the vCenter Server Appliance. The example assumes a previously existing session with the vSphere Automation API endpoint.

This example requests statistics at 30-minute intervals for a recent 2-hour report period. The example requests the storage used by the inventory component and the storage used by the combination of statistics, events, alarms, and tasks. The example adds the two values to calculate the vCenter Server database usage in each 30-minute roll-up interval, and then reports the maximum size found over the 2-hour report period.

```
from com.vmware import appliance_client
import datetime

# This example assumes you have previously created a session
# and stored the session ID in my_stub_config.

# Issue request for core inventory and 'SEAT' (stats, events, & alarms) usage.
req = appliance_client.Monitoring.MonitoredItemDataRequest()
req.names = ['storage.used.filesystem.vcdb_core_inventory',
             'storage.used.filesystem.vcdb_seat']
req.interval = appliance_client.Monitoring.IntervalType.MINUTES30
req.function = appliance_client.Monitoring.FunctionType.MAX
d_now = datetime.datetime.utcnow()
 req.start_time = d_now - datetime.timedelta( minutes=135 )
 req.end_time = d_now - datetime.timedelta( minutes=15 )
 Monitoring_stub = appliance_client.Monitoring( my_stub_config )
 resp = Monitoring_stub.query( req )

 # Extract resulting arrays.
 core_sizes = resp[0].data
 seat_sizes = resp[1].data
 # Remove empty data points:
 core_sizes = filter( (lambda x: x != ''), core_sizes )
 seat_sizes = filter( (lambda x: x != ''), seat_sizes )

 # Add the usage stats for each interval, and display maximum usage.
 highest = max( map( (lambda a,b: int(a) + int(b)),
                     core_sizes, seat_sizes ) )
 print( 'vCenter database inventory + stats, events, alarms, tasks:' +
         ' (max) size = {0} KB'.format( highest ) )
```

List Storage Consumption By Data Type in the vCenter Server Appliance

The vCenter Server Appliance provides statistics on several types of storage used in the appliance.

For example, you can query statistics about inventory storage, transaction log, and vCenter Server tasks. Many of these statistics are available both for storage consumed and storage available.

This task provides data for system administrators who need to monitor storage consumption in the guest operating system of the vCenter Server Appliance.

Prerequisites

Verify that you have authenticated and created a client session.

Procedure

1 Prepare a request for database usage statistics.

  Include metric IDs for each data type you wish to monitor.

2 Issue the request to the API endpoint.

3 Process the resulting data points as needed.

4 Format and print the results.

Python Example of Listing Storage Consumption By Data Type in the vCenter Server Appliance

This example shows how to use the Monitoring interface to break down database usage by data type. The example requests the individual data types that you can also query as a composite metric for all storage used by Alarms, Statistics, Events, and Tasks in the vCenter Server instance.

```
from com.vmware import appliance_client
import datetime

# This example assumes you have previously created a session
# and stored the session ID in my_stub_config.

# Prepare request for chosen data types.
req = appliance_client.Monitoring.MonitoredItemDataRequest()
req.interval = appliance_client.Monitoring.IntervalType.MINUTES30
req.function = appliance_client.Monitoring.FunctionType.MAX
d_now = datetime.datetime.utcnow()
req.start_time = d_now - datetime.timedelta( minutes=30 )
req.end_time = d_now
mon = {'storage.totalsize.directory.vcdb_hourly_stats' :
         'Hourly stats',
       'storage.totalsize.directory.vcdb_daily_stats' :
         'Daily stats',
       'storage.totalsize.directory.vcdb_monthly_stats' :
         'Monthly stats',
        'storage.totalsize.directory.vcdb_yearly_stats' :
          'Yearly stats',
        'storage.totalsize.directory.vcdb_events' :
          'Events',
        'storage.totalsize.directory.vcdb_alarms' :
          'Alarms',
        'storage.totalsize.directory.vcdb_tasks' :
          'Tasks'}

req.names = []
for item in mon.keys() :
   req.names.append( item )

# Issue request.
Monitoring_stub = appliance_client.Monitoring( my_stub_config )
resp = Monitoring_stub.query( req )

# Assemble data from response.
out = {}
for metric in resp :
  # Discard empty data points:
  stat = ''
  while (stat == '') :
    stat = metric.data.pop()
  stat = int(stat)
  out[mon[metric.name]] = stat

# Format and print statistics.
for label in sorted( out.keys() ) :
  print( '{0:15s}: {1:8d} KB'.format( label, out[label] ) )
```

Maintenance of the vCenter Server Appliance

8

The vCenter Server Appliance Management API facilitates backup and restore operations.

You can create a backup file that includes the database and configuration of the vCenter Server instance. You can also use the API to restore the backup file into a freshly deployed appliance.

This chapter includes the following topics:

Backing up the vCenter Server Appliance

The vCenter Server Appliance Management API supports backing up key parts of the appliance. This allows you to protect vCenter Server data and to minimize the time required to restore data center operations.

The backup process collects key files into a tar bundle and compresses the bundle to reduce network load. To minimize storage impact, the transmission is streamed without caching in the appliance. To reduce total time required to complete the backup operation, the backup process handles the different components in parallel.

You have the option to encrypt the compressed file before transmission to the backup storage location. When you choose encryption, you must supply a password which can be used to decrypt the file during restoration.

The backup operation always includes the vCenter Server database and system configuration files, so that a restore operation has all the data needed to re-create an operational appliance. Current Alarms are included as well. You also have the option to specify additional data sets, called parts. In this release, you can specify a data set that includes Statistics, Events, and Tasks.

Backup and Restore Protocols for the vCenter Server Appliance

The vCenter Server Appliance backup and restore feature supports a number of plug-in communication protocols.

Choose one of these protocols as the backup location type when you invoke the operation.

- FTP

- FTPS
- SCP
- HTTP
- HTTPS

The value `PATH` for the location type field indicates a locally mounted volume.

Note If you specify the `SCP` protocol, you must specify an absolute path as the value of the location type field when you create the backup job.

Calculate the Size Needed To Store the Backup File

When you prepare to do a backup of the vCenter Server Appliance, you can use the API to calculate the storage space needed for the backup file.

You can do this task when you are choosing a backup storage location or whenever your existing storage location may be approaching full capacity.

Prerequisites

- Verify that you have a vCenter Server Appliance running.
- Verify that you are familiar with authentication methods. See Chapter 4 Authentication Mechanisms.

Procedure

1. Authenticate to the vSphere Automation API endpoint and establish a session.
2. Request a list of backup parts available.
3. For each available backup part, request the size of the backup file.

   The backup process calculates the compressed size of each backup part.
4. Choose which parts to include in the backup, and sum their sizes.

   The backup storage server must have sufficient space to contain the chosen parts.

What to do next

After you choose which backup parts you will store, and verify that the backup storage server has sufficient free space, you can launch a backup job. For information, see Back up a vCenter Server Appliance by Using the API.

Python Example of Calculating the Size Needed To Store the Backup Image

This example shows how to use Python to collect the information you need to calculate the size needed to store a backup image of the vCenter Server instance.

```
from com.vmware.appliance.recovery.backup_client import Parts

 # This example assumes you have previously created a session
 # and stored the session ID in my_stub_config.

 # Issue a request to list the backup image parts.
 Parts_stub = Parts( my_stub_config )
 parts = Parts_stub.list()

 # Extract IDs of backup image parts.
 sizes = {}
 total = 0
 for part in parts :
    size = Parts_stub.get( part.id )
    sizes[part.id] = size
    total += size

 # Show the result.
 print( 'Backup image parts:' )
 for part_id in sizes.keys() :
    print( '  part {0} = {1}KB'.format( part_id, sizes[part_id] ) )
    print( 'Total size: {0}KB'.format( total ) )
```

Back up a vCenter Server Appliance by Using the API

You can use the Management API of the vCenter Server Appliance to create a backup of the vCenter Server database and key components of the appliance.

This procedure explains the sequence of operations you use to create a backup file of the vCenter Server instance in the appliance. You can do this as part of a regular maintenance schedule.

Prerequisites

- Verify that the vCenter Server instance is in a ready state. All processes with start-up type automatic must be running.
- Verify that no other backup or restore jobs are running.
- Verify that the destination storage location is accessible to the appliance backup process.
- Verify that the path to the destination directory already exists, as far as the parent directory.
- If the destination directory does not exist, the backup process will create it. If the directory does exist, verify that it is empty.

- Verify that the destination storage device has sufficient space for the backup file. For information about how to calculate the space needed for the backup file, see Calculate the Size Needed To Store the Backup File.

Procedure

1 Authenticate to the vSphere Automation API endpoint and establish a session.

2 Create a backup request object to describe the backup operation.

   The request specifies several attributes, especially the backup location, the protocol used to communicate with the storage server, the necessary authorization, and which optional parts of the database you want to back up. The core inventory data and Alarms are always backed up, but you can choose whether or not to back up Statistics, Events, and Tasks. Collectively, this optional part of the backup is referred to as `seat`.

3 Issue a request to start the backup operation.

4 From the response, save the unique job identifier of the backup operation.

5 Monitor progress of the job until it is complete.

6 Report job completion.

Python Example of Backing Up a vCenter Server Appliance

This example specifies that the backup image should include Statistics, Events, and Tasks as well as the core inventory and alarm data. The value for `req.parts` indicates the optional data part for Statistics, Events, and Tasks.

This example uses the following global variables.

- *my_storage_server*
- *my_backup_folder*
- *my_scp_user*
- *my_scp_password*
- *my_stub_config*

When you back up the vCenter Server instance, you need two sets of authentication credentials. The API client needs to authenticate to the vCenter Server Appliance, and the Appliance backup service needs to authenticate to the backup storage server.

The example assumes that your API client has already authenticated the connection to the vCenter Server Appliance, and the security context is stored in *my_stub_config*.

In the backup request, you need to specify the folder that will contain the backup image. The folder name must be specified as a path name relative to the home directory of the user that authenticates with the storage server.

```
from com.vmware.appliance.recovery.backup_client import Job
import time

# This example assumes you have previously created a session
# and stored the session ID in my_stub_config.

# Create a backup request object.
req = Job.BackupRequest()
# Include optional backup part for Statistics, Events, and Tasks.
req.parts = ['seat']
req.location_type = Job.LocationType.SCP
req.comment = 'On-demand backup'
req.location = my_storage_server + ':/home/scpuser/' + my_backup_folder \
  + '/' + time.strftime('%Y-%m-%d-%H-%M-%S')
req.location_user = my_scp_user
req.location_password = my_scp_password

# Issue a request to start the backup operation.
backup_job = Job( my_stub_config )
job_status = backup_job.create( req )
job_id = job_status.id

# Monitor progress of the job until it is complete.
while (job_status.state == Job.BackupRestoreProcessState.INPROGRESS) :
    print( 'Backup job state: {} ({}%)'.format( job_status.state, \
                                             job_status.progress ) )
    time.sleep( 10 )
    job_status = backup_job.get( job_id )

# Report job completion.
print( 'Backup job completion status: {}'.format( job_status.state) )
```

Restoring the vCenter Server Appliance

The vCenter Server Appliance Management API supports restoring the appliance from a backup copy. The API simplifies the process by unifying the handling of various components of vCenter Server in a single operation.

The process of restoring a vCenter Server Appliance from a backup has two phases.

1 Deploy a new appliance. OVF deployment is described in the *vSphere Automation SDKs Programming Guide*.

2 Invoke the `restore` operation from the Management API to apply configuration settings and load the vCenter Server database from the backup file.

Note You cannot specify optional parts for the restore operation. The restore operation includes all optional parts, such as Events and Tasks, that were specified at the time when the backup file was created.

Authentication When Restoring the vCenter Server Appliance

During the process of restoring the vCenter Server Appliance from a backup image, you cannot use vCenter Single Sign-On authentication. You must use local authentication until the appliance is fully configured.

When you restore your vCenter Server Appliance from a backup file, it begins in an unconfigured state. During this time, you must use local authentication to access the Management API. When you use local authentication, do not use the vSphere Automation API endpoint. Instead, you must connect your client to port 5480 of the appliance.

When you use local authentication you must pass user name and password credentials with each method invocation. Use credentials that are known to the guest operating system of the appliance.

Availability of Services While Restoring the vCenter Server Appliance

During the process of restoring the vCenter Server backup file in the vCenter Server Appliance, services in the appliance must restart. While they are restarting, your API client receives an error message.

You can write your client to trap the error, but you have no way to know when the Appliance services are running again. To determine when the restore process is complete, you must retry the API connection until it succeeds, then request the status of the job.

Restore the vCenter Server Appliance by Using the API

You can use the Management API of the vCenter Server Appliance to restore the appliance from a backup file containing the vCenter Server database and key components of the appliance.

Prerequisites

- Verify that the backed up vCenter Server Appliance is powered off.
- A new vCenter Server Appliance must be deployed in an unconfigured state, except that it must have a fully qualified domain name or IP address that matches the old one.
- Verify that the new vCenter Server Appliance has the same build number as the one in the backup file.
- Verify that the new vCenter Server Appliance has a size equal to or greater than the old one. If the old vCenter Server Appliance was customized to exceed the largest template size, the new one must be customized to the same size.

- If the old vCenter Server Appliance was deployed with an external Platform Services Controller, the new one must be also. If the old one was deployed in an embedded configuration, the new one must be also.
- Verify that no other backup or restore jobs are running.
- Verify that the destination storage location is accessible to the appliance restore process.

Procedure

1. Create a restore request object to describe the restore operation.
2. Issue a request to start the restore operation.
3. Monitor progress of the job until it is complete.
4. Report job completion.

What to do next

After the vCenter Server Appliance is fully configured by the restore operation, you can resume using the vSphere Automation API endpoint for subsequent operations.

Python Example of Restoring the vCenter Server Instance

This example shows how to use Python to restore a vCenter Server instance in a newly deployed vCenter Server Appliance. This operation is the second phase of restoring the appliance from a backup image.

This example uses the following global variables.

- *my_vcsa_hostname*
- *my_vcsa_username*
- *my_vcsa_password*
- *my_backup_name*
- *my_storage_server*
- *my_scp_user*
- *my_scp_password*
- *my_backup_folder*

When you restore the vCenter Server instance from a backup image, you need two sets of authentication credentials. The API client needs to authenticate to the vCenter Server Appliance, and the appliance backup service needs to authenticate to the backup storage server.

The example uses local username and password authentication for the connection to the vCenter Server Appliance because the vSphere Automation API endpoint is not yet running when you restore the appliance. The client must connect to port 5480 for this operation.

In the restore request, you need to specify the folder that contains the backup image. The folder name is the same name that was specified in the backup request. It must be specified as a path name relative to the home directory of the user that authenticates with the storage server.

This example assumes the backup image is not encrypted.

```
import requests
 from vmware.vapi.lib.connect import get_requests_connector
 from vmware.vapi.security.user_password import create_user_password_security_context
 from vmware.vapi.stdlib.client.factories import StubConfigurationFactory
 from com.vmware.appliance.recovery.restore_client import (Job)
 import time

 # Create a session object in the client.
 session = requests.Session()

 # For development environment only, suppress server certificate checking.
 session.verify = False
 from requests.packages.urllib3 import disable_warnings
 from requests.packages.urllib3.exceptions import InsecureRequestWarning
 disable_warnings(InsecureRequestWarning)

 # Create a connection to Appliance port 5480.
 local_url = 'https://%s:5480/api' % my_vcsa_hostname
 connector = get_requests_connector(session=session, url=local_url)

 # Add username/password security context to the connector.
 basic_context = create_user_password_security_context(my_vcsa_username, my_vcsa_password)
 connector.set_security_context(basic_context)

 # Create a stub configuration by using the username-password security context.
 local_stub_config = StubConfigurationFactory.new_std_configuration(connector)

 # Create a restore request object.
 req = Job.RestoreRequest()
 req.location_type = Job.LocationType.SCP
 req.location = my_storage_server + ':/home/scpuser/' + my_backup_folder + '/' + my_backup_name
 req.location_user = my_scp_user
 req.location_password = my_scp_password

 # Issue a request to start the restore operation.
 restore_job = Job( local_stub_config )
 job_status = restore_job.create( req )

 # Monitor progress of the job until it is complete.
 while (job_status.state == Job.BackupRestoreProcessState.INPROGRESS) :
     print( 'Restore job state: {} ({}%)'.format( job_status.state,
 job_status.progress ) )
     time.sleep( 10 )
     job_status = restore_job.get()

 # Report job completion.
 print( 'Restore job completion status: {}'.format( job_status.state) )
```