



Version 1.1.0

Controller REST API Developer Guide

302-000-496

01

EMC²

Copyright © 2013-2014 EMC Corporation. All rights reserved. Published in USA.

Published February, 2014

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (<https://support.emc.com>). For documentation on EMC Data Domain products, go to the EMC Data Domain Support Portal (<https://my.datadomain.com>).

EMC Corporation
Hopkinton, Massachusetts 01748-9103
1-508-435-1000 In North America 1-866-464-7381
www.EMC.com

CONTENTS

Chapter 1	Getting Started with the ViPR Controller REST API	7
	Welcome to the ViPR Controller REST API.....	8
	ViPR REST API access.....	8
	API Ports.....	8
	Identifying resources.....	9
	Controller API authentication overview.....	9
	Building an API super user.....	10
	Authentication and cookies.....	11
	Authentication redirects.....	12
	Logout.....	15
	Whoami.....	15
	Proxyuser.....	16
	Using proxyuser to run a controller REST API script.....	16
	Destroying a proxy token.....	17
Chapter 2	Setting Up a Single Tenant Environment	19
	Single Tenancy vs Multitenancy.....	20
	Adding users to the provider tenant with authentication providers.....	20
	Adding an authentication provider.....	20
	Changing the user mapping in the provider tenant.....	22
	Assigning roles to a tenant.....	23
Chapter 3	Authorization	25
	Authorization overview.....	26
	Virtual data center roles.....	26
	Tenant roles.....	28
	ACLs.....	29
	Virtual array and virtual pool ACLs.....	29
	Examples: Virtual array and virtual pool ACL APIs.....	30
	Project ACLs.....	31
	Examples: Project ACL APIs.....	32
	Examples: Changing a project's owner.....	33
Chapter 4	ViPR REST API Resources	35
	ViPR services overview.....	36
	System resources.....	38
	System resources: virtual array.....	38
	System resources: virtual pool.....	39
	System resources: networks and network systems.....	41
	API review: storage systems.....	49
	Tenant and project resources.....	50
Chapter 5	Setting Up a Multi-tenant Environment	51
	Prerequisites for creating multiple tenants	52
	Example format: XML.....	52
	Configuring multiple tenants with the REST API.....	52

Chapter 6	Common Operations	59
	Bulk operations.....	60
	Searching API resources.....	61
	Tagging API resources.....	62
	Tracking asynchronous operations.....	62
	Deactivating, or decommissioning, resources.....	63
Chapter 7	Setting Up a Virtual Data Center	65
	Setting up the virtual data center.....	66
	Example format: JSON.....	66
	Adding a storage system.....	66
	Registering network systems.....	67
	Registering a RecoverPoint protection system.....	68
	Host.....	70
	Registering hosts and assigning initiators and IP interfaces.....	70
	Setting up the virtual array.....	72
	Setting up the virtual pool.....	83
	Creating a project for a tenant.....	90
Chapter 8	Managing File Systems and Snapshots	91
	File system.....	92
	Creating a file system.....	92
	Expanding a file share.....	94
	Exporting a file system.....	96
	Unexporting a file share.....	97
	Deleting a file system export.....	99
	Deleting a file system.....	101
	Snapshot.....	102
	Creating a file system snapshot.....	102
	Exporting a file system snapshot.....	104
	Restoring a file system snapshot.....	106
	Unexporting a file system snapshot.....	107
	Deleting a file system snapshot.....	109
Chapter 9	Managing Volumes	111
	Create block volume.....	112
	Exporting a volume to a host.....	114
	Creating a block volume snapshot.....	116
	Deleting a block volume snapshot.....	118
	Unexporting a volume from a host.....	121
	Deleting a block volume.....	123
Chapter 10	Managing Hosts	125
	Host API Overview.....	126
	Adding and discovering a Windows host.....	127
	Windows clusters.....	128
	Adding and discovering a LINUX host.....	130
	Adding and discovering a vCenter Server.....	131
	Examples: Accessing vCenter resources.....	133

Chapter 11	Discovering Storage Systems	137
Registering and discovering physical storage systems in VNX Block and VMAX	138
Registering and discovering a VPLEX	139
Registering and discovering an Isilon storage system	140
Registering and discovering a NetApp storage system	142
Registering and discovering a VNX File storage system	143
Chapter 12	User Interface Services	145
Catalog, Services, Orders, and Approvals	146
Accessing UI services	148
UI Services response formats	148
Approval requests	149
Approving or rejecting	149
Retrieving a list of pending approvals	150
URL notification	150
Asset options	151
Upstream values	151
Executing a service	153
Service catalog	154
Service descriptors	156
Field descriptors	156
Field validation	157
Type field values	157
Tracking orders	158
Chapter 13	Error Codes and HTML Return Codes	161
Error code descriptions	162
HTTP return codes	163

CONTENTS

CHAPTER 1

Getting Started with the ViPR Controller REST API

This chapter contains the following topics.

◆ Welcome to the ViPR Controller REST API	8
◆ ViPR REST API access	8
◆ Identifying resources	9
◆ Controller API authentication overview	9
◆ Logout	15
◆ Whoami	15
◆ Proxyuser	16

Welcome to the ViPR Controller REST API

The ViPR Controller REST API describes the programmatic interfaces that allow you to create, read, update, and delete resources in a ViPR data center.

This guide should be used in conjunction with the *ViPR REST API Reference*, which contains a comprehensive description of all ViPR resources, and the REST API calls that manage them.

This guide assumes that you have ViPR installed. For information on installing ViPR, refer to the *EMC ViPR Installation and Configuration Guide*.

ViPR REST API access

The ViPR REST API is accessible using any web browser or programming platform that can issue HTTP requests.

For example, after you log into ViPR, you could type the following URL into a browser to retrieve the current user's tenant.

Request

```
GET https://<ViPR_VIP>:4443/tenant
```

Response

```
HTTP 200
{
    "name": "Provider Tenant",
    "id": "urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:",
    "link": {
        "rel": "self",
        "href": "/tenants/
urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:"
    }
}
```

- ◆ <ViPR_VIP> is the IP address or fully qualified domain name (FQDN) of the public virtual IP address of your ViPR vApp.
- ◆ Port 4443 is the HTTP port used for REST client access to many services on the ViPR virtual appliance. Other REST services are accessed through port 443. (See [API Ports on page 8](#).)

To issue POST, PUT, and DELETE HTTP requests from a web browser, you need browser plugins, such as Internet Explorer's [httpAnalyzer](#). Most major browsers, such as [Firefox](#) and [Chrome](#), have a poster plugin that will allow you to call HTTP commands directly from your browser.

You can also access the REST API using scripting platforms such as curl and perl. EMC also provides a Java client that wraps the ViPR Controller REST API calls in a set of Java classes.

API Ports

The ViPR Controller REST API uses two different ports.

The following table show the services available on each of the two ports used by the ViPR REST API.

REST API Service	Port	Description
ViPR API	4443 (HTTPS)	<p>Send ViPR REST API calls to port 4443 to manage resources controlled by the following services:</p> <ul style="list-style-type: none"> • Block services (volumes, consistency groups, snapshots, exports) • File services (file systems, snapshots) • Computer services (cluster, host, host discovery, ip interface, initiators, vCenter data center, vCenter) • Virtual Data Center services (block virtual pools, file virtual pools, networks, Storage systems, storage pools, and others.) • Tenant and Project services
ViPR UI Services API	443 (HTTPS)	<p>Send ViPR REST API calls to port 443 to manage the following resources:</p> <ul style="list-style-type: none"> • Approvals • Asset options • Catalog • Execution window • Orders • Schema

Identifying resources

Each ViPR managed storage resource is uniquely identified by an ID that is generated by ViPR when it is created. A URN is a Uniform Resource Name.

The URN for a resource is a unique identifier for the resource and cannot be changed.

To find the URN of a particular resource, you can use the `GET {resource_URL}/bulk` API call, which gives you a list of all the URNs of a given resource class. Then you can use the URNs obtained from this GET call and use them in the payload of the `POST {resource_URL}/bulk` call to obtain more detailed information about the specified resources.

See [Bulk Operations on page 60](#) for more information on how to retrieve URNs and details associated with a given resource URN.

Controller API authentication overview

Before you call any ViPR Controller REST API, you must successfully authenticate a user.

Clients pass security credentials to ViPR using [Basic HTTP Authentication](#). All API requests are delivered over secure sockets (https). ViPR accepts security credentials, and uses them to authenticate the user against LDAP or Active Directory.

You can login to ViPR by calling this API:

Request

```
GET https://<ViPR_VIP>:4443/login?using-cookies
```

Response

ViPR challenges the user with a dialog box. Enter the name of the user in the **User Name** field, and the password of the user in the **Password** field.

The ViPR API is always accessed over secure sockets, so use HTTPS instead of HTTP.

ViPR_VIP is the IP address or Fully Qualified Domain Name(FQDN) of the virtual IP of the ViPR vApp.

Use port 4443 for logging in to ViPR with the API.

With some application development platforms, you can pass in user credentials without interacting with the ViPR login dialog box. For example, this curl command allows you to log in as the root user with a password of **ChangeThis**:

```
curl <ViPR_VIP>:4443/login -u "root:ChangeThis" -k
```

where <ViPR_VIP> is <https://<ViPR Public IP Address>>.

Note

The <ViPR_VIP> token is used throughout this documentation.

Building an API super user

The user with which you log in to ViPR must have access to the ViPR resources you want to manage. It may be convenient to build a general-purpose super user with wide access to ViPR to run your API scripts.

The root user is a built-in user in ViPR. It is commonly used for initial ViPR setup, and can be used by a Controller REST API application as a super user. The root user has the following roles:

- ◆ SECURITY_ADMIN
- ◆ SYSTEM_ADMIN
- ◆ SYSTEM_MONITOR
- ◆ TENANT_ADMIN
- ◆ SYSTEM_AUDITOR

In some data centers, the root user may be rendered unavailable by your data center administrator. You can assign the roles used by root to a standard LDAP user by calling the following REST calls:

- ◆ PUT <ViPR_VIP>:4443/tenants/{id}/role-assignments
- ◆ PUT <ViPR_VIP>:4443/vdc/role-assignments

To call either of these APIs, you need to be authenticated as a user with the SECURITY_ADMIN role.

[Authorization on page 25](#) contains a full description of the user roles you can assign in a ViPR environment.

[Setting up a single tenant environment on page 19](#) contains a full description of setting up ViPR users and tenants, and assigning roles to those users.

Authentication and cookies

ViPR uses a token-based authentication system for all its public API calls. Once a user is authenticated against ViPR, an authentication token is returned and can be used to authenticate the user in subsequent calls.

Authentication tokens expire after eight hours or after two hours of idle time. Once expired, the token is internally destroyed and any attempt to call an API with that token returns an HTTP 401 code. This code indicates you need to login and authenticate again to obtain a new token.

You can retrieve and use authentication tokens by:

- ◆ Reading the X-SDS-AUTH-TOKEN HTTP header from a successful authentication request and copying that header into any subsequent request.
- ◆ Saving the X-SDS-AUTH-TOKEN cookie from a successful authentication request and sending that cookie along in subsequent requests.

Authentication with cookies

This example shows how to use authentication tokens by reading the X-SDS-AUTH-TOKEN http header from a successful authentication request, copying that information into a cookie, then passing the cookie in a subsequent request. The examples here are written in curl and formatted for readability.

```
curl -L --location-trusted
      -k https://<ViPR_VIP>:4443/login?using-cookies=true
      -u "root:Password"
      -c cookiefile
      -v
```

In this example, you specify the ?using-cookies=true parameter to indicate that you want to receive cookies in addition to the normal HTTP header. This curl command saves the authentication token to a file named cookiefile in the current directory. The next command passes the cookie with the authentication token through the -b switch, and returns the user's tenant information.

```
curl -k https://10.247.100.247:4443/tenant -b cookiefile -v

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tenant_info><name>Provider Tenant</name>
  <link href="/tenants/
urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:</id>
</tenant_info>
```

Authentication without cookies

This example shows how to use authentication tokens by reading the X-SDS-AUTH-TOKEN http header from a successful authentication request and copying that header into a subsequent request. This example does not use cookies. The examples here are written in curl and formatted for readability.

```
curl -L --location-trusted -k https://10.247.100.247:4443/login -u "root:ChangeMe" -v

> GET /login HTTP/1.1
> Authorization: Basic cm9vdDpDaGFuZ2VNzQ==
> User-Agent: curl/7.24.0 (i386-pc-win32) libcurl/7.24.0 OpenSSL/0.9.8t zlib/1.2.5
> Host: 10.247.100.247:4443
> Accept: */*
```

```
>
< HTTP/1.1 200 OK
< Date: Tue, 26 Nov 2013 22:18:25 GMT
< Content-Type: application/xml
< Content-Length: 93
< Connection: keep-alive
< X-SDS-AUTH-TOKEN:
BAAcQ0xOd3g0MjRCUG4zT3NJdnNuM1AvQTFYb1NrPQMAUAQADTEzODU0OTQ4NzYzNTICAA
EABQA5dXJu
```

```
OnN0b3JhZ2VvczpUb2tlbj02MjIxOTcyZS01NGUyLTRmNWQtYWZjOC1kMGE3ZDJmZDU3Mm
U6AgAC0A8=
<
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<loggedIn>
    <user>root</user>
</loggedIn>
* Connection #0 to host 10.247.100.247 left intact
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):
```

This executes a GET on the /login resource. The -u option indicates the user of basic authentication header. The user designation must be included in the request. Upon successful authentication, a HTTP 200 code is returned as well as the X-SDS-AUTH-TOKEN header containing the encoded token.

The token can then be passed back in the next API call. You can copy the X-SDS-AUTH-TOKEN contents and pass it to the next request through curl's -H switch.

```
curl https://10.247.100.247:4443/tenant
-k
-H "X-SDS-AUTH-TOKEN:
BAAcOHZLaGF4MT13eFhpY0czZ0tWUGHJV2xreUE4PQMAUAQADTEzODU0OTQ4NzYzNTICAA
EABQA5dXJu
```

```
OnN0b3JhZ2VvczpUb2tlbjpkYzc3ODU3Mi04NWRmLTQ2YjMtYjgwZi05YTdlNDFkY2QwZD
g6AgAC0A8="
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tenant_info>
    <name>Provider Tenant</name>\n
    <link href="/tenants/\nurn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:"\n
rel="self"/>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:</\n
id>
</tenant_info>
```

Authentication redirects

If you try to access a ViPR REST resource without a valid token, ViPR will issue an HTTP 302 redirect code.

An HTTP 302 redirect code includes the URL of the authentication resource, appended with a service= parameter that indicates where to be redirected after successful authentication. It also includes a signature parameter to prevent forgery on the service= parameter.

Handling authentication redirects without using cookies

This example shows what to do when you authenticate with an invalid security token or no token. This example does not use cookies. The examples in this section are written in curl.

In this example, the initial request for the current user's tenant returns an HTTP 302 error. Note the following:

- ◆ The X-SDS-AUTH-TOKEN header has to be copied into each request.
- ◆ The X-SDS-AUTH-TOKEN header is a custom HTTP header.
- ◆ If you are not using cookies, HTTP clients that use the **automatically follow redirects** option need to disable it. Automatically following redirects would mean the client follows all HTTP 302 responses without copying the custom header. This results in an authentication failure.

Procedure

1. Request the current user's tenant.

Request

```
curl -k "<ViPR_VIP>:4443/tenant" -v
```

Response

```
GET /tenants HTTP/1.1
User-Agent: curl/7.24.0 (i386-pc-win32) libcurl/7.24.0 OpenSSL/
0.9.8t zlib/1.2.5
Host: 10.247.100.247:4443
Accept: */*

HTTP/1.1 302 Found
Date: Wed, 27 Nov 2013 15:30:13 GMT
Content-Length: 0
Connection: keep-alive
Location: <ViPR_VIP>/login?service={LocationString}
```

2. Perform a GET against the location in the response body.

Request

```
curl -k "<ViPR_VIP>:4443/login?service={LocationString} -v
```

Response

```
HTTP/1.1 401 Unauthorized
```

```
WWW-Authenticate: basic realm="ViPR"
```

3. Present basic authentication credentials to the URL returned in step 1.

Request

```
curl -k "<ViPR_VIP>:4443/login?service={LocationString} -v -u
"root:ChangeMe"
```

On successful authentication, you see another 302 code, this time redirecting you to the original service. The authentication token is also in the HTTP header or cookie (depending on whether ?using-cookies=true appeared in the original request).

Response

```
GET /login?service={LocationString} HTTP/1.1
> Authorization: Basic cm9vdDpDaGFuZ2VNZQ==
> User-Agent: curl/7.24.0 (i386-pc-win32) libcurl/7.24.0 OpenSSL/
0.9.8t zlib/1.2.5
> Host: 10.247.100.247:4443
> Accept: */*
>
< HTTP/1.1 302 Found
```

```

< Date: Wed, 27 Nov 2013 16:53:28 GMT
< Content-Type: application/xml
< Content-Length: 0
< Connection: keep-alive
< Location: https://10.247.100.247:4443/tenant?auth-redirected
< X-SDS-AUTH-TOKEN: {Token_String}

```

4. Access the location, making sure to supply either the cookie or the http header.

Request

```
curl -k <ViPR_VIP>:4443/tenant?auth-redirected -H "X-SDS-AUTH-TOKEN:{token_text}"
```

Response

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tenant_info>
<name>Provider Tenant</name>
<link href="/tenants/
urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
<id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:</id>
</tenant_info>

```

Authentication redirects using cookies

Using cookies, you can automatically follow redirects.

The following curl example allows access to the API resource by passing credentials to it:

Request

```
curl -k "<ViPR_VIP>:4443/tenant?using-cookies=true" -u "root:ChangeMe" -c cookie1 -b cookie1 -L -v
```

Response

HTTP 302

In this example, curl's `-L` option indicates that we want to follow redirects.

Response Body

```

GET /tenant?using-cookies=true HTTP/1.1
Authorization: Basic cm9vdDpDaGFuZ2VNzQ==
User-Agent: curl/7.24.0 (i386-pc-win32) libcurl/7.24.0 OpenSSL/0.9.8t
zlib/1.2.5
Host: 10.247.100.247:4443
Accept: */*
Cookie: X-SDS-AUTH-
TOKEN=BAAcV1M5TkkwdnRvUFBJbXJkbzVqSzB3azzZBQ0VnPQMAUAQADTEzODU0OTQ4NzYz
NTICAAEABQA5dXJuOnN0b3JhZ2VvczpUb2tlbj03OGM4OD1jOS1hZTE5LTQ2NTgtODYxNS
00ZDk5YTY
xNWVmOTU6AgAC0A8=


HTTP/1.1 200 OK
Date: Wed, 27 Nov 2013 18:57:12 GMT
Content-Type: application/xml
Content-Length: 276
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tenant_info>
<name>Provider Tenant</name>
<link href="/tenants/urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
<id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:</id>
</tenant_info>

```

Logout

The logout API ends a session.

A given user is allowed a maximum of 100 concurrent authentication tokens. Past this limit, the system refuses any new connection for this user until tokens free up. They can free up by expiring naturally, or by explicitly calling this URI:

`https://<ViPR_VIP>:4443/logout`

If you have multiple sessions running simultaneously, this URI forces the termination of all tokens related to the current user.

`GET <ViPR_VIP>:4443/logout?force=true`

An example logout request follows.

Request

`GET: <ViPR_VIP>:4443/logout`

`X-SDS-AUTH-TOKEN: {Auth_Token}`

Pass in the header or cookie with the authentication token to logout.

Response

`HTTP 200`

Whoami

A ViPR user can view their own user name, tenant association, and roles using the whoami API call.

Request

`GET <ViPR_VIP>:4443/user/whoami`

Response

`HTTP 200`

```
GET /user/whoami
<user>
  <common_name>root</common_name>
  <distinguished_name>root</distinguished_name>
  <roles>
    <role>SYSTEM_AUDITOR</role>
    <role>SECURITY_ADMIN</role>
    <role>SYSTEM_ADMIN</role>
    <role>SYSTEM_MONITOR</role>
    <role>TENANT_ADMIN</role>
  </roles>
  <subtenant_roles/>
  <tenant>
    urn:storageos:TenantOrg:7985d438-9980-41df-bba1-29d6a873f811:
  </tenant>
</user>
```

This example shows the whoami output for the Root user. Root is associated with the tenant indicated in the `<tenant>` field. Root has the roles listed in the `<roles>` field.

Proxyuser

Because standard ViPR security tokens expire after 8 hours, ViPR provides a special user ID that can run a Controller REST API script on a schedule without having to repeatedly log in. For example, you might set up a script to check particular file services every 12 hours, and take appropriate action in the presence of certain conditions.

The proxy token feature allows a user to retrieve a persistent security token from ViPR, then pass that token to a special user called proxyuser, who runs the script. The proxyuser is a built-in user in ViPR. This user has the PROXY_USER role, and is the only ViPR user that can have that role.

The proxyuser cannot perform any security-related operations. For example, a proxy user cannot register an authentication provider, or do role assignments for a user. The proxyuser is best used for monitoring and provisioning operations.

Using proxyuser to run a controller REST API script

ViPR provides a way to run a Controller REST API script on a schedule without having to repeatedly log in. For example, you might set up a script to check particular file services every 12 hours, and take appropriate action in the presence of certain conditions. The examples in this section are written in `curl`, and formatted for readability.

Before you begin

You must have a user from your LDAP repository mapped to a tenant in ViPR.

Note

Built-in users, such as root, can also use proxy tokens.

Procedure

1. Log in to ViPR.

```
curl -k <ViPR_VIP>:4443/login?using-cookies -c cookie2 -u "myUser@MyCompany.com:ChangeThis"
```

ViPR delivers a standard ViPR authentication token. (This token has an 8 hour timeout.)

2. Retrieve your user's proxy token. Each ViPR user has one – and only one – proxy token.

```
curl -k <ViPR_VIP>:4443/proxytoken -b cookie2 -v
```

```
GET /proxytoken HTTP/1.1
User-Agent: curl/7.24.0 (i386-pc-win32) libcurl/7.24.0 OpenSSL/0.9.8t zlib/1.2.5
Host: 10.247.100.247:4443
Accept: /*
Cookie: X-SDS-AUTH-TOKEN={Token_Text}
```

```
HTTP/1.1 200 OK
Date: Wed, 27 Nov 2013 20:05:02 GMT
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
X-SDS-AUTH-PROXY-TOKEN: {Token_Text}
```

The user's proxy token is contained in the header X-SDS-AUTH-PROXY-TOKEN.

3. Schedule your script to run – say, once every 12 hours. You can use standard scheduling software provided by your platform operating system to do this. For example, CRON is available for most LINUX workstations.

4. Pass the proxy token - X-SDS-AUTH-PROXY-TOKEN - for your user to the proxyuser. How you do this is specific to your application. One method is to save the token to a file that is accessible by the proxyuser.
5. The proxyuser logs in, and uses the LDAP user's proxy token to run the scheduled script as that user. For example, if `vipr_user` passes a proxy token to `proxyuser`, `proxyuser` can run `vipr_user`'s script as `vipr_user` using `vipr_user`'s proxy token. The proxy token does not age out. The script can run repeatedly for an indefinite period of time.

Destroying a proxy token

A proxy token does not expire, but it can be destroyed by the user who created it, or by a ViPR SECURITY_ADMIN. The examples in this section are written in curl, and formatted for readability.

To destroy your own proxy token, call:

```
curl -k "<ViPR_VIP>:4443/logout?force=true&proxytokens=true" -b cookie1 -v
```

```
GET /logout?force=true&proxytokens=true HTTP/1.1
User-Agent: curl/7.24.0 (i386-pc-win32) libcurl/7.24.0 OpenSSL/0.9.8t
zlib/1.2.5
Host: 10.247.100.247:4443
Accept: */
Cookie: X-SDS-AUTH-TOKEN={Token_Text}

HTTP/1.1 200 OK
Date: Wed, 27 Nov 2013 20:49:06 GMT
Content-Type: application/xml
Content-Length: 95
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LoggedOut>
  <user>root</user>
</LoggedOut>
```

To destroy another user's proxy token, call:

```
curl -k "<ViPR_VIP>:4443/logout?
force=true&proxytokens=true&username={ViPR_User}" -b cookie1 -v
```


CHAPTER 2

Setting Up a Single Tenant Environment

This chapter contains the following topics.

- ◆ [Single Tenancy vs Multitenancy](#)..... 20
- ◆ [Adding users to the provider tenant with authentication providers](#)..... 20

Single Tenancy vs Multitenancy

ViPR automatically includes a provider tenant.

Tenants allow you to separate groups of users and give those groups access to different resources in your data center. In some environments, a single tenant is all that is required to manage your data center. In other environments, you may need multiple tenants. For example, if you are a service provider, a multi-tenant environment may be necessary to separate users and protect their data.

This chapter describes how to use authentication providers to map LDAP users into the ViPR provider tenant. For information on building a multi-tenant environment with the ViPR REST API, see [Setting up a Multi-tenant Environment on page 51](#).

Adding users to the provider tenant with authentication providers

Authentication providers are added to ViPR to allow users from Active Directory (AD) or Lightweight Directory Access Protocol (LDAP) v3 compliant servers to authenticate users and allows those users to participate in ViPR operations.

The first authentication provider you add to ViPR maps users from Active Directory or LDAP into the provider tenant.

Once an authentication provider is registered, tenants can be configured to restrict access to a subset of users and groups obtained from the provider. For example, if an authentication provider is created for domain mycompany.com, the provider tenant can be configured to only accept users from this domain by adding a tenant user mapping to it.

User mappings for a tenant can be as simple as a single domain name, or as complex as a combination of domain names, group names and specific user attributes found in the LDAP/AD schema.

You can use authentication providers to create multi-tenancy environments where you create two or more groups of users , and give those groups access to different resources in your data center. [Setting up a Multi-tenant Environment on page 51](#) describes the API calls required to set up a multi-tenant environment.

Adding an authentication provider

An authentication provider defines a set of LDAP or Active Directory users who are initially assigned to the provider tenant. Each of those users has access to ViPR resources according to the roles and ACLs assigned to the provider tenant.

Before you begin

In this example, you login as the root user. By default, the root user has the Security Administrator (SECURITY_ADMIN) role on the entire virtual data center. The root user is allowed to add an authentication provider for the mycompany.com domain.

Procedure

1. Login to ViPR as the root user and obtain an authentication token that can then be used in subsequent ViPR API calls.

Request

```
GET <ViPR_VIP>:4443/login?using-cookies
```

Provide credentials to satisfy the authentication challenge.

Response

```
HTTP 200
X-SDS-AUTH-TOKEN:{Token_Text}

<loggedIn>
  <user>root</user>
</loggedIn>
```

2. POST an authentication provider for the mycompany.com domain.

Request

```
POST <ViPR_VIP>:4443/vdc/admin/authnproviders

{ 'Content-Type': 'application/xml',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/xml' }

<authnprovider_create>
  <server_urls>
    <server_url>ldap://10.247.100.165</server_url>
  </server_urls>
  <domains>
    <domain>sanity.local</domain>
  </domains>
  <group_whitelist_values>
    <group_whitelist_value>*Admin*</group_whitelist_value>
    <group_whitelist_value>*Test*</group_whitelist_value>
  </group_whitelist_values>
  <mode>ad</mode>
  <name>ad configuration2</name>
  <description>ad configuration2</description>
  <disable>no</disable>
  <manager_dn>CN=Administrator,CN=Users,DC=mycompany,DC=com</manager_dn>
  <manager_password>P@ssw0rd</manager_password>
  <search_base>CN=Users,DC=mycompany,DC=com</search_base>
  <search_filter>userPrincipalName=%u</search_filter>
  <search_scope>SUBTREE</search_scope>
  <group_attribute>CN</group_attribute>
</authnprovider_create>
```

The *ViPR Installation and Configuration Guide* has more information about the meaning of each authentication provider parameter.

Response

```
HTTP 200
```

```
<authnprovider>
  <creation_time>1386017748095</creation_time>
  <id>urn:storageos:AuthnProvider:0555dc65-598f-4dea-951a-d9f5b2b61c5e:</id>
  <inactive>false</inactive>
  <link href="/vdc/admin/authnproviders/urn:storageos:AuthnProvider:0555dc65-598f-4dea-951a-d9f5b2b61c5e;" rel="self"/>
  <name>ad configuration2</name>
  <tags/>
  <description>ad configuration2</description>
  <disable>false</disable>
  <domains>
    <domain>mycompany.com</domain>
  </domains>
  <group_attribute>CN</group_attribute>
  <group_whitelist_values>
    <group_whitelist_value>*Admin*</group_whitelist_value>
    <group_whitelist_value>*Test*</group_whitelist_value>
  </group_whitelist_values>
  <manager_dn>CN=Administrator,CN=Users,DC=mycompany,DC=com</manager_dn>
  <mode>ad</mode>
```

```

<search_base>CN=Users,DC=mycompany,DC=com</search_base>
<search_filter>userPrincipalName=%u</search_filter>
<search_scope>SUBTREE</search_scope>
<server_urls>
    <server_url>ldap://192.247.100.165</server_url>
</server_urls>
</authnprovider>

```

Changing the user mapping in the provider tenant

In this example, you modify the provider tenant to include a subset of users from the domain mycompany.com.

Before you begin

Follow the instructions in the topic [Adding an authentication provider on page 20](#) to set up an authentication provider. This example uses the same authentication token as that procedure. Headers are hidden for brevity.

In this example, you modify the provider tenant to only map users from the groups AdminUsers and TestUsers in the domain mycompany.com. Users in the domain mycompany.com that are not members of the AdminUsers and TestUsers groups will no longer be mapped into the provider tenant.

Modify the provider tenant by sending a PUT request to the resource

```
PUT <ViPR_VIP>:4443/tenants/{tenant_urn}
```

where {tenant_urn} is the URN ID of the tenant. In this example, you authenticate as root and are modifying the provider tenant.

Procedure

- Obtain the URN ID of the provider tenant. The following API call returns the current user's tenant. The current user is root.

Request

```
GET <ViPR_VIP>:4443/tenant
```

Response

```
HTTP 200
```

```
GET /tenant
```

```

<tenant_info>
    <name>Provider Tenant</name>
    <link href="/tenants/
urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f;">
    <rel="self"/>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:</id>
</tenant_info>

```

- Use that URN ID to access the provider tenant and modify it, adding the desired user mapping.

Request

```
PUT <ViPR_VIP>:4443/tenants/{Tenant_URN}
```

```

<tenant_update>
    <name>Provider Tenant</name>
    <user_mapping_changes>
        <add>
            <user_mapping>
                <domain>mycompany.com</domain>
                <groups>
                    <group>TestUsers</group>
                    <group>AdminUsers</group>

```

```

        </groups>
        </user_mapping>
    </add>
</user_mapping_changes>
</tenant_update>
```

Response

HTTP 200

```

<tenant>
    <creation_time>1383585703830</creation_time>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:</id>
    <inactive>false</inactive>
    <link href="/tenants/
urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
    <name>Provider Tenant</name>
    <tags/>
    <description>Root Provider Tenant</description>
    <user_mappings>
        <user_mapping>
            <attributes/>
            <domain>mycompany.com</domain>
            <groups>
                <group>TestUsers</group>
                <group>AdminUsers</group>
            </groups>
        </user_mapping>
    </user_mappings>
</tenant>
```

For the mapping to be valid, AdminUsers and TestUsers need to be available to ViPR through the white list parameter of the authentication provider that this tenant uses. The *Admin* and *Test* expressions (see [Adding an authentication provider on page 20](#)) make this possible.

Assigning roles to a tenant

Once an authentication provider is available and users can be mapped to a tenant, roles can be assigned to users and groups.

For example, the Tenant Administrator (TENANT_ADMIN) role can be assigned to a user. This allows that user to perform administrative tasks on the tenant, such as looking at the tenant information. It also allows that user to assign roles to other users within the tenant.

Assigning the Tenant Administrator role to an Active Directory user

By assigning the TENANT_ADMIN role to a user, you allow that user to perform administrative tasks on the tenant, such as looking at the tenant information.

Before you begin

You must be logged in as a user with the SECURITY_ADMIN role.

The tenantadmin@mycompany.com user is a valid AD user who is part of a group called AdminUsers. This user is eligible to be TENANT_ADMIN because the tenant lists AdminUsers as one of the acceptable groups for user mapping.

Procedure

1. Call the following API to apply the role assignment to `tenantadmin@mycompany.com`.

Request

```
PUT <ViPR_URL>:4443/tenants/{tenant_URN}/role-assignments.json
```

```
{
  "add": [{"subject_id": "tenantadmin@mycompany.com", "role": ["TENANT_ADMIN"]}]
}
```

where tenant_URN is the ID of the ViPR tenant to which you are applying the role assignment. The following API returns the URLs of all tenants known to ViPR.

```
GET <ViPR_URL>:4443/tenants/bulk
```

Response

```
HTTP 200
```

```
{
  "link": {"rel": "self", "href": "/tenants/{tenant_URN}/role-assignments"},
  "role_assignments":
  [
    {
      "role":
      ["TENANT_ADMIN"], "subject_id": "tenantadmin@mycompany.com",
      {"role": ["TENANT_ADMIN"], "subject_id": "root"}
    }
  ]
}
```

2. To test out that tenantadmin@mycompany.com has TENANT_ADMIN privileges in this tenant, login with that user and request role assignment information for the provider tenant.

Request

```
GET <ViPR url>:4443/login.json -u
tenantadmin@mycompany.com:password
```

Response

```
HTTP 200
```

```
X-SDS-AUTH-TOKEN:{Token_text}

{
  "user": "tenantadmin@mycompany.com"
}
```

Request

```
GET <ViPR_URL>:4443/tenants/{Provider_Tenant_URI}/role-assignments.json
```

```
X-SDS-AUTH-TOKEN:{Token_text}
```

Response

```
HTTP 200
```

```
{
  "link": {"rel": "self", "href": "/tenants/
urn:storageos:TenantOrg:d082a5b2-3201-4faf-9e62-ae1c989f1995:/role-assignments"},
  "role_assignments":
  [
    {
      "role":
      ["TENANT_ADMIN"], "subject_id": "tenantadmin@mycompany.com",
      {"role": ["TENANT_ADMIN"], "subject_id": "root"}
    }
  ]
}
```

CHAPTER 3

Authorization

This chapter contains the following topics.

◆ Authorization overview	26
◆ Virtual data center roles	26
◆ Tenant roles	28
◆ ACLs	29
◆ Virtual array and virtual pool ACLs	29
◆ Project ACLs	31

Authorization overview

Authorization in ViPR is controlled by roles and Access Control Lists (ACLs). There are two types of roles (virtual data center roles and tenant level roles), and different types of ACLs.

Virtual data center roles

Virtual data center roles are scoped to the entire ViPR virtual data center. Note that they can only be assigned to users and groups that belong to the top level provider tenant. These roles define what the user can do at the virtual data center level.

The virtual data center level roles are:

- ◆ Security Administrator
- ◆ System Administrator
- ◆ System Monitor
- ◆ System Auditor

A user with a Security Administrator role can assign roles to users. In the example below, a Security Administrator assigns the Security Admin role to a user with a subject_id of `username@mycompany.com`.

Request:

```
PUT <ViPR_VIP>/vdc/role-assignments
```

Request body

```
<role_assignment_change>
  <add>
    <role>SECURITY_ADMIN</role>
    <subject_id>username@mycompany.com</subject_id>
  </add>
</role_assignment_change>
```

Response

HTTP 200

```
<role_assignments_create>
  <role_assignment>
    <role>SYSTEM_ADMIN</role>
    <subject_id>username@mycompany.com</subject_id>
  </role_assignment>
  <role_assignment>
    <role>SYSTEM_AUDITOR</role>
    <role>SECURITY_ADMIN</role>
    <role>SYSTEM_MONITOR</role>
    <role>SYSTEM_ADMIN</role>
    <subject_id>username@mycompany.com</subject_id>
  </role_assignment>
  <link href="/vdc/role-assignments" rel="self"/>
</role_assignments_create>
```

The following table lists the authorized actions for each ViPR virtual data center level role.

Virtual Data Center-Level Role	Authorized Actions
Security Administrator	<ul style="list-style-type: none"> • Manages the authentication provider configuration for the ViPR virtual data center to identify and authenticate users. Authentication

Virtual Data Center-Level Role	Authorized Actions
	<p>providers are configured to add specified users into ViPR using existing Active Directory/Lightweight Directory Access Protocol (AD/LDAP) user accounts/domains.</p> <ul style="list-style-type: none"> Assigns roles and project ACLs to users or groups. Configures ACLs for virtual arrays and virtual pools to control which tenants may use them. Restores access to tenants and projects, if needed. (For example, in the event that the Tenant Administrator locks himself/herself out, the Security Administrator can reset user roles to restore access.) Changes ViPR virtual data center properties. Shuts down/reboots/restarts ViPR services. Manages ViPR virtual data center software and license updates.
System Administrator	<ul style="list-style-type: none"> Sets up the physical storage infrastructure of the ViPR virtual data center and carves the physical storage into two types of virtual resources: virtual arrays and virtual pools. Authorized actions include: <ul style="list-style-type: none"> Adding physical storage resources into ViPR such as arrays, ports, pools, switches, and networks. Creating virtual pools; defining storage capabilities and assigning physical storage pools to virtual pools. Creating virtual arrays; partitioning storage into discrete pods of compute, network, and storage resources to control file system/volume/object pathing through SAN/IP networks. Sets up object storage in the ViPR virtual data center; this includes creating the object virtual pool and data stores, assigning an IP network to the Object Data Service, and assigning an object namespace to the tenant. Configures Access Control Lists (ACLs) for virtual arrays and virtual pools to control which tenants may use them. Manages the ViPR virtual data center resources that are not managed by tenants. Retrieves ViPR virtual data center status and health information. Retrieves bulk event and statistical records for the ViPR virtual data center.
System Monitor	<ul style="list-style-type: none"> Has read-only access to all resources in the ViPR virtual data center. Retrieves bulk event and statistical records for the ViPR virtual data center. Does not have visibility into security-related resources, such as authentication providers, ACLs, and role assignments. Retrieves ViPR virtual data center status and health information.
System Auditor	Has read-only access to the ViPR virtual data center audit logs.

Tenant roles

Tenant roles are scoped to the tenant in which the assignment is made. Tenant Administrator, Tenant Approver, and Project Administrator are tenant-level roles.

Tenant roles can be assigned to users/groups from the corresponding tenant. These roles define what the user can do at the tenant level. The tenant level roles are:

- ◆ Tenant Administrator
- ◆ Tenant Approver
- ◆ Project Administrator

A user with a Security Admin or Tenant Admin role can assign roles to users within a tenant. The URL indicates the tenant. In the following example, a Security or Tenant Administrator with access to the tenant assigns a Tenant Administrator role to a user with a subject ID of user@mycompany.com.

PUT: <ViPR url>:4443/tenants/<tenant urn>/role-assignments.json

Request body:

```
{
  "add": [ {"subject_id": "user@mycompany.com", "role": [
    "TENANT_ADMIN"
  ]}]
}
```

Response:

HTTP 200

The following table lists the authorized actions for each ViPR tenant-level role.

Tenant-Level Role	Authorized Actions
Tenant Administrator	<ul style="list-style-type: none"> • Creates tenants under the provider tenant (in a multi-tenant environment). (In a single-tenant enterprise private cloud environment, there is just the provider tenant and the Tenant Administrator(s) has access to all projects.) • Maps AD/LDAP users into their tenant to define who can log into the tenant. • Creates, modifies, and deletes projects in their tenant. • Manages resources in the tenant's projects. • Configures ACLs for projects and the Service Catalog in their tenant. • Assigns roles to tenant users. (Can assign Tenant Administrator or Project Administrator roles to other users.)
Tenant Approver	<ul style="list-style-type: none"> • Approves or rejects Service Catalog orders in their tenant. • Views all approval requests in their tenant.
Project Administrator	<ul style="list-style-type: none"> • Creates projects in their tenant, getting an OWN ACL on the created project.

ACLs

An Access Control List (ACL) is a list of permissions attached to a ViPR resource that specifies which tenants are authorized to access that resource as well as what operations are allowed on that resource.

ACLs can be assigned to the following ViPR resources:

- ◆ Block virtual pool
- ◆ File virtual pool
- ◆ Projects
- ◆ Virtual arrays

Note

Only Project ACLs are managed through the ViPR user interface. Other ACLs must be managed using the API.

For each virtual array, virtual pool, or project resource, there is a limit of 100 ACLs. In other words, there is a maximum of 100 user/group assignments for projects and 100 tenant assignments for a virtual array or virtual pool.

Virtual array and virtual pool ACLs

At creation time, virtual arrays and virtual pools are public. They are accessible to all tenants.

A System or Security Administrator can assign an ACL to a virtual pool or virtual array to restrict them to be used by specified tenants.

The ACL permission associated with virtual arrays and pools is of the type USE. If a specific tenant has a USE ACL on a virtual pool, this means that all the users who are mapped to that tenant will be allowed to use that virtual pool in their provisioning operations.

All newly created virtual arrays and pools will have an empty ACL. The System or Security Administrator is responsible for managing the ACL. If no ACLs are set, the virtual arrays and pools remain accessible to the provider tenant and all other tenants in the ViPR system.

For virtual pools and virtual arrays, you cannot assign an ACL to a user (subject ID) or group. Only tenants can be assigned ACLs for these resources.

The following table shows the APIs that allow a system or security administrator to modify ACLs for virtual arrays and virtual pools. Note that there are separate API calls for file and block virtual pools.

API	Description
GET /block/vpools/{id}/acl	Show ACL assignment for block store virtual pool .
PUT /block/vpools/{id}/acl	Add or remove block store virtual pool ACL entries.
GET /file/vpools/{id}/acl	Show ACL entries for file store VirtualPool.
PUT /file/vpools/{id}/acl	Add or remove ACL entries from file store VirtualPool.
GET /vdc/varrays/{id}/acl	Show VirtualArray ACL Virtual Array.

API	Description
PUT /vdc/varrays/{id}/acl	Add or remove ACL for VirtualArray Virtual Array.

Examples: Virtual array and virtual pool ACL APIs

The examples in this section show some commonly-used APIs for managing virtual array and virtual pool ACLs.

Virtual array: Assigning the USE ACL to a tenant

The following example shows how to give a tenant privileges to use a virtual array. If no ACL exists on the virtual array, all tenants have access to it.

Request

```
PUT https://<ViPR_VIP>:4443/vdc/varrays/
urn:storageos:VirtualArray:f49f6e36-0fe5-4181-9622-49d116204d86:/acl

<acl_assignment_changes>
  <add>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:7985d438-9980-41df-
bba1-29d6a873f811:</tenant>
  </add>
</acl_assignment_changes>
```

Response:

HTTP 200

```
<acl_assignments>
  <acl_assignment>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:7985d438-9980-41df-
bba1-29d6a873f811:</tenant>
  </acl_assignment>
</acl_assignments>
```

Virtual array: Removing the USE ACL from a tenant

Request

```
PUT https://<ViPR_VIP>:4443/vdc/varrays/
urn:storageos:VirtualArray:f49f6e36-0fe5-4181-9622-49d116204d86:/acl

<acl_assignment_changes>
  <remove>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:7985d438-9980-41df-
bba1-29d6a873f811:</tenant>
  </remove>
</acl_assignment_changes>
```

Response:

HTTP 200

```
<acl_assignments/>
```

Virtual pool: Assigning the USE ACL to a tenant

The following example shows how to give a tenant privileges to use a virtual pool. If no ACL exists on the virtual pool, all tenants have access to it.

Request

```
PUT <ViPR_VIP>/file/vpools/urn:storageos:VirtualPool:4394653f-
cf2e-4301-8f11-9e8d3e7e7fa9:/acl

<acl_assignment_changes>
  <add>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:</tenant>
  </add>
</acl_assignment_changes>
```

Response:

HTTP 200

```
<acl_assignments>
  <acl_assignment>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:</tenant>
  </acl_assignment>
</acl_assignments>
```

Project ACLs

A Tenant Administrator can access all projects for their tenant. Project Administrators can only access projects that they own.

The ACL permissions associated with projects are listed in the following table.

The project ACLs can be created or modified by a Tenant Administrator, a Security Administrator, or a project owner. Project owners are assigned the OWN ACL. The user that creates the project is the owner of that project unless they, or a tenant administrator, transfers ownership of that project to another user.

The default ACL behavior of a project is different from the default ACL behavior of a Virtual Array or Virtual Pool. Whereas, the default ACL for a Virtual Array or Virtual Pool enables anyone to use them, the default ACL for the Project prevents all but the Tenant Admin, Security Admin, or Project owner from using it. For other users or groups to use a project, that user or group must be explicitly added to the ACL for that project.

Project ACL	Description
ALL	The user can perform create, read, update, and delete (CRUD) operations on file systems, volumes, snapshots, exports, and buckets within the project.
BACKUP	The user has read-only access to volumes, file systems and buckets in the project. The user can create, delete, and export snapshots for volumes and file systems in the project.
OWN	The user with the OWN ACL is the project owner. A project owner can do all of the following: <ul style="list-style-type: none"> • Perform CRUD operations on project resources. • Set ACLs on the project. • Transfer ownership of the project to another user. • Delete the project. • Set project properties, such as the project name and owner.

Project ACL	Description
ANY	The ANY ACL appears in the <i>EMC ViPR REST API Reference Guide</i> . It means "any user that has either the BACKUP or ALL ACL".

Newly created projects will have an empty ACL. The Tenant Administrator or the project owner is responsible for managing the ACL.

Examples: Project ACL APIs

The examples in this section show some commonly-used APIs for managing project ACLs.

Get the ACLs for a project

Request

```
GET /projects/{Project_URN}/acl
```

Response

```
<acl_assignments>
  <acl_assignment>
    <privilege>ALL</privilege>
    <subject_id>jordab@sanity.local</subject_id>
  </acl_assignment>
  <acl_assignment>
    <privilege>BACKUP</privilege>
    <subject_id>jordab2@sanity.local</subject_id>
  </acl_assignment>
</acl_assignments>
```

Assigning the USE ACL to a user

The following example sets the project ACL using a user's subject_id. A subject_id or group can be assigned the ALL or BACKUP permission.

Request

```
PUT <ViPR_VIP>/projects/<project_urn>/acl
<acl_assignment_changes>
  <add>
    <privilege>ALL</privilege>
    <subject_id>jordab2@sanity.local</subject_id>
  </add>
</acl_assignment_changes>
```

Response

HTTP 200

```
<acl_assignments>
  <acl_assignment>
    <privilege>ALL</privilege>
    <subject_id>jordab@sanity.local</subject_id>
  </acl_assignment>
  <acl_assignment>
    <privilege>ALL</privilege>
    <privilege>BACKUP</privilege>
    <subject_id>jordab2@sanity.local</subject_id>
  </acl_assignment>
</acl_assignments>
```

Examples: Changing a project's owner

The example in this section shows how to change the owner of a project.

The OWN ACL is assigned to a project's creator, giving that user ownership rights to that project. A tenant admin, a project admin or the project's owner can reassign the ownership of the project to another user.

Checking the owner of a project

The user that owns the project is displayed in the `<owner>` field of the project resource. Here, the user `jordab@sanity.local` is displayed as the project owner.

```
GET /projects/{Project_urn}
```

```
<project>
  <creation_time>1384272649906</creation_time>
  <id>urn:storageos:Project:2c4d5503-a935-405d-9795-3a08e4bd3ee3:</id>
  <inactive>false</inactive>
  <link href="/projects/urn:storageos:Project:2c4d5503-a935-405d-9795-3a08e4bd3ee3:" rel="self"/>
  <name>project1</name>
  <tags/>
  <owner>jordab@sanity.local</owner>
  <tenant>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:</id>
    <link href="/tenants/urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
  </tenant>
</project>
```

Changing the owner of a project

This example changes the ownership of the project shown in the previous example to `jordab2@sanity.local`. Note that this is done by changing the `owner` attribute of the project, rather than through an ACL call.

Request

```
PUT /projects/{Project_URN}
```

```
<project_update>
  <owner>jordab2@sanity.local</owner>
</project_update>
```

Response

HTTP 200

CHAPTER 4

ViPR REST API Resources

This chapter contains the following topics:

◆ ViPR services overview	36
◆ System resources	38
◆ Tenant and project resources	50

ViPR services overview

The ViPR Controller REST API includes services that allow you to programmatically manage a wide variety of resources in your data center.

This chapter describes each API service, and the data center components that those services manage. For specific information on each ViPR Controller REST API call, see the *EMC ViPR REST API reference*.

API Service	Description	Example API Calls
Block services	<p>ViPR supports block storage on VMAX, VNX Block and other arrays. The following ViPR resources are managed using Block Service API calls.</p> <ul style="list-style-type: none"> • Volumes • Protection systems (block) • Consistency groups • Snapshots • Exports 	POST /block/volumes GET /block/volumes/{id}
File services	<p>ViPR supports file storage on VNX file, NetApp and other arrays. The following ViPR resources are managed using File Service API calls.</p> <ul style="list-style-type: none"> • File systems • File system snapshots 	GET /file/filesystems/{id} POST /file/filesystems GET /file/snapshots/{id}
Compute services	<p>Compute services are used to manage hosts in a data center. These resources are managed through the compute services.</p> <ul style="list-style-type: none"> • Cluster • Host • Host discovery • IP interface • Initiator • vCenter data center • vCenter 	GET /compute/hosts/{id} PUT /compute/clusters/{id}
Virtual Data Center Services	<p>Virtual data center services manage resources that are built from the physical assets in your data center. Virtual data center services include:</p> <ul style="list-style-type: none"> • Authentication providers • Block virtual pools • File virtual pools 	GET /vdc/admin/authnproviders GET /block/vpools GET /file/vpools POST /vdc/networks/{id}/register POST /vdc/storage-systems/discover GET /vdc/varrays

API Service	Description	Example API Calls
	<ul style="list-style-type: none"> • Networks • SMI Providers • Storage systems • Virtual arrays <p>Setting up a virtual data center on page 65 describes how to set up a virtual data center using the ViPR Controller REST API.</p> <p>Setting Up a Single Tenant Environment on page 19 and Setting up a Multi-tenant Environment on page 51 describe how to use authentication providers to build single tenant and multi-tenant environments.</p>	
Tenant services	Tenant services allow developers to manage tenants, projects, and associated resources.	GET /tenants/{id} GET /projects/{id}/acl PUT /projects/{id} GET /tenants/{id}/role-assignments
UI Services	<p>The UI services include the following resources:</p> <ul style="list-style-type: none"> • Approvals • Asset options • Execution windows • Orders • Schema <p>UI Services on page 145 describes the UI Service resources and contains examples of how these are used.</p>	GET /api/approvals POST /api/services/{serviceId} POST /api/executionwindows GET /api/orders GET /api/schema.xsd
Other Services	<p>ViPR offers a number of services for logging in and out of ViPR, viewing audit logs, monitoring and metering, and building workflows.</p> <ul style="list-style-type: none"> • Audit • Authentication • User Info • Monitoring • Metering • Workflow 	GET /audit/logs GET /monitoring/events GET /metering/stats GET /vdc/workfloww GET /vdc/workflows/active

System resources

System resources are the physical and virtual data center resources managed by a user with a System Administrator role.

Physical storage resources are only visible to and managed by the System Administrator. These resources include:

- ◆ Physical storage systems (EMC VNX, VMAX, VPLEX, Isilon, and NetApp arrays).
- ◆ Storage Pools and Storage Ports that are discovered through SMI-S providers.
- ◆ Switches from Brocade and Cisco. (Switches are called Network-Systems in the ViPR API).
- ◆ Auto-tiering policies such as Fully Automated Storage Tiering (FAST).
- ◆ IP and fibre channel network connectivity. Networks are ViPR resources that include switches, host ports, and array ports.

The System Administrator is also responsible for setting up two types of virtual resources: virtual arrays and virtual pools.

System resources: virtual array

A virtual array is a container of networks with their connected storage system ports and host initiator ports. A virtual array represents a realm of connectivity of host, network, and storage resources grouped together for purposes of fault tolerance.

Virtual arrays are set up to define and ensure connectivity from host to array. The resources in the virtual array allow for multiple potential paths between the hosts and arrays through your SAN. ViPR uses the hosts ports, array ports and switches in the virtual array to define multiple paths from your hosts to your arrays to enable the most redundant communication path possible.

The following table shows some important REST calls that manage virtual arrays.

ViPR REST API Call	Description
GET /vdc/varrays	Get a list of all virtual arrays configured in ViPR.
POST /vdc/varrays	Create a virtual array.
GET /vdc/varrays/{id}	Get information on a single virtual array.
POST /vdc/varrays/{id}/networks	Create a network for a virtual array. This call is needed to create an IP network. Fibre Channel networks are created by registering a network-system.
GET /vdc/varrays/{id}/storage-pools	Get a list of physical storage pools that have been assigned to the virtual array. This call displays pools explicitly assigned to the virtual array through this API: PUT /vdc/storage-pools/{id} If no pools have been explicitly assigned to the virtual array, GET /vdc/varrays/{id}/storage-pools returns all storage pools allocated on arrays whose endpoints are included in networks assigned to your virtual array.

ViPR REST API Call	Description
GET /vdc/varrays/{id}/vpools	Returns a list of virtual pools that have been assigned to a virtual array.

System resources: virtual pool

Virtual pools are created in ViPR by the System Administrator to define the storage service capabilities associated with file, block, and object data services.

The System Administrator associates physical storage pools to defined virtual pools. Once the physical storage pools are associated to virtual pools, the virtual pools can be used by self-service provisioning end users (no role required) to provision storage resources such as volumes, file systems, and buckets. Virtual pools are visible to self-service provisioning end users, but the underlying physical storage systems, pools, ports, and networks are not.

There is also an object virtual pool that is backed by file-based storage to store object data.

A virtual pool contains a very large amount of information about the underlying physical storage. The following shows an example of a file virtual pool.

Request

```
GET /block/vpools/urn:storageos:VirtualPool:2eaacbc2-a3a2-4491-9ddd-9d4c4c9ae2:
```

Response

```
<block_vpool>
  <creation_time>1382552786501</creation_time>
  <id>urn:storageos:VirtualPool:2eaacbc2-a3a2-4491-9ddd-9d4c4c9ae2:</id>
  <inactive>false</inactive>
  <link href="/block/vpools/urn:storageos:VirtualPool:2eaacbc2-a3a2-4491-9ddd-9d4c4c9ae2:" rel="self"/>
  <name>Pauls_virtual_pool thin block symm 1185</name>
  <tags/>
  <assigned_storage_pools>
    <storage_pool>
      <id>urn:storageos:StoragePool:a8e049d4-491a-4de2-ba2f-2f2be502601b:</id>
      <link href="/vdc/storage-systems/urn:storageos:StorageSystem:54c9a969-b8ba-4a68-862c-ac214f0453a9:/storage-pools/urn:storageos:StoragePool:a8e049d4-491a-4de2-ba2f-2f2be502601b:" rel="self"/>
    </storage_pool>
  </assigned_storage_pools>
  <description>Pauls_virtual_pool</description>
  <invalid_matched_pools>
    <storage_pool>
      <id>urn:storageos:StoragePool:fc9ae4f6-e32b-4503-97b6-fcbafe6eb4c3:</id>
      <link href="/vdc/storage-systems/urn:storageos:StorageSystem:6b23e681-b3da-4a7f-a4f0-4da6bf50cad9:/storage-pools/urn:storageos:StoragePool:fc9ae4f6-e32b-4503-97b6-fcbafe6eb4c3:" rel="self"/>
    </storage_pool>
    <!-- More Storage Pools -->
  </invalid_matched_pools>
  <matched_storage_pools/>
  <num_paths>1</num_paths>
  <num_resources>0</num_resources>
  <protocols>
    <protocol>FC</protocol>
```

```

</protocols>
<provisioning_type>Thin</provisioning_type>
<system_type>NONE</system_type>
<type>block</type>
<use_matched_pools>false</use_matched_pools>
<varrays>
    <varray>
        <id>urn:storageos:VirtualArray:339a534e-f69b-495b-87b2-
c95a4a8f71c9:</id>
        <link href="/vdc/varrays/urn:storageos:VirtualArray:339a534e-
f69b-495b-87b2-c95a4a8f71c9:" rel="self"/>
    </varray>
</varrays>
<drive_type>NONE</drive_type>
<expandable>true</expandable>
<multi_volume_consistency>false</multi_volume_consistency>
<protection>
    <snapshots>
        <max_native_snapshots>0</max_native_snapshots>
    </snapshots>
    <continuous_copies>
        <max_native_continuous_copies>0</max_native_continuous_copies>
    </continuous_copies>
</protection>
<raid_levels>
    <unique_auto_tier_policy_names>false</unique_auto_tier_policy_names>
</block_vpools>

```

The following table shows some commonly used calls for file, block, and object virtual pools.

ViPR REST API Call	Description
POST /block/vpools	Create a block virtual pool.
GET /block/vpools	Get a list of block virtual pools.
GET /block/vpools/{id}	Get information on a specific block virtual pool.
PUT /block/vpools/{id}	Update a block virtual pool.
POST /file/vpools	Create a file virtual pool.
GET /file/vpools	Get a list of file virtual pools.
GET /file/vpools/{id}	Get information on a specific file virtual pool.
PUT /file/vpools/{id}	Update a block file pool.
GET /object/data-services-vpools/{id}/data-stores	Get data stores
POST /object/data-services-vpools	Create a data services virtual pool
GET /object/data-services-vpools	List all data services virtual pools
GET /object/data-services-vpools/{id}	Get data services virtual pool details
PUT /object/data-services-vpools/{id}	Update data services virtual pool details
POST /object/data-services-vpools/{id}/deactivate	Delete data services virtual pool

System resources: networks and network systems

ViPR can manage switches and fabrics through the network and network-systems resources.

A network in a virtual array represents connectivity within the virtual array and the control path of volumes and file systems exposed by that virtual array.

ViPR supports two types of networks: Fibre Channel networks and IP networks. ViPR automatically discovers fibre channel networks during network-system (SAN switch) discovery. IP networks must be built manually by the System Administrator or ViPR API developer.

A network-system is the virtual representation of a physical switch. The most efficient method of adding fibre channel networks to ViPR is to register a network-system. When the Brocade or Cisco switch is registered, ViPR automatically executes a discovery on that switch. Any VSANs or FABRICS configured on the switch are ingested into ViPR as network resources. You can see the network-system resource referenced in the fibre channel network representation.

```
<network_systems>
  <network_system>urn:storageos:NetworkSystem:
76adbd2a-6827-405d-870d-79606342807b:</network_system>
</network_systems>
```

One network maps to each VSAN/fabric discovered on a switch with its collection of endpoints; that is, the storage system ports and host ports (initiator ports on a host ESX server) to which the switch is connected.

Note

In the ViPR user interface, a network-system is called a *Fabric Manager*.

API review - Fibre Channel networks

Networks are part of a virtual storage array. ViPR handles Fibre Channel networks differently from IP networks.

The following APIs allow the ViPR system administrator to create Fibre Channel networks and add them to virtual storage arrays. Note that fibre channel networks are not directly created through the API. Fibre channel networks are created by ViPR when you successfully discover either a Cisco or Brocade switch.

ViPR REST API Call	Description
/vdc/network-systems	Creates a network-system. To create Fibre Channel networks, you must:
/vdc/network-systems/{id}/register	<ul style="list-style-type: none"> • Create a network-system. • Register it with ViPR. <p>After a successful network-system registration, the fibre channel switch, its host ports and its array ports are discovered, and one or more networks is built from that information.</p>
GET /vdc/networks	List networks.
POST /vdc/networks/{id}/deactivate	Delete a network from ViPR.
POST /vdc/varrays	Creates a virtual array

ViPR REST API Call	Description
POST /vdc/varrays/{id}/networks	Add a network to the virtual array

The following shows a fibre channel network resource.

```
GET /vdc/networks/urn:storageos:Network:  
9afa585c-8411-43e8-9abc-8d1eb80d03e5:  
  
<network>  
  <creation_time>1382128881407</creation_time>  
  <id>urn:storageos:Network:9afa585c-8411-43e8-9abc-8d1eb80d03e5:</id>  
  <inactive>false</inactive>  
  <link href="/vdc/networks/urn:storageos:Network:  
9afa585c-8411-43e8-9abc-8d1eb80d03e5:" rel="self"/>  
  <name>FABRIC_losbe175</name>  
  <tags/>  
  <native_guid>FC+BROCADE+10:00:00:05:1E:35:C0:30</native_guid>  
  <discovered>true</discovered>  
  <endpoints>  
    <endpoint>21:00:00:1B:32:08:74:1F</endpoint>  
    <endpoint>10:00:00:00:C9:62:CB:10</endpoint>  
    <endpoint>50:06:01:60:47:20:26:1A</endpoint>  
    <endpoint>50:01:24:82:67:64:A0:C7</endpoint>  
  </endpoints>  
  <endpoints_discovered>  
    <endpoint_discovered>  
      <name>21:00:00:1B:32:08:74:1F</name>  
      <value>true</value>  
    </endpoint_discovered>  
    <endpoint_discovered>  
      <name>10:00:00:00:C9:62:CB:10</name>  
      <value>true</value>  
    </endpoint_discovered>  
    <endpoint_discovered>  
      <name>50:06:01:60:47:20:26:1A</name>  
      <value>true</value>  
    </endpoint_discovered>  
    <endpoint_discovered>  
      <name>50:01:24:82:67:64:A0:C7</name>  
      <value>true</value>  
    </endpoint_discovered>  
  </endpoints_discovered>  
  <fabric_id>losbe175</fabric_id>  
  <network_systems>  
    <network_system>urn:storageos:NetworkSystem:  
76adbd2a-6827-405d-870d-79606342807b:</network_system>  
  </network_systems>  
  <registration_status>REGISTERED</registration_status>  
  <transport_type>FC</transport_type>  
  <varray>  
    <id>urn:storageos:VirtualArray:339a534e-f69b-495b-87b2-  
c95a4a8f71c9:</id>  
    <link href="/vdc/varrays/urn:storageos:VirtualArray:339a534e-  
f69b-495b-87b2-c95a4a8f71c9:" rel="self"/>  
  </varray>  
</network>
```

API review: IP networks

Networks are part of a virtual storage array. Unlike fibre channel networks, IP networks must be built by the system administrator by hand and added to the virtual storage array.

The following APIs allow the ViPR system administrator to create IP networks and add them to a virtual storage array. Note that IP networks must be explicitly created and populated - they cannot be discovered like fibre channel networks.

ViPR REST API Call	Description
POST /vdc/varrays/{varray_id}/networks	Creates an IP network resource.
GET /vdc/networks	List all networks in the virtual data center.
POST /vdc/storage-systems	Add storage systems to ViPR.
POST /vdc/storage-systems/{id}/register	Register a storage system with ViPR
GET /vdc/storage-systems/{id}/storage-ports/{storage_port_id}	<p>Retrieve information on a storage port. The information required to add an array endpoint is contained in the <code>port_network_id</code> field in the response body.</p> <pre><port_network_id> 50:00:09:82:A1:7E:70:C4 </port_network_id></pre>
GET /compute/hosts/bulk	Get a list of ViPR hosts.
GET /compute/hosts/{id}	Get information about a host. The id provided here is the URN of a host. This returns an IP Interface for the host.
GET /compute/ip-interfaces/{ip_interface_id}	Retrieve information on the IP Interface. Information about the Host endpoint is in the <code>ip_address</code> field in the response body for this call.
PUT /vdc/networks/{id}/endpoints	Add storage ports and host ports to the IP network.

The following shows a ViPR representation of an IP network.

```
GET /vdc/networks/urn:storageos:Network:7e92077c-cd55-4264-
ae79-7b51cd1f35dc:
```

```
<network>
  <creation_time>1384888781885</creation_time>
  <id>urn:storageos:Network:7e92077c-cd55-4264-ae79-7b51cd1f35dc:</id>
  <inactive>false</inactive>
  <link href="/vdc/networks/urn:storageos:Network:7e92077c-cd55-4264-
ae79-7b51cd1f35dc:" rel="self"/>
  <name>bobs_ip</name>
  <tags/>
  <native_guid/>
  <discovered>false</discovered>
  <endpoints>
    <endpoint>10.247.166.180</endpoint>
  </endpoints>
  <endpoints_discovered>
    <endpoint_discovered>
      <name>10.247.166.180</name>
      <value>false</value>
    </endpoint_discovered>
  </endpoints_discovered>
  <registration_status>REGISTERED</registration_status>
  <transport_type>IP</transport_type>
  <varray>
    <id>urn:storageos:VirtualArray:b60def12-
a703-4be6-9505-369af471f6e7:</id>
    <link href="/vdc/varrays/urn:storageos:VirtualArray:b60def12-
a703-4be6-9505-369af471f6e7:" rel="self"/>
  </varray>
</network>
```

API review - network systems

Network systems are physical switches in your data center.

Network systems come in two types:

- ◆ Brocade
- ◆ Cisco

Brocade switches are discovered through CMCNE software, which must be installed on a host in your network. Cisco switches are discovered through an SMI-S provider.

ViPR REST API Call	Description
GET /vdc/network-systems	List switches registered with ViPR.
POST /vdc/network-systems/{id}/discover	Discover a network-system. This is rarely called directly by the API programmer. Network-system discoveries are performed automatically by ViPR when you successfully register a network system.
POST /vdc/network-systems/{id}/register	Register a network system.

Example: <ViPR_URL>/vdc/network-systems

In this example, the Brocade CMCNE host based management software is installed on a Windows host in the data center. This API call returns a list of network-systems. In this example, ViPR has only one network-system resource discovered.

Request

```
Get <ViPR_VIP>/vdc/network-systems
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<network_systems>
    <network_system>
        <id>urn:storageos:NetworkSystem:8ef79865-c829-48c6-84d1-
b184e63025c6:</id>
        <link href="/vdc/network-systems/urn:storageos:NetworkSystem:
8ef79865-c829-48c6-84d1-b184e63025c6:" rel="self"/>
        <name>CMCNE 10.247.71.183</name>
    </network_system>
</network_systems>
```

Example: <ViPR_URL>/vdc/network-systems/{network-system_URN}

This example shows the information returned for a single network-system resource.

Request

```
Get <ViPR_VIP>/vdc/network-systems/{network-system_URN}
```

Response

```
<network_system>
    <creation_time>1384377287345</creation_time>
    <id>urn:storageos:NetworkSystem:8ef79865-c829-48c6-84d1-
b184e63025c6:</id>
    <inactive>false</inactive>
    <link href="/vdc/network-systems/urn:storageos:NetworkSystem:
8ef79865-c829-48c6-84d1-b184e63025c6:" rel="self"/>
    <name>CMCNE 10.247.71.183</name>
    <tags/>
    <native_guid>BROCADE+10.247.71.183+5989</native_guid>
    <compatibility_status>COMPATIBLE</compatibility_status>
    <job_discovery_status>COMPLETE</job_discovery_status>
    <last_discovery_run_time>1384542614578</last_discovery_run_time>
```

```

<last_discovery_status_message>Discovery completed successfully
for Network System : urn:storageos:NetworkSystem:8ef79865-
c829-48c6-84d1-b184e63025c6:</last_discovery_status_message>
<last_metering_run_time>0</last_metering_run_time>
<job_metering_status>CREATED</job_metering_status>
<next_discovery_run_time>1384546213777</next_discovery_run_time>
<next_metering_run_time>0</next_metering_run_time>
<registration_status>REGISTERED</registration_status>
<system_type>brocade</system_type>
<ip_address>10.247.71.183</ip_address>
<port_number>5989</port_number>
<smis_port_number>5989</smis_port_number>
<smis_provider_ip>10.247.71.183</smis_provider_ip>
<smis_use_ssl>true</smis_use_ssl>
<smis_user_name>administrator</smis_user_name>
<uptime>2 days, 10 hours, 51 minutes, 33 seconds</uptime>
<user_name>administrator</user_name>
<version>11.2.1 build 165</version>
</network_system>

```

Adding an IP network to a virtual array

ViPR supports two types of networks: fibre channel and IP. This procedure shows how to build an IP network, and add it to a virtual array.

Before you begin

You must have the following resources available in ViPR:

- ◆ At least one storage system
- ◆ At least one host
- ◆ A virtual array

For simplicity, the following procedure shows how to add a single host port and a single array port to the IP network. Typically, an IP network would include multiple host and array ports.

Procedure

1. Retrieve the URN of a valid virtual array.

Request

```
GET /vdc/varrays
```

Response

```

<varrays>
  <varray>
    <id>urn:storageos:VirtualArray:339a534e-f69b-495b-87b2-
c95a4a8f71c9:</id>
    <link href="/vdc/varrays/urn:storageos:VirtualArray:339a534e-
f69b-495b-87b2-c95a4a8f71c9:" rel="self"/>
    <name>pauls_virtual_array</name>
  </varray>
  <varray>
    <id>urn:storageos:VirtualArray:b60def12-
a703-4be6-9505-369af471f6e7:</id>
    <link href="/vdc/varrays/urn:storageos:VirtualArray:b60def12-
a703-4be6-9505-369af471f6e7:" rel="self"/>
    <name>new test</name>
  </varray>
</varrays>

```

Use the ID of a listed virtual array to create an IP network resource in the next step.

2. Build a new IP network.

Request

```
POST /vdc/varrays/{id}/networks
```

```
<network_create>
  <name>NewIPNetwork</name>
  <transport_type>IP</transport_type>
</network_create>
```

Response

```
<network>
  <creation_time>1385067752563</creation_time>
  <id>urn:storageos:Network:26f3efbf-9afd-4a23-b828-ae5aba0d3a81:</id>
  <inactive>false</inactive>
  <link href="/vdc/networks/urn:storageos:Network:26f3efbf-9afd-4a23-b828-ae5aba0d3a81:" rel="self"/>
  <name>NewIPNetwork</name>
  <tags/>
  <native_guid/>
  <discovered>false</discovered>
  <endpoints/>
  <endpoints_discovered/>
  <registration_status>REGISTERED</registration_status>
  <transport_type>IP</transport_type>
  <varray>
    <id>urn:storageos:VirtualArray:339a534e-f69b-495b-87b2-c95a4a8f71c9:</id>
    <link href="/vdc/varrays/urn:storageos:VirtualArray:339a534e-f69b-495b-87b2-c95a4a8f71c9:" rel="self"/>
  </varray>
</network>
```

3. Get a list of hosts.**Request**

```
GET /compute/hosts/bulk
```

Response

```
<ids>
  <id>urn:storageos:Host:44295238-9d8e-44ad-940d-0272483b772e:</id>
  <id>urn:storageos:Host:4f8ccf42-f9fb-4088-a46f-791ed7afc110:</id>
  <id>urn:storageos:Host:62d77c0b-83e5-4fa8-a6e9-10b0291c1bad:</id>
  <id>urn:storageos:Host:71c0513f-38fe-45d8-a07e-55856965f53a:</id>
  <id>urn:storageos:Host:8bb9081b-94cf-4694-8952-f4f8f6739c1d:</id>
  <id>urn:storageos:Host:cc7d6de2-05cc-485a-b9ba-7196ddf060c1:</id>
  <id>urn:storageos:Host:d683a9bf-358b-40ad-89e3-db22acef1e50:</id>
  <id>urn:storageos:Host:ed2b0a9a-f3a3-4d4d-9e9d-3b6449513995:</id>
  <id>urn:storageos:Host:f06a2acc-2d9f-4105-8804-a0f4019bc6e5:</id>
  <id>urn:storageos:Host:f6ad7311-10eb-4f3b-8b00-7becff73b84d:</id>
</ids>
```

4. Choose a host from the list, and retrieve the IP interfaces for that host.**Request**

```
GET /compute/hosts/{Host_URN}/ip-interfaces
```

Response

```
<ip_interfaces>
  <ip_interface>
    <id>urn:storageos:IpInterface:4f518e6e-8e0d-4a4c-84ee-95a683f923f1:</id>
    <link href="/compute/ip-interfaces/urn:storageos:IpInterface:4f518e6e-8e0d-4a4c-84ee-95a683f923f1:" rel="self"/>
    <name>10.247.71.11</name>
  </ip_interface>
</ip_interfaces>
```

Choose an IP Interface. (In the above example, there is only 1.)

5. Call the following API to get detailed information about the interface.**Request**

```
GET /compute/ip-interfaces/{IpInterface_urn}
```

Response

```
<ip_interface>
  <creation_time>1382128965902</creation_time>
  <id>urn:storageos:IpInterface:
4f518e6e-8e0d-4a4c-84ee-95a683f923f1:</id>
  <inactive>false</inactive>
  <link href="/compute/ip-interfaces/urn:storageos:IpInterface:
4f518e6e-8e0d-4a4c-84ee-95a683f923f1:" rel="self"/>
  <name>7/ipv4</name>
  <tags/>
  <host>
    <id>urn:storageos:Host:f6ad7311-10eb-4f3b-8b00-7becff73b84d:</id>
    <link href="/compute/hosts/
urn:storageos:Host:f6ad7311-10eb-4f3b-8b00-7becff73b84d:" rel="self"/>
  </host>
  <protocol>IPV4</protocol>
  <ip_address>10.247.71.11</ip_address>
  <netmask>255.255.255.0</netmask>
</ip_interface>
```

The `<ip_address>` field has information you need to add a host endpoint to the IP network.

6. Get a list of storage systems.

Request

```
GET /vdc/storage-systems
```

Response

```
<storage_systems>
  <storage_system>
    <id>urn:storageos:StorageSystem:ab79974d-2097-4666-
bebe-25a299d8211b:</id>
    <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:ab79974d-2097-4666-bebe-25a299d8211b:" rel="self"/>
    <name>SYMMETRIX+999024668923</name>
  </storage_system>
</storage_systems>
```

7. Get a list of storage ports for that storage system.

Request

```
GET /vdc/storage-systems/{StorageSystem_URN}/storage-ports
```

Response

```
<storage_ports>
  <storage_port>
    <id>urn:storageos:StoragePort:00c2a754-b67a-4b1c-
b7e2-8e0b583e408c:</id>
    <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:d53ee95e-ee7f-4f8a-bd1c-923bd6ec840a:/storage-ports/urn:storageos:StoragePort:00c2a754-b67a-4b1c-b7e2-8e0b583e408c:" rel="self"/>
    <name>SYMMETRIX+000195700932+PORT+50:00:09:73:00:0E:91:5D</name>
  </storage_port>
  <storage_port>
    <id>urn:storageos:StoragePort:241b767f-afd7-4c4c-8136-
b1d59dd61ea1:</id>
    <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:d53ee95e-ee7f-4f8a-bd1c-923bd6ec840a:/storage-ports/urn:storageos:StoragePort:241b767f-afd7-4c4c-8136-b1d59dd61ea1:" rel="self"/>
    <name>SYMMETRIX+000195700932+PORT+ign.1992-04.com.emc:50000973000e91e0</name>
```

```

        </storage_port>
    </storage_ports>

```

8. Get information for a storage port in the list.

Request

```
GET /vdc/storage-ports/{StoragePort_urn}
```

Response

```

<storage_port>
    <creation_time>1384185364056</creation_time>
    <id>urn:storageos:StoragePort:00c2a754-b67a-4b1c-
b7e2-8e0b583e408c:</id>
    <inactive>false</inactive>
    <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:d53ee95e-ee7f-4f8a-bd1c-923bd6ec840a:/"
storage-ports/urn:storageos:StoragePort:00c2a754-b67a-4b1c-
b7e2-8e0b583e408c:" rel="self"/>
    <name>SYMMETRIX+000195700932+PORT+50:00:09:73:00:0E:91:5D</name>
    <tags/>
    <native_guid>SYMMETRIX+000195700932+PORT+50:00:09:73:00:0E:
91:5D</native_guid>
    <operational_status>OK</operational_status>
    <port_group>FA-8F</port_group>
    <port_name>FA-8F:1</port_name>
    <port_network_id>50:00:09:73:00:0E:91:5D</port_network_id>
    <port_speed_gbps>8</port_speed_gbps>
    <port_type>frontend</port_type>
    <registration_status>REGISTERED</registration_status>
    <storage_system>
        <id>urn:storageos:StorageSystem:d53ee95e-ee7f-4f8a-
bd1c-923bd6ec840a:</id>
        <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:d53ee95e-ee7f-4f8a-bd1c-923bd6ec840a:"
rel="self"/>
        <storage_system>
            <transport_type>FC</transport_type>
        </storage_system>
    </storage_port>

```

The **<port_network_id>** holds the information for the endpoint you want to add to the IP network.

9. Add the host and array ports to the IP network.

Request

```
PUT /vdc/networks/{id}/endpoints
```

```

<network_endpoints>
    <endpoints>
        <endpoint>10.247.71.11</endpoint>
        <endpoint>50:00:09:73:00:0E:91:5D</endpoint>
        <!--...more "endpoint" elements...-->
    </endpoints>
    <op>add</op>
</network_endpoints>

```

The network id is the one you generated in step 2.

10. Update the network with the ID of the virtual array to which you want to assign the virtual array.

Request

```
PUT /vdc/networks/{network_urn}
```

```

<network_update>
    <name>NewIPNetwork</name>
    <varrays>
        <varray>pauls_virtual_array</varray>

```

```
</varrays>
</network_update>
```

API review: storage systems

ViPR supports the discovery of many kinds of storage arrays. The addition and discovery of storage arrays through the API varies according to the type of array you are discovering.

ViPR can discover the following storage arrays.

- ◆ VNX File
- ◆ VNX Block
- ◆ VMAX
- ◆ Isilon
- ◆ NetApp
- ◆ VPLEX

The following table shows a set of important REST API calls that add, discover, and return information on storage systems.

ViPR REST API Call	Description
POST /vdc/storage-systems	Create a new storage system.
POST /vdc/storage-systems/discover	Discover all storage systems registered with ViPR.
POST /vdc/storage-systems/{id}/discover	Discover a specific storage system.
PUT /vdc/storage-systems/{id}	Modify the credentials of a storage system.
GET /vdc/storage-systems	Retrieve a list of storage systems.
POST /vdc/storage-systems/{id}/deactivate	Remove a storage system from ViPR.

The following table shows some calls used to manage storage pools and storage ports. Typically, storage pools and storage ports are automatically discovered when the storage system discovery is successful, but ViPR administrators may want to add or delete the storage pools or ports discovered. These calls allow you to do that.

ViPR REST API Call	Description
GET /vdc/storage-systems/{id}/storage-pools	Get information about the storage pools on a specific array.
GET /vdc/storage-systems/{id}/storage-pools/{poolId}	Get information about a specific storage pool on an array.
POST /vdc/storage-systems/{id}/storage-pools/{poolId}/register	Register a storage pool on a specific array. This would typically be called if the storage pool was specifically de-registered by a previous API call.
POST /vdc/storage-pools/{id}/deregister	De-register a storage pool from an array.
GET /vdc/storage-systems/{id}/storage-ports	Get information about the storage ports on a specific array.
GET /vdc/storage-systems/{id}/storage-ports/{portId}	Get information about a specific storage port on an array.

ViPR REST API Call	Description
POST /vdc/storage-systems/{id}/storage-ports/{portId}/register	Register a storage port on a specific array. This would typically be called if the storage port was specifically de-registered by a previous API call.
POST /vdc/storage-ports/{id}/deregister	De-register a storage port from an array.

ViPR also discovers the Fully Automated Storage Tiering (FAST) policies (called auto tier policies in ViPR) and the storage tiers associated with the physical storage pools on VPLEX, VMAX, and VNX block arrays.

ViPR REST API Call	Description
GET /vdc/storage-systems/{id}/auto-tier-policies	List storage system autotier policies

Tenant and project resources

In the ViPR virtual data center, there is a top level provider tenant (this would be a service provider or an enterprise IT department in a private cloud deployment). This provider tenant can contain one level of subtenants.

ViPR users reside in your corporate Active Directory or LDAP repository and are mapped to ViPR tenants through an authentication provider. Users are then assigned roles, which grant them privileges to create, read, update and delete resources in the ViPR environment. Users mapped to a tenant and assigned sufficient privileges can create one level of projects within their tenancy, and allocate the appropriate storage resources (that is, volumes, file systems, and objects) to each project.

In an enterprise environment, the provider tenant is used and no additional subtenants are needed. So for example, the ABC enterprise would be the provider tenant and would create one level with many projects within its tenancy. (There are no subtenants under the ABC enterprise provider tenant.)

In a service provider environment, the provider tenant (that is, the service provider) can create one level of tenants. For example, suppose the Acme Service Provider is the provider tenant and this service provider has two customers: the ABC and XYZ companies. The service provider could then create two subtenants within ViPR named ABC and XYZ. These tenants represent organizations operating within the ViPR environment.

Tenants are created in ViPR for the purpose of strict security isolation. Typically, each tenant is configured with its own Authentication Provider. The users in the ABC and XYZ tenants see a URL namespace that is limited to the information associated with their tenancy. There is no provision to create the tenants from the ViPR Administration and Self Service UI. The ViPR REST API can be used to create tenants if needed.

As shown in the resource model, users can create their own projects within their tenant. Projects are used to create enterprise departments or organizations. A user in the ABC tenant could create projects for HR, Sales, and Payroll; these projects are a way of grouping resources mapped to specific applications.

Users can provision multiple storage resources from different data services (block, file, and object stores) to their projects (that is, storage volumes, file systems, or objects such as files or images). Resources from one project can be shared between users under the same tenant.

CHAPTER 5

Setting Up a Multi-tenant Environment

This chapter contains the following topics.

- ◆ [Prerequisites for creating multiple tenants](#)..... 52
- ◆ [Configuring multiple tenants with the REST API](#)..... 52

Prerequisites for creating multiple tenants

ViPR can be configured with multiple tenants. Each tenant has its own environment for creating and managing storage. Storage resources assigned to a tenant cannot be accessed by users from other tenants.

Before creating additional tenants, you must have performed the deployment and initial configuration of the ViPR controller described in the *EMC ViPR Installation and Configuration Guide* chapter entitled "Initial configuration of ViPR virtual appliance".

The initial configuration of ViPR sets up a single-tenant environment. Only the provider tenant is available by default. To set up a multi-tenant ViPR environment, you must use the API procedure described in this chapter. (An analogous procedure using the ViPR CLI is also available.) You cannot create subtenants under the provider tenant from the ViPR User Interface.

Physical assets, such as storage systems and fabrics, added to ViPR are available to the virtual data center and can be assigned to any tenant. Similarly, virtual arrays and virtual pools created in ViPR are virtual data center assets and can be assigned to any tenant, including new tenants that you create. For example, the configuration of virtual arrays and virtual pools can be performed for the provider tenant from the UI. Those virtual arrays and virtual pools can then be assigned to new tenants.

ViPR role requirements

The root user for your ViPR vApp has all the role assignments you need to complete the multi-tenant setup.

Example format: XML

The ViPR Controller REST API can accept and deliver request and response data in either XML or JSON format.

The examples in this chapter show requests and responses in XML format.

Configuring multiple tenants with the REST API

This section shows how to configure multiple tenants with the ViPR API.

Before you begin

Complete the deployment and initial configuration steps in the *EMC ViPR Installation and Configuration Guide*.

Procedure

1. Authenticate with ViPR using an account that has Security Administrator and System Administrator roles. The root user has these roles and can be used.

How you authenticate depends on the HTTP client that you are using. If you are using a browser-based client, you could log in at the ViPR UI and the session cookie created will authenticate the HTTP client connection.

2. Ensure you have an authentication provider configured that will authenticate users in your domain.

Either:

- Create an authentication provider at the **Admin > Security > Authentication Providers** menu of the ViPR UI.
- Use the `/vdc/admin/authnproviders` API. For example:

Request

```
POST /vdc/admin/authnproviders
<authnprovider_create>
  <mode>ad</mode>
  <domains>
    <domain>yourco.com</domain>
    <domain>domain2.yourco.com</domain>
    <domain>domain3.yourco.com</domain>
  </domains>
  <name>multi-domain forest</name>
  <server_urls>
    <server_url>ldaps://MyLDAPServer.yourco.com:3269</server_url>
  </server_urls>
  <server_cert>my_server_certificate</server_cert>

  <manager_dn>CN=manager_bind,OU=Test1,OU=Test,DC=yourco,DC=com</manager_dn>
    <manager_password>Password</manager_password>
    <group_attribute>CN</group_attribute>
    <search_base>DC=yourco,DC=com</search_base>
    <search_filter>userPrincipalName=%u</search_filter>
    <search_attribute_key>userPrincipalName</search_attribute_key>
      <group_whitelist_values></group_whitelist_values>
      <search_scope>SUBTREE</search_scope>
    </search_attribute_key>
  </authnprovider_create>
```

3. Get the urn ID of the root provider tenant.

You must have the Tenant Administrator role to perform this operation.

Request

```
GET /tenant
```

Response

```
<tenant_info>
  <id>urn:storageos:TenantOrg:e5013f5e-41d7-4cf9-b1fd-4fecfad0c18c:</id>
    <name>Provider Tenant</name>
    <link href="/tenants/urn:storageos:TenantOrg:e5013f5e-41d7-4cf9-b1fd-4fecfad0c18c:" rel="self"/>
  </tenant_info>
```

The urn of the root provider tenant in this example is:

`urn:storageos:TenantOrg:e5013f5e-41d7-4cf9-b1fd-4fecfad0c18c:`

Use this urn as the parent when creating a new tenant in the following step.

4. Create a new tenant and map users to it through a domain that is included in the authentication provider.

Note

The set of LDAP users assigned to a subtenant is always a subset of the users mapped to the Provider Tenant.

In this example, the users in the domain `domain2.yourco.com` are mapped into the tenant called `EMC_tenant`. You must have the Tenant Administrator role for the parent tenant to perform this operation. The `{id}` variable is the URN of the provider tenant.

Request

```
POST /tenants/{id}/subtenants
<tenant_create>
  <name>EMC_tenant</name>
```

```

<user_mappings>
  <user_mapping>
    <domain>domain2.yourco.com</domain>
  </user_mapping>
</user_mappings>
</tenant_create>

```

You can control the users mapped into a tenant by specifying attributes. For example, if you only want users assigned to a specific department in AD to be mapped into the tenant, you can set key/value attributes. For example:

```

<user_mapping>
  <domain>domain2.yourco.com</domain>
  <attributes>
    <attribute>
      <key>department</key>
      <value>development</value>
    </attribute>
  </attributes>
</user_mapping>

```

Alternatively, you can map users into the tenant based on their AD group. The following user mapping maps members of the "lab users" group into the tenant:

```

<user_mapping>
  <domain>domain2.yourco.com</domain>
  <groups>
    <group>lab users</group>
  </groups>
</user_mapping>

```

You can include more than one `<user_mapping>` to enable users to be mapped from any of the specified mappings.

If you included more than one group in a `<user_mapping>`, the user must belong to all groups. In the example below, users must belong to both "lab users" and "lab administrators" groups to be mapped into the tenant.

```

<user_mapping>
  <domain>domain2.yourco.com</domain>
  <groups>
    <group>lab users</group>
    <group>lab administrators</group>
  </groups>
</user_mapping>

```

Response

```

<tenant>
  <creation_time>1378919846777</creation_time>
  <id>urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:</id>
  <inactive>false</inactive>
  <link href="/tenants/urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:" rel="self"/>
  <name>EMC tenant</name>
  <tags/>
  <parent_tenant>
    <id>urn:storageos:TenantOrg:e5013f5e-41d7-4cf9-
b1fd-4fecfad0c18c:</id>
    <link href="/tenants/
urn:storageos:TenantOrg:e5013f5e-41d7-4cf9-b1fd-4fecfad0c18c:" rel="self"/>
  </parent_tenant>
  <user_mappings>
    <user_mapping>
      <attributes/>
      <domain>domain2.yourco.com</domain>
      <groups/>
    </user_mapping>
  </user_mappings>
</tenant>

```

```
</user_mappings>
</tenant>
```

5. If you want to assign access to a virtual array to the newly created tenant you can use the following steps.

By default, the access control list (ACL) for a virtual array is wide open and all tenants have access. Once you assign a tenant to the ACL for a virtual array, only that tenant will have access unless you assign other tenants to the ACL.

- a. Get a list of virtual arrays.

Request

```
GET /vdc/varrays
```

Response

```
<varrays>
  <varray>
    <id>urn:storageos:VirtualArray:1b86bbe1-c939-49d3-
b0ae-027dc95b1ccc:</id>
    <link href="/vdc/varrays/urn:storageos:VirtualArray:
1b86bbe1-c939-49d3-b0ae-027dc95b1ccc:" rel="self"/>
    <name>VSA</name>
  </varray>
</varrays>
```

Use one of the virtual array IDs for the next step. This example shows the following ID:

```
<id>urn:storageos:VirtualArray:1b86bbe1-c939-49d3-
b0ae-027dc95b1ccc:</id>
```

- b. Assign the new tenant access to a virtual array by adding this tenant to the ACL for the virtual array.

You must be authenticated as a user with the System Administrator or Security Administrator role to perform this operation.

Request

```
PUT /vdc/varrays/urn:storageos:VirtualArray:1b86bbe1-c939-49d3-
b0ae-027dc95b1ccc:/acl
  <acl_assignment_changes>
    <add>
      <privilege>USE</privilege>
      <tenant>urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:</tenant>
    </add>
  </acl_assignment_changes>
```

Response

```
<acl_assignments>
  <acl_assignment>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:</tenant>
  </acl_assignment>
</acl_assignments>
```

6. If you want to assign access to a virtual pool to the new tenant you can use the following steps.

By default, the access control list (ACL) for a virtual pool is wide open and all tenants have access. Once you assign a tenant to the ACL for a virtual pool, only that tenant will have access unless you assign other tenants to the ACL.

- a. Get a list of virtual pools. In the example below, file virtual pools have been listed.

Request

```
GET /file/vpools
```

Response

```
<vpool_list>
  <virtualpool>
    <id>urn:storageos:VirtualPool:58406f8b-5a0e-41c0-
a91b-5a8c59ac3a02:</id>
    <link href="/file/vpools/urn:storageos:VirtualPool:
58406f8b-5a0e-41c0-a91b-5a8c59ac3a02:" rel="self"/>
    <name>vsp1</name>
    <vpool_type>file</vpool_type>
  </virtualpool>
</vpool_list>
```

- b. Retrieve the urn of a virtual pool and add the tenant to the ACL for that pool.

You must be authenticated as a user with the System Administrator or Security Administrator role to perform this operation.

Request

```
PUT /file/vpools/urn:storageos:VirtualPool:58406f8b-5a0e-41c0-
a91b-5a8c59ac3a02:/acl
<acl_assignment_changes>
  <add>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:</tenant>
  </add>
</acl_assignment_changes>
```

Response

```
<acl_assignments>
  <acl_assignment>
    <privilege>USE</privilege>
    <tenant>urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:</tenant>
  </acl_assignment>
</acl_assignments>
```

7. To perform any tenant-specific administration, you need to have a Tenant Administrator for the tenant. You can create a Tenant Administrator using the /tenants/{id}/role-assignments path, as shown below:

Request

```
PUT /tenants/urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:/role-assignments
```

```
<role_assignment_change>
<add>
<role>TENANT_ADMIN</role>
<subject_id>fjones@domain2.yourco.com</subject_id>
</add>
</role_assignment_change>
```

8. For users to provision file or block storage, or to access object storage, the user must be assigned to a project. To create projects for the tenant, you can use /tenants/{id}/projects. For example:

Request

```
POST /tenants/urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:/projects
```

```
<project_create>
  <name>marketing_project</name>
</project_create>
```

Response

```
<tenant_project>
  <id>urn:storageos:Project:
  60a3069e-74cc-4e79-9857-1c121ce1635a:</id>
  <link href="/projects/urn:storageos:Project:
  60a3069e-74cc-4e79-9857-1c121ce1635a:" rel="self"/>
  <name>marketing_project</name>
</tenant_project>
```

If you have assigned a user to the Tenant Administrator role for the tenant, they will automatically have access to the project.

You can use the `projects/{id}/acl` path to assign permissions to a user for the project. For example:

Request

```
PUT projects/ urn:storageos:Project:
60a3069e-74cc-4e79-9857-1c121ce1635a:/acl
<acl_assignment_changes>
  <add>
    <privilege>USE</privilege>
    <subject_id>bsmith@domain2.yourco.com</subject_id>
  </add>
</acl_assignment_changes>
```

9. If you want to use Data Services, you need to assign a namespace to the tenant and assign a default data services virtual pool. You can also assign a default project. You must be authenticated as a user with the System Administrator role to perform this operation.

Request

```
POST /object/namespaces/namespace
<namespace_create>
  <namespace>namespace1</namespace>
  <vdcs>
    <tenant>urn:storageos:TenantOrg:4edc456c-
c7f5-4c54-84b2-29715cc8f504:</tenant>
  </vdcs>
</namespace_create>
```

Response

```
<namespace>
  <id>namespace1</id>
  <inactive>false</inactive>
  <link href="/object/namespaces/namespace/namespace1"
rel="self"/>
  <tags/>
  <vdcs/>
</namespace>
```


CHAPTER 6

Common Operations

This chapter contains the following topics.

◆ Bulk operations	60
◆ Searching API resources	61
◆ Tagging API resources	62
◆ Tracking asynchronous operations	62
◆ Deactivating, or decommissioning, resources	63

Bulk operations

Each ViPR storage resource class has a pair of /bulk APIs - a GET and a POST call.

These calls do the following:

- ◆ GET {resource_URN}/bulk returns a list of all resource names (URNs) of the given resource class.
- ◆ POST {resource_URN}/bulk returns all available detail information for each resource in the given resource class.

The following GET example returns a list of storage system URNs in the ViPR virtual data center.

Request

```
GET <ViPR_VIP>/vdc/storage-systems/bulk
```

Response

```
<ids>
  <id>urn:storageos:StorageSystem:0e0f247c-7fde-41e2-
b704-15acd21ae311:</id>
  <id>urn:storageos:StorageSystem:164749e2-e5c5-4f05-
b315-1ff6ab2e496f:</id>
  <id>urn:storageos:StorageSystem:1bc24c32-
e3f0-41ca-925e-410187258236:</id>
</ids>
```

The following POST example using the same storage system resource class returns detailed information for each storage system URN listed in the POST request payload. The POST request payload is usually a subset of the resource URNs returned by the GET request. In the example below, three storage system URNs returned by the GET call are included in the payload for the PUT request. Only information pertaining to those resources is returned.

Request

```
POST <ViPR_VIP>/vdc/storage-systems/bulk
```

Request body

```
<ids>
  <id>urn:storageos:StorageSystem:0e0f247c-7fde-41e2-
b704-15acd21ae311:</id>
  <id>urn:storageos:StorageSystem:164749e2-e5c5-4f05-
b315-1ff6ab2e496f:</id>
  <id>urn:storageos:StorageSystem:1bc24c32-
e3f0-41ca-925e-410187258236:</id>
</ids>
```

Response

```
<bulk_storage_systems>
  <storage_system>
    <creation_time>1379347335171</creation_time>
    <id>urn:storageos:StorageSystem:164749e2-e5c5-4f05-
b315-1ff6ab2e496f:</id>
    <inactive>false</inactive>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
164749e2-e5c5-4f05-b315-1ff6ab2e496f:" rel="self"/>
    <name>SYMMETRIX+999316031302</name>
    <...>
  </storage_system>
  <storage_system>
    <creation_time>1379347335171</creation_time>
    <id>urn:storageos:StorageSystem:0e0f247c-7fde-41e2-
```

```

b704-15acd21ae311:</id>
  <inactive>false</inactive>
  <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
0e0f247c-7fde-41e2-b704-15acd21ae311:" rel="self"/>
    <name>CLARIION+APM12381623880</name>
    <...>
  </storage_system>
  <storage_system>
    <creation_time>1379347335171</creation_time>
    <id>urn:storageos:StorageSystem:1bc24c32-
e3f0-41ca-925e-410187258236:</id>
    <inactive>false</inactive>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
1bc24c32-e3f0-41ca-925e-410187258236:" rel="self"/>
    <name>SYMMETRIX+999000414197</name>
    <...>
  </storage_system>
</bulk_storage_systems>

```

Searching API resources

The REST API call GET <ViPR_VIP>:4443/<resource_URN>/search?<search_options> allows you to perform search API operations on ViPR resources by name, tag, project, and/or WWN.

<ViPR_VIP>/<resource_URN> is a base resource URN such as https://<ViPR_VIP>/file/filesystems.

Note

This example URL only returns resources that the user is authorized to see. The user must have an ACL privilege to the project in which the file systems reside in order to get a valid search result.

Search option	Required?	Description	Example
/search? name={namevalue}	Yes	Search for resources by name. This option is case-insensitive and matches name.	/block/volumes/ search?name=vnx (search for all volumes named vnx)
/search? tag={tagvalue}	No	Search for resources by tag. This option is case-insensitive and matches tag.	/block/volumes/ search?tag=export1 &name=vnx (search for all volumes tagged with export1 and named vnx)
/search? project={projectID}	No	Limit search results to resources in a specific project. Note: User must have access rights to the project.	/block/volumes/ search? name=VNX&project=urn :storageos:Project:X XX: (search for all volumes named vnx in the project with a project ID of urn:storageos:Project:XXX:)
/search? wwn={wwnvalue}	No	Search for volumes, exports, or networks by a specific wwn.	/block/volumes/ search? wwn=50:00:09:72:c0:0

Search option	Required?	Description	Example
			5:3d:1c (search for volumes with a WWN of 50:00:09:72:c0:05:3d:1c)

Tagging API resources

A tag is a key word or label that you can add to a resource to make it easier to find. You can tag a resource with words like *production*, *dev*, *provisioning*, *testgroup*, *VMAX*, *RAID-5*, and so on.

For example, you could search for all volumes with tags containing the word *VNX* by issuing the following GET request:

```
GET <ViPR_VIP>/block/volumes/search?tag=VNX.xml
```

This URL returns a list of volumes tagged with *VNX* in XML format.

```
<volumes>
  <volume>
    <id>urn:storageos:Volume:60d64226-7c11-49b4-8233-90fc352eedb6:</id>
    <link href="/block/volumes/urn:storageos:Volume:60d64226-7c11-49b4-8233-90fc352eedb6:" rel="self" />
    <tag>VNX30Sept</tag>
  </volume>
</volumes>
```

URL	Description
GET <ViPR_VIP>/{resource_URL}/tags	Get tags for a specific storage resource.
POST <ViPR_VIP>/{resource_URL}/tags	Add or remove tag(s) to/from a specific storage resource.

Tracking asynchronous operations

A number of APIs operate asynchronously. Asynchronous APIs return a task (or list of tasks) with a self link to the created resource and operation that uniquely identifies this request.

This task can be polled to check if the operation has succeeded, failed or is in progress. If the request was successful and the asynchronous operation is being processed, the service returns status code 202 (Accepted). Note that this status code does not indicate whether the operation itself has been processed successfully, but only that the request has been received by the service.

Task response contains status information. This information is set to pending initially. Polling on task URI can be done until the returned state is error or ready. If the operation fails, the response includes additional error information. If there are multiple tasks, each task URI can be polled to determine that particular operation status.

Note

An asynchronous operation takes an unspecified amount of time to complete. Your application should poll every 10 seconds to see when the operation is complete.

`GET {resource_URL}/tasks` returns a list of tasks created by asynchronous operations. The `resource_URL` is the path to the resource. For example: `GET /block/volume/tasks`.

`GET {resource_URL}/tasks/{op_id}` returns detailed information about a task. The `resource_URL` is the path to the resource. The `op_id` is the operation ID returned from an asynchronous API. For example: `GET /block/volume/tasks/{op_id}`

The following example shows how to track asynchronous operation status.

Request:

```
POST /block/volumes
```

Request body

```
<volume_create>
  <name>Sample_Volume</name>
  <size>1073741824</size>
  <count>1</count>
  <vpool>{vpool_id}</vpool>
  <varray>{varray_id}</varray>
  <project>{project_id}</project>
</volume_create>
```

Response

202

Response body

```
<tasks>
  <task>
    <resource>
      <name>Sample_Volume</name>
      <id>urn:storageos:Volume:5ba5b8d8-a0ca-4827-84f9-c1fef57733f5:</id>
      <link rel="self" href="/block/volumes/urn:storageos:Volume:5ba5b8d8-a0ca-4827-84f9-c1fef57733f5:" />
    </resource>
    <state>pending</state>
    <start_time>1379398608574</start_time>
    <op_id>265cf333-76a1-4129-903e-fac63f9b4adc</op_id>
    <link rel="self" href="/block/volumes/urn:storageos:Volume:5ba5b8d8-a0ca-4827-84f9-c1fef57733f5:/tasks/265cf333-76a1-4129-903e-fac63f9b4adc" />
  </task>
</tasks>
```

Deactivating, or decommissioning, resources

ViPR resources are not deleted, but are marked as Inactive. Inactive resources are periodically collected and removed permanently from the database by an internal process called *garbage collection*.

ViPR allows ten minutes to settle all current activities with a resource before it is eligible for garbage collection.

The garbage collection process collects all resources that have been inactive for more than 10 minutes and removes them from the database if no related resources exist in the database. Thus, a resource is mostly likely to be garbage collected within 20 minutes after it was made inactive.

Most (but not all) API requests return resources that might be inactive. For example, bulk search APIs do not filter out inactive resources. Inactive resources are not appropriate for any operations as they can disappear from the database at any moment. It is the responsibility of the API developer to determine if a resource is active before engaging it in further processing. You can check the state of the resource by checking the **inactive** property of that resource.

CHAPTER 7

Setting Up a Virtual Data Center

This chapter contains the following topics.

◆ Setting up the virtual data center	66
◆ Example format: JSON	66
◆ Adding a storage system	66
◆ Registering network systems	67
◆ Registering a RecoverPoint protection system	68
◆ Host	70
◆ Registering hosts and assigning initiators and IP interfaces	70
◆ Setting up the virtual array	72
◆ Setting up the virtual pool	83
◆ Creating a project for a tenant	90

Setting up the virtual data center

The virtual data center represents the ViPR storage control point in a physical data center. Storage resources, such as volumes, file systems, and objects are provisioned into the virtual data center.

Before you begin

- ◆ An authentication provider has been added to ViPR as described in the chapter [Setting Up a Single Tenant Environment on page 19](#)
- ◆ Roles have been assigned to users.

Procedure

1. Log into ViPR with credentials that match the roles required for the REST requests used to setup the virtual data center.
The root user for your ViPR vApp has all the role assignments you need.
2. Register and discover the physical storage systems.
3. Register the network systems.
4. Register the RecoverPoint protection system, if applicable.
5. Register Host and assign initiators.
6. Set up the virtual array.
7. Set up the virtual pool.
8. Create a project for a tenant.

Example format: JSON

The ViPR Controller REST API can accept and deliver request and response data in either XML or JSON format.

The examples in this chapter show requests and responses in JSON format.

Adding a storage system

The first step in configuring ViPR after deploying the virtual appliance is to add physical storage systems into the ViPR virtual data center.

Adding storage systems to ViPR can only be performed by a user with a System Administrator role. When storage systems are added, ViPR automatically discovers the storage pools and ports on each storage system and adds them into the ViPR virtual data center.

ViPR supports the following types of storage systems:

- ◆ EMC VPLEX, VMAX, VNX, Isilon, and NetApp storage systems for block and file storage provisioning.
- ◆ VNX file, Isilon, and NetApp filers for object storage.

Add your storage system, using one of the following procedures:

- ◆ [Registering and discovering physical storage systems in VNX Block and VMAX on page 138](#)
- ◆ [Registering and discovering an Isilon storage system on page 140](#)

- ◆ [Registering and discovering a NetApp storage system on page 142](#)
- ◆ [Registering and discovering a VPLEX on page 139](#)

Registering network systems

A network system describes and manages the SAN. This can represent an SSH connection to a Cisco MDS or Nexus switch, or an SMI-S connection to the Brocade Network Advisor. Once a network system is registered, the SAN groups, zones, and initiators of the network system can be discovered.

Before you begin

You need the following information:

- ◆ The IP address of the network system for SSH communication and only applicable for network systems of type mds.
- ◆ The username and password used for SSH login to the network system.
- ◆ The TCP port number to be used for SSH communication.
- ◆ The TCP port number used for communication with the SMI-S provider.

Procedure

1. Register a network system that describes and manages the SAN by sending a POST /vdc/network-systems request.

The request returns a task whose URI can be queried to determine the status of the task such as completion, any failures, time of task request, and time of task completion

In this example, a network system with an IP address of 10.247.99.250 is being registered.

Request

```
POST https://<ViPR_VIP>:4443/vdc/network-systems
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}

{
  "network_system_create": {
    "name": "Brocade1",
    "ip_address": "192.168.0.100",
    "port_number": "5989",
    "smis_provider_ip": "192.168.0.0",
    "smis_user_name": "Administrator",
    "smis_password": "password",
    "smis_use_ssl": "true",
    "smis_port_number": "5989",
    "system_type": "brocade"
  }
}
```

Response

```
HTTP 202 Accepted
{
  "task": {
    "op_id": "e97d0f47-bc42-4095-830a-b8c0fa4df71e",
    "resource": {
      "id": "urn:storageos:NetworkSystem:241cd3d2-825a-44f1-86de-2c0d23417b1e",
      "link": {
        "-href": "/vdc/network-systems/urn:storageos:NetworkSystem:241cd3d2-825a-44f1-86de-2c0d23417b1e"
      }
    }
  }
}
```

```

        "241cd3d2-825a-44f1-86de-2c0d23417b1e:",
        "-rel": "self"
    },
    "name": "Brocade1"
},
"link": {
    "-href": "/vdc/network-systems/urn:storageos:NetworkSystem:
241cd3d2-825a-44f1-86de-2c0d23417b1e:/tasks/e97d0f47-
bc42-4095-830a-b8c0fa4df71e",
    "-rel": "self"
},
"start_time": "1386950209819",
"state": "pending"
}
}
}

```

2. Query the discover network system task URI until the `message` attribute of the task is `Operation completed successfully`.

Request

```

GET https://<ViPR_VIP>:4443/vdc/network-systems/
urn:storageos:NetworkSystem:241cd3d2-825a-44f1-86de-2c0d23417b1e:
tasks/e97d0f47-bc42-4095-830a-b8c0fa4df71e
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}

```

Response

```

HTTP 200 OK
{
    "task": {
        "end_time": "1386950212551",
        "message": "Operation completed successfully",
        "op_id": "e97d0f47-bc42-4095-830a-b8c0fa4df71e",
        "resource": {
            "id": "urn:storageos:NetworkSystem:
241cd3d2-825a-44f1-86de-2c0d23417b1e:",
            "link": {
                "-href": "/vdc/network-systems/urn:storageos:NetworkSystem:
241cd3d2-825a-44f1-86de-2c0d23417b1e:",
                "-rel": "self"
            },
            "name": "Brocade1"
        },
        "link": {
            "-href": "/vdc/network-systems/urn:storageos:NetworkSystem:
241cd3d2-825a-44f1-86de-2c0d23417b1e:/tasks/e97d0f47-
bc42-4095-830a-b8c0fa4df71e",
            "-rel": "self"
        },
        "start_time": "1386950209819",
        "state": "ready"
    }
}

```

Registering a RecoverPoint protection system

Register a RecoverPoint protection system using POST /vdc/protection-systems. This step only applies if you have a RecoverPoint appliance.

Before you begin

You need the following information:

- ◆ The fully qualified domain name or IP address of the host.

- ◆ The username and password for connecting to the protection system device management port. The credentials must be for an account that has the RecoverPoint admin role to access the RecoverPoint site.
- ◆ The management port number of the protection system device. The default port to communicate with RecoverPoint is 7225.

Procedure

1. Register the protection system by sending a POST <ViPR_VIP>:4443/vdc/protection-systems request.

The request returns a task whose URI can be queried to determine the status of the task such as completion, any failures, time of task request, and time of task completion.

Request

```
POST https://<ViPR_VIP>:4443/vdc/protection-systems
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}

{
  "ip_address": "192.168.0.10",
  "name": "RP_DEV_MET",
  "password": "admin",
  "port_number": "7225",
  "system_type": "rp",
  "user_name": "admin"
}
```

Response

```
202 Accepted
{
  "task": {
    "op_id": "f39b188a-472f-4e2c-8079-01cf0f0a48b2",
    "resource": {
      "id": "urn:storageos:ProtectionSystem:70f63a68-5d69-4fe4-9934-847f23a9db5a:",
      "link": {
        "-href": "/vdc/protection-systems/urn:storageos:ProtectionSystem:70f63a68-5d69-4fe4-9934-847f23a9db5a:",
        "-rel": "self"
      },
      "name": "RP_DEV_MET"
    },
    "link": {
      "-href": "/vdc/protection-systems/urn:storageos:ProtectionSystem:70f63a68-5d69-4fe4-9934-847f23a9db5a:/tasks/f39b188a-472f-4e2c-8079-01cf0f0a48b2",
      "-rel": "self"
    },
    "start_time": "1386302068992",
    "state": "pending"
  }
}
```

2. Repeat the query of the protection system registration task, using the task URI from the response body of the POST request, until the message attribute of the task is Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/vdc/protection-systems/urn:storageos:ProtectionSystem:
```

```

70f63a68-5d69-4fe4-9934-847f23a9db5a:/tasks/
f39b188a-472f-4e2c-8079-01cf0f0a48b2
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}

Response
200 OK
{
    "task": {
        "end_time": "1386302069431",
        "message": "Operation completed successfully",
        "op_id": "f39b188a-472f-4e2c-8079-01cf0f0a48b2",
        "resource": {
            "id": "urn:storageos:ProtectionSystem:
70f63a68-5d69-4fe4-9934-847f23a9db5a:",
            "link": {
                "-href": "/vdc/protection-systems/
urn:storageos:ProtectionSystem:
70f63a68-5d69-4fe4-9934-847f23a9db5a:",
                "-rel": "self"
            },
            "name": "RP_DEV_MET"
        },
        "link": {
            "-href": "/vdc/protection-systems/
urn:storageos:ProtectionSystem:
70f63a68-5d69-4fe4-9934-847f23a9db5a:/tasks/
f39b188a-472f-4e2c-8079-01cf0f0a48b2",
            "-rel": "self"
        },
        "start_time": "1386302068992",
        "state": "error"
    }
}

```

Host

Hosts are computers that use software including Windows, Linux, and VMware for their operating system. In ViPR, hosts are tenant resources like volumes, file systems, and buckets. Unlike those resources, however, hosts are imported and discovered rather than provisioned by ViPR.

Hosts must be imported into ViPR by the Tenant Administrator before storage may be exported and attached to them. By default, hosts are not assigned to a project which means only the Tenant Administrator may see them and export/attach storage to them. If further delegation is required, the Tenant Administrator may assign a host to a project. Anyone who has privileges to manage resources in that project may then see and export/attach storage to that host.

Hosts are not explicitly associated with virtual arrays. The host-to-virtual array association is implied based on network connectivity.

Registering hosts and assigning initiators and IP interfaces

To manage hosts with ViPR, you need to register (add) those hosts with ViPR and then discover the hosts. Registering and discovering a Windows or SuSE LINUX 11 host automatically discovers the IPs and the initiator ports of the host. But for non-Windows or non-SuSE LINUX 11 hosts, you need to register the initiators with the host after registering the host itself.

Before you begin

You need the following information:

- ◆ A valid authentication token.
- ◆ The OS version, name, credentials, and port number of the host being registered.
- ◆ Whether SSL will be used to communicate with the host.
- ◆ The node address of an initiator.
- ◆ The port address of an initiator.

In this example, a Windows host is registered and instead of using the discovery to automatically discover the initiator ports, an initiator is manually registered with the host. [Adding and discovering a Windows host on page 127](#), [Adding and discovering a LINUX host on page 130](#), and [Adding and discovering a vCenter Server on page 131](#) provide the steps to register a Windows, Linux, and vCenter host, respectively.

Procedure

1. Get the URN of your tenant.

Request

```
GET https://<ViPR_VIP>:4443/tenant
```

2. Use the tenant URN to register the host with ViPR.

Request

```
PUT https://<ViPR_VIP>:4443/tenants/urn:storageos:TenantOrg:757053b7-b952-41b9-83e1-44204b67a368:/hosts

{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}

{
  "host_name": "myhost",
  "name": "myhost",
  "os_version": "1.0",
  "password": "password",
  "port_number": "8111",
  "type": "Windows",
  "use_ssl": null,
  "user_name": "user"
}
```

Response

HTTP 202 Accepted

```
{
  "link": {
    "href": "/compute/hosts/{Host_URN}",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "674689c2-74fd-45f8-a13a-8ee48525ffe3",
  "resource": {
    "id": "urn:storageos:Host:037aeb4e-c6e4-4fa3-8a67-6db94a93a6c5:",
    "link": {
      "href": "/compute/hosts/urn:storageos:Host:037aeb4e-c6e4-4fa3-8a67-6db94a93a6c5:",
      "rel": "self"
    }
  }
}
```

```

        },
        "name": "myhost"
    },
    "start_time": 1379202603051,
    "state": "ready"
}

```

3. Register an initiator associated with the host by calling the following POST.

Request

```
POST https://<ViPR_VIP>:4443/compute/hosts/{Host_URN}/initiators

{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}

{
    "initiator_node": "20:99:59:52:95:00:92:BE",
    "initiator_port": "10:13:27:65:60:38:68:BE",
    "protocol": "FC"
}
```

Response

HTTP 200 OK

```
{
    "creation_time": 1379202603661,
    "host": {
        "id": "urn:storageos:Host:037aeb4e-c6e4-4fa3-8a67-6db94a93a6c5:",
        "link": {
            "href": "/compute/hosts/urn:storageos:Host:037aeb4e-c6e4-4fa3-8a67-6db94a93a6c5:",
            "rel": "self"
        }
    },
    "hostname": "myhost",
    "id": "urn:storageos:Initiator:07b2e71d-cb4c-49c9-94fe-1feab7878d35:",
    "inactive": false,
    "initiator_node": "20:99:59:52:95:00:92:BE",
    "initiator_port": "10:13:27:65:60:38:68:BE",
    "link": {
        "href": "/compute/initiators/urn:storageos:Initiator:07b2e71d-cb4c-49c9-94fe-1feab7878d35:",
        "rel": "self"
    },
    "protocol": "FC",
    "tags": []
}
```

Setting up the virtual array

The virtual array is created and an ACL is assigned to it. A physical storage pool is then associated with the virtual array so it can be used for self-service provisioning.

Procedure

1. Create a virtual array by sending a POST request to /vdc/varrays.

The request returns the representation of the new virtual array.

Request

```
POST: https://<ViPR_VIP>:4443/vdc/varrays

{'Content-Type': 'application/json',
```

```
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}
```

```
{
    "auto_san_zoning": "true",
    "name": "Freemont"
}
```

Response

HTTP 200 OK

```
{
    "auto_san_zoning": true,
    "creation_time": 1379202577579,
    "id":
    "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
    "inactive": false,
    "link": {
        "href": "/vdc/varrays/
    urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
        "rel": "self"
    },
    "name": "Freemont",
    "tags": []
}
```

2. Create an ACL for the varray and assign the tenant user to the ACL by sending a PUT /vdc/varrays/{identifier}/acl request.

The request returns the representation of the new ACL.

Request

```
PUT https://<ViPR_VIP>:4443/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:/
acl
```

```
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}
```

```
{
    "add": [
        {
            "privilege": [
                "USE"
            ],
            "tenant": "urn:storageos:TenantOrg:757053b7-
b952-41b9-83e1-44204b67a368:"
        }
    ]
}
```

Response

HTTP 200 OK

```
{
    "acl": [
        {
            "privilege": [
                "USE"
            ],
            "tenant": "urn:storageos:TenantOrg:757053b7-
b952-41b9-83e1-44204b67a368:"
        }
    ]
}
```

3. Get the ID of the physical storage array, on which the storage pools were created, by sending a POST request to /vdc/storage-systems.

This request returns a list of all of the physical storage arrays.

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems
```

```
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}
```

Response

HTTP 200 OK

```
{
  "storage_system": [
    {
      "id": "urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
      "link": {
        "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
        "rel": "self"
      },
      "name": "CLARIION+APM00121500018"
    }
  ]
}
```

4. Use the storage array ID to get detailed information about the storage array by sending a GET/vdc/storage-systems/{identifier} request.

The request returns the representation of the storage array.

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:
```

```
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}
```

Response

HTTP 200 OK

```
{
  "active_provider_uri": {
    "id": "urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
    "link": {
      "href": "/vdc/smис-providers/urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
      "rel": "self"
    }
  },
  "associated_systems": [],
  "async_actions": [
    "CreateGroupReplica",
    "CreateElementReplica"
  ],
  "compatibility_status": "COMPATIBLE",
  "creation_time": 1379202543258,
  "export_masks": [],
  "firmware_version": "05.32.000.5.207",
  "id": "urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
  "inactive": false,
  "ip_address": "10.247.99.27",
  "job_discovery_status": "COMPLETE",
  "job_metering_status": "CREATED",
  "last_discovery_run_time": 1379202575496,
  "last_discovery_status_message": "Discovery completed successfully for Storage System: urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
  "last_metering_run_time": 0,
  "link": {
    "href": "/vdc/storage-systems/urn:storageos:StorageSystem:
```

```

56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
    "rel": "self"
},
"max_resources": -1,
"model": "Rack Mounted VNX5500",
"name": "CLARIION+APM00121500018",
"native_guid": "CLARIION+APM00121500018",
"next_discovery_run_time": 1379206145976,
"next_metering_run_time": 0,
"num_resources": 0,
"protocols": [],
"reachable": true,
"registration_status": "REGISTERED",
"secondary_ips": [],
"serial_number": "APM00121500018",
"smis_port_number": 5988,
"smis_provider_ip": "10.247.99.27",
"smis_providers": [
    {
        "id":
"urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
            "link": {
                "href": "/vdc/smис-providers/
urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
                    "rel": "self"
            }
        }
    ],
"smis_use_ssl": false,
"smis_user_name": "admin",
"supported_provisioning_type": "THIN_AND_THICK",
"system_type": "vnxblock",
"tags": []
}

```

5. Get the list of virtual arrays by sending a GET /vdc/varrays request.

```

GET https://<ViPR_VIP>:4443/vdc/varrays
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '',
'ACCEPT': 'application/json'}

```

Response

```

HTTP 200 OK
{
    "varray": [
        {
            "id":
"urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
                "link": {
                    "href": "/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
                        "rel": "self"
                },
            "name": "Freemont"
        }
    ]
}

```

6. Use the virtual array ID to get detailed information about the virtual array by sending a GET /vdc/varrays/{identifier} request.

```

GET https://<ViPR_VIP>:4443/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:

{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}

```

Response

```

HTTP 200 OK
{

```

```

        "auto_san_zoning": true,
        "creation_time": 1379202577579,
        "id":
      "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
        "inactive": false,
        "link": {
          "href": "/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
          "rel": "self"
        },
        "name": "Freemont",
        "tags": []
      }
    }
  
```

7. Get the list of storage pools on the physical storage array by sending a GET /vdc/storage-systems/{identifier}/storage-pools request.

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools
```

```
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '',
'ACCEPT': 'application/json'}
```

Response

```
HTTP 200 OK
{
  "storage_pool": [
    {
      "id": "urn:storageos:StoragePool:0d3c05e3-40f6-4044-89c6-5067e87d24c4:",
      "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:0d3c05e3-40f6-4044-89c6-5067e87d24c4:",
        "rel": "self"
      },
      "name": "CLARIION+APM00121500018+POOL+C+0008"
    },
    {
      "id": "urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
      "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
        "rel": "self"
      },
      "name": "CLARIION+APM00121500018+POOL+U+Pool 2"
    },
    {
      "id": "urn:storageos:StoragePool:7537a16d-0901-4ded-af7b-fbb9a9ec6ac9:",
      "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:7537a16d-0901-4ded-af7b-fbb9a9ec6ac9:",
        "rel": "self"
      },
      "name": "CLARIION+APM00121500018+POOL+C+0000"
    },
    {
      "id": "urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:",
      "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
```

```

storage-pools/urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-
a545a45e5e28:", {
    "rel": "self"
},
{
    "name": "CLARIION+APM00121500018+POOL+U+Pool 1"
},
{
    "id": "urn:storageos:StoragePool:
84f2d5f0-4e7c-4914-903c-c6abb7015229:", {
        "link": {
            "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:84f2d5f0-4e7c-4914-903c-c6abb7015229:", {
                "rel": "self"
            },
            "name": "CLARIION+APM00121500018+POOL+C+0002"
        },
        {
            "id": "urn:storageos:StoragePool:
9ae020ba-544b-4ce3-88b7-f1019cd7a9eb:", {
                "link": {
                    "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:9ae020ba-544b-4ce3-88b7-f1019cd7a9eb:", {
                        "rel": "self"
                    },
                    "name": "CLARIION+APM00121500018+POOL+C+0001"
                },
                {
                    "id": "urn:storageos:StoragePool:ad0dae57-d318-409c-a859-6c9e59a28251:", {
                        "link": {
                            "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:ad0dae57-d318-409c-a859-6c9e59a28251:", {
                                "rel": "self"
                            },
                            "name": "CLARIION+APM00121500018+POOL+U+Pool 0"
                        }
                    }
                }
            }
        }
    }
}

```

- Get the detailed information of the storage pool to add to the virtual array by sending a GET /vdc/storage-systems/{identifier}/storage-pools/{pool_id} request.

In this example the detailed information of two different storage pools is being obtained.

```

GET https://<ViPR_VIP>:4443/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:
0d3c05e3-40f6-4044-89c6-5067e87d24c4:
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}' ,
'ACCEPT': 'application/json'}

```

Response

```

HTTP 200 OK
{
    "assigned_varrays": [],
    "connected_varrays": [],
    "controller_params": [],
    "copy_types": [
        "UNSYNC_UNASSOC",
        "SYNC",
        "UNSYNC_ASSOC"
    ],
}

```

```

    "creation_time": 1379202572346,
    "drive_types": [
        "SAS"
    ],
    "free_gb": 0,
    "id": "urn:storageos:StoragePool:0d3c05e3-40f6-4044-89c6-5067e87d24c4:",
    "inactive": false,
    "link": {
        "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:0d3c05e3-40f6-4044-89c6-5067e87d24c4:",
        "rel": "self"
    },
    "max_pool_utilization_percentage": 75,
    "max_resources": -1,
    "maximum_thick_volume_size_gb": 0,
    "maximum_thin_volume_size_gb": 0,
    "minimum_thick_volume_size_gb": 0,
    "minimum_thin_volume_size_gb": 0,
    "name": "CLARIION+APM00121500018+POOL+C+0008",
    "native_guid": "CLARIION+APM00121500018+POOL+C+0008",
    "operational_status": "READY",
    "percent_subscribed": 101,
    "percent_used": 100,
    "pool_name": "0008",
    "pool_service_type": "block",
    "protocols": [
        "iSCSI",
        "FC"
    ],
    "raid_levels": [
        "RAID5"
    ],
    "registration_status": "REGISTERED",
    "storage_system": {
        "id": "urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
        "link": {
            "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
            "rel": "self"
        }
    },
    "subscribed_gb": 1606,
    "supported_resource_types": "THICK_ONLY",
    "tagged_varrays": [],
    "tags": [],
    "thin_volume_allocation_supported": false,
    "tier_utilization_percentages": [],
    "usable_gb": 1605,
    "used_gb": 1605
}

```

Send another request to get the details of a second storage pool.

```

GET https://<ViPR_VIP>:4443/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}' ,
'ACCEPT': 'application/json'}

```

Response

HTTP 200 OK

```
{
    "assigned_varrays": [],
    "connected_varrays": [],
    "controller_params": []
}
```

```

"copy_types": [
    "UNSYNC_UNASSOC",
    "SYNC",
    "UNSYNC_ASSOC"
],
"creation_time": 1379202572346,
"drive_types": [
    "SAS"
],
"free_gb": 294,
"id": "urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
"inactive": false,
"link": {
    "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
    "rel": "self"
},
"max_pool_utilization_percentage": 75,
"max_resources": -1,
"max_thin_pool_subscription_percentage": 300,
"maximum_thick_volume_size_gb": 285,
"maximum_thin_volume_size_gb": 16384,
"minimum_thick_volume_size_gb": 0,
"minimum_thin_volume_size_gb": 0,
"name": "CLARIION+APM00121500018+POOL+U+Pool 2",
"native_guid": "CLARIION+APM00121500018+POOL+U+Pool 2",
"operational_status": "READY",
"percent_subscribed": 108,
"percent_used": 45,
"pool_name": "Pool 2",
"pool_service_type": "block",
"protocols": [
    "iSCSI",
    "FC"
],
"raid_levels": [
    "RAID10"
],
"registration_status": "REGISTERED",
"storage_system": {
    "id": "urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
    "link": {
        "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
        "rel": "self"
    }
},
"subscribed_gb": 568,
"supported_resource_types": "THIN_AND_THICK",
"tagged_varrays": [],
"tags": [],
"thin_volume_preallocation_supported": false,
"tier_utilization_percentages": [
    {
        "name": "SATA",
        "value": "100"
    }
],
"usable_gb": 528,
"used_gb": 234
}
}

```

9. Assign a storage pool to the virtual array by sending a `PUT /vdc/storage-pools/{identifier}` request.

```

PUT https://<ViPR_VIP>:4443/vdc/storage-pools/
urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:
{'Content-Type': 'application/json',

```

```
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}
```

```
{
    "varray_assignment_changes": {
        "add": {
            "varrarrays": [
                "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:"
            ]
        }
    }
}
```

Response

```
HTTP 200 OK
{
    "assigned_varrarrays": [
        "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
        ],
        "connected_varrarrays": [],
        "controller_params": [],
        "copy_types": [
            "UNSYNC_UNASSOC",
            "SYNC",
            "UNSYNC_ASSOC"
        ],
        "creation_time": 1379202572346,
        "drive_types": [
            "SAS"
        ],
        "free_gb": 294,
        "id": "urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
        "inactive": false,
        "link": {
            "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
            "rel": "self"
        },
        "max_pool_utilization_percentage": 75,
        "max_resources": -1,
        "max_thin_pool_subscription_percentage": 300,
        "maximum_thick_volume_size_gb": 285,
        "maximum_thin_volume_size_gb": 16384,
        "minimum_thick_volume_size_gb": 0,
        "minimum_thin_volume_size_gb": 0,
        "name": "CLARIION+APM00121500018+POOL+U+Pool 2",
        "native_guid": "CLARIION+APM00121500018+POOL+U+Pool 2",
        "operational_status": "READY",
        "percent_subscribed": 108,
        "percent_used": 45,
        "pool_name": "Pool 2",
        "pool_service_type": "block",
        "protocols": [
            "iSCSI",
            "FC"
        ],
        "raid_levels": [
            "RAID10"
        ],
        "registration_status": "REGISTERED",
        "storage_system": {
            "id": "urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
            "link": {
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:",
                "rel": "self"
            }
        }
    ]
}
```

```

        "rel": "self"
    }
},
"subscribed_gb": 568,
"supported_resource_types": "THIN_AND_THICK",
"tagged_varrays": [
    {
        "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
        ],
        "tags": [],
        "thin_volume_preallocation_supported": false,
        "tier_utilization_percentages": [
            {
                "name": "SATA",
                "value": "100"
            }
        ],
        "usable_gb": 528,
        "used_gb": 234
    }
}

```

10. Repeat steps to add more storage pools to the virtual array.

11. The virtual array must be associated with a network. You can get which networks are available by sending a GET /vdc/networks request.

```
GET https://<ViPR_VIP>:4443/vdc/networks
```

```
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'}
```

Response

```

HTTP 200 OK
{
    "network": [
        {
            "id": "urn:storageos:Network:416c286c-3bbd-4923-840d-07945db2705b:",
            "link": {
                "href": "/vdc/networks/urn:storageos:Network:416c286c-3bbd-4923-840d-07945db2705b:",
                "rel": "self"
            },
            "name": "FABRIC_losam082-fabric"
        },
        {
            "id": "urn:storageos:Network:7bc25e2b-2221-48f9-9234-087096d71520:",
            "link": {
                "href": "/vdc/networks/urn:storageos:Network:7bc25e2b-2221-48f9-9234-087096d71520:",
                "rel": "self"
            },
            "name": "FABRIC_VPLEX_WAN_lglw9208/lglw9209"
        },
        {
            "id": "urn:storageos:Network:1253ec15-3ef9-4c73-a9e1-785904d17ff1:",
            "link": {
                "href": "/vdc/networks/urn:storageos:Network:1253ec15-3ef9-4c73-a9e1-785904d17ff1:",
                "rel": "self"
            },
            "name": "FABRIC_fake array fabric"
        },
        {
            "id": "urn:storageos:Network:6e7069ed-ba69-4342-90d4-ffdaed89aff8:",
            "link": {
                "href": "/vdc/networks/urn:storageos:Network:"
            }
        }
    ]
}

```

```

6e7069ed-ba69-4342-90d4-ffdaed89aff8:",
    "rel": "self"
},
{
    "name": "FABRIC_lglah043_11-10:00:00:05:1e:9b:49:9a"
},
{
    "id": "urn:storageos:Network:
22a50b48-1cb6-4948-96f3-57c546724333:",
    "link": {
        "href": "/vdc/networks/urn:storageos:Network:
22a50b48-1cb6-4948-96f3-57c546724333:",
        "rel": "self"
    },
    "name":
"urn:storageos:Network:ef704425-95b4-4f7a-9fce-91f4ddb322c6:",
        "link": {
            "href": "/vdc/networks/
urn:storageos:Network:ef704425-95b4-4f7a-9fce-91f4ddb322c6:",
            "rel": "self"
        },
        "name": "FABRIC_VPlex_Meta_Fid_20"
},
{
    "id": "urn:storageos:Network:d9abe55b-7b0c-43a5-
aaa6-5ddeb769f32f:",
        "link": {
            "href": "/vdc/networks/
urn:storageos:Network:d9abe55b-7b0c-43a5-aaa6-5ddeb769f32f:",
            "rel": "self"
        },
        "name": "FABRIC_VPlex_LGL6221_FID_40"
},
{
    "id": "urn:storageos:Network:c11db7cd-3f26-42a4-894c-
f476d06c7f36:",
        "link": {
            "href": "/vdc/networks/
urn:storageos:Network:c11db7cd-3f26-42a4-894c-f476d06c7f36:",
            "rel": "self"
        },
        "name": "FABRIC_vplex154nbr2"
},
{
    "id": "urn:storageos:Network:2973cd79-
b9c1-4a20-9466-95ad06d5f192:",
        "link": {
            "href": "/vdc/networks/urn:storageos:Network:
2973cd79-b9c1-4a20-9466-95ad06d5f192:",
            "rel": "self"
        },
        "name": "FABRIC_Vplex_WAN-10:00:00:27:f8:58:f6:bc"
}
]
}
}

```

12. Assign the VSAN to the virtual array (endpoints trimmed) by sending a `PUT /vdc/networks/{identifier}` request.

```

PUT https://<ViPR_VIP>:4443/vdc/networks/urn:storageos:Network:
416c286c-3bbd-4923-840d-07945db2705b:

{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}' ,
'ACCEPT': 'application/json'}

{
    "varrarrays": [

```

```

"urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:"
    ]
}
}

Response
HTTP 200 OK
{
    "creation_time": 1379202583924,
    "discovered": true,
    "endpoints": [
        "10:00:00:90:FA:13:83:97",
        "10:00:00:00:C9:47:25:9E"
    ],
    "endpoints_discovered": [
        {
            "name": "10:00:00:00:C9:47:25:9E",
            "value": "true"
        },
        {
            "name": "10:00:00:90:FA:13:83:97",
            "value": "true"
        }
    ],
    "fabric_id": "losam082-fabric",
    "id": "urn:storageos:Network:
416c286c-3bbd-4923-840d-07945db2705b:",
    "inactive": false,
    "link": {
        "href": "/vdc/networks/urn:storageos:Network:
416c286c-3bbd-4923-840d-07945db2705b:",
        "rel": "self"
    },
    "name": "FABRIC_losam082-fabric",
    "native_guid": "FC+BROCADE+10:00:00:27:F8:49:E8:7C",
    "network_systems": [
        "urn:storageos:NetworkSystem:518f41d0-ab22-4267-
b558-0d366e4ffc42:"
    ],
    "registration_status": "REGISTERED",
    "tags": [],
    "transport_type": "FC",
    "varray": {
        "id": "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
        "link": {
            "href": "/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
            "rel": "self"
        }
    }
}
}

```

Setting up the virtual pool

Physical storage pools on a physical storage system are automatically discovered and added into ViPR when the physical storage system is added into the ViPR virtual data center. A physical storage pool has to be associated to a virtual pool before it can be used for self-service provisioning.

Before you begin

You need the following information.

- ◆ The URNs of the virtual arrays to which the virtual pool is assigned.
- ◆ The URNs of the storage pools to be assigned to the virtual pool.

- ◆ The URN of the tenant to be assigned as the ACL to the virtual pool.
- ◆ A valid authentication token.

This example shows how to create a virtual pool, find storage pools that match your required properties, and then assign those storage pools to the virtual pool. The final step applies an ACL to the virtual pool.

Procedure

1. Create a virtual pool by sending a POST request to /block/vpools.

The representation of the new virtual pool is returned.

Request

```
POST https://<ViPR_VIP>:4443/block/vpools

{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}

{
    "description": "Basic Virtual Pool for Block Provisioning",
    "name": "basic_vpool",
    "num_paths": "1",
    "protection": {
        "snapshots": {
            "max_native_snapshots": "10"
        }
    },
    "protocols": [
        "FC"
    ],
    "provisioning_type": "Thin",
    "use_matched_pools": "true",
    "varrays": [
        "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:"
    ]
}
```

Response

```
HTTP 200 OK
{
    "assigned_storage_pools": [],
    "creation_time": 1379202604198,
    "description": "Basic Virtual Pool for Block Provisioning",
    "expandable": true,
    "id": "urn:storageos:VirtualPool:6cd9f843-1b41-4b9f-8eb1-a26797d0a268:",
    "inactive": false,
    "invalid_matched_pools": [],
    "link": {
        "href": "/block/vpools/urn:storageos:VirtualPool:6cd9f843-1b41-4b9f-8eb1-a26797d0a268:",
        "rel": "self"
    },
    "matched_storage_pools": [
        {
            "id": "urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:",
            "link": {
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:",
                "rel": "self"
            }
        },
        {
            "id": "urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:",
            "link": {
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:",
                "rel": "self"
            }
        }
}
```

```

        "id": "urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:",
        "link": {
            "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:",
            "rel": "self"
        }
    },
    {
        "id": "urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
        "link": {
            "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
            "rel": "self"
        }
    }
],
{
    "name": "basic_vpool",
    "num_paths": 1,
    "protection": {
        "continuous_copies": {
            "max_native_continuous_copies": 0
        },
        "snapshots": {
            "max_native_snapshots": 10
        }
    },
    "protocols": [
        "FC"
    ],
    "provisioning_type": "Thin",
    "raid_levels": [],
    "tags": [],
    "type": "block",
    "unique_auto_tier_policy_names": false,
    "use_matched_pools": true,
    "varrays": [
        {
            "id": "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
            "link": {
                "href": "/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
                "rel": "self"
            }
        }
    ]
}

```

2. Find the list of storage pools that match the properties of the virtual pool that was just created by sending a POST /block/vpools/matching-pools request.

Note

There is also a comparable /file/vpools/matching-pools for file virtual pools.

Request

```

POST https://<ViPR_VIP>:4443/block/vpools/matching-pools
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}' ,
'ACCEPT': 'application/json'}
{
    "num_paths": "1",

```

```

        "protection": {
            "snapshots": {
                "max_native_snapshots": "10"
            }
        },
        "protocols": [
            "FC"
        ],
        "provisioning_type": "Thin",
        "use_matched_pools": "true",
        "varrays": [
            "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:"
        ]
    }
}

```

Response

```

200 OK
{
    "storage_pool": [
        {
            "id": "urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:",
            "name": "SYMMETRIX+000198700420+POOL+TP+3R5-A",
            "link": {
                "rel": "self",
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:62d2e40d-ced0-4114-8bb7-8d333e7ef878:"
            }
        },
        {
            "id": "urn:storageos:StoragePool:9e551963-d596-48ae-b208-1b870126d195:",
            "name": "SYMMETRIX+000198700420+POOL+TP+M3",
            "link": {
                "rel": "self",
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:9e551963-d596-48ae-b208-1b870126d195:"
            }
        },
        {
            "id": "urn:storageos:StoragePool:ad0dae57-d318-409c-a859-6c9e59a28251:",
            "name": "SYMMETRIX+000198700420+POOL+TP+M2",
            "link": {
                "rel": "self",
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:ad0dae57-d318-409c-a859-6c9e59a28251:"
            }
        },
        {
            "id": "urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:",
            "name": "SYMMETRIX+000198700420+POOL+TP+data",
            "link": {
                "rel": "self",
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-a545a45e5e28:"
            }
        },
        {
            "id": "urn:storageos:StoragePool:38e2a0c2-89c9-4ec7-ad9b-29420e464012:",
            "name": "SYMMETRIX+000198700420+POOL+TP+t3",
            "link": {
                "rel": "self",
                "href": "/vdc/storage-systems/urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/urn:storageos:StoragePool:38e2a0c2-89c9-4ec7-ad9b-29420e464012:"
            }
        }
    ]
}

```

```

        "href": "/vdc/storage-systems/urn:storageos:StorageSystem:
56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/
urn:storageos:StoragePool:38e2a0c2-89c9-4ec7-ad9b-29420e464012:",
        }
    },
    {
        "id": "urn:storageos:StoragePool:dbcd42a-
b0a4-49b4-8a6e-204a09b8b1ef:",
        "name": "SYMMETRIX+000198700420+POOL+TP+3R5-B",
        "link": {
            "rel": "self",
            "href": "/vdc/storage-systems/urn:storageos:StorageSystem:
56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-pools/
urn:storageos:StoragePool:dbcd42a-b0a4-49b4-8a6e-204a09b8b1ef:",
        }
    }
]
}

```

3. Using the URN returned for the new virtual pool returned from the POST /block/vpools request, assign one or more of the storage pools that match the properties of the virtual pool by sending a PUT/block/vpools/{identifier}/assign-matched-pools request. The representation of the updated virtual pool is returned.

Request

```

PUT https://<ViPR_VIP>:4443/block/vpools/urn:storageos:VirtualPool:
6cd9f843-1b41-4b9f-8eb1-a26797d0a268:/assign-matched-pools

{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}' ,
'ACCEPT': 'application/json' }

{
    "assigned_pool_changes": {
        "add": {
            "storage_pool": [
                "urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
                "urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-
a545a45e5e28:",
                "urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:"
            ]
        }
    }
}

```

Response

```

HTTP 200 OK
{
    "assigned_storage_pools": [
        {
            "id": "urn:storageos:StoragePool:78caaf4a-673e-4580-
ae41-a545a45e5e28:",
            "link": {
                "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-
pools/urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-
a545a45e5e28:",
                "rel": "self"
            }
        },
        {
            "id": "urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:",
            "link": {
                "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/storage-
pools/urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:"
            }
        }
    ]
}

```

```

storage-pools/urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:",
    "rel": "self"
}
},
{
    "id": "urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
    "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
        "rel": "self"
    }
},
],
"creation_time": 1379202604198,
"description": "Basic Virtual Pool for Block Provisioning",
"expandable": true,
"id": "urn:storageos:VirtualPool:6cd9f843-1b41-4b9f-8eb1-
a26797d0a268:",
"inactive": false,
"invalid_matched_pools": [],
"link": {
    "href": "/block/vpools/urn:storageos:VirtualPool:
6cd9f843-1b41-4b9f-8eb1-a26797d0a268:",
    "rel": "self"
},
"matched_storage_pools": [
{
    "id": "urn:storageos:StoragePool:78caaf4a-673e-4580-
ae41-a545a45e5e28:",
    "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:78caaf4a-673e-4580-ae41-
a545a45e5e28:",
        "rel": "self"
    }
},
{
    "id": "urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:",
    "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:ad0dae57-d318-409c-
a859-6c9e59a28251:",
        "rel": "self"
    }
},
{
    "id": "urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
    "link": {
        "href": "/vdc/storage-systems/
urn:storageos:StorageSystem:56d8aa7e-45fe-4383-b49a-fec72e9927d3:/
storage-pools/urn:storageos:StoragePool:62d2e40d-
ced0-4114-8bb7-8d333e7ef878:",
        "rel": "self"
    }
}
],
"name": "basic_vpool",
"num_paths": 1,
"protection": {
    "continuous_copies": {
        "max_native_continuous_copies": 0
    },

```

```

        "snapshots": {
            "max_native_snapshots": 10
        }
    },
    "protocols": [
        "FC"
    ],
    "provisioning_type": "Thin",
    "raid_levels": [],
    "tags": [],
    "type": "block",
    "unique_auto_tier_policy_names": false,
    "use_matched_pools": true,
    "varrays": [
        {
            "id":
            "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
            "link": {
                "href": "/vdc/varrays/
urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
                "rel": "self"
            }
        }
    ]
}

```

4. Apply an ACL to the virtual pool by sending a PUT request to /block/vpools/{vpool_urn}/acl. The updated ACL is returned.

Request

```

PUT https://<ViPR_VIP>:4443/block/vpools/urn:storageos:VirtualPool:
6cd9f843-1b41-4b9f-8eb1-a26797d0a268:/acl

{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}

{
    "add": [
        {
            "privilege": [
                "USE"
            ],
            "tenant": "urn:storageos:TenantOrg:757053b7-
b952-41b9-83e1-44204b67a368:"
        }
    ]
}

```

Response

HTTP 200 OK

```

{
    "acl": [
        {
            "privilege": [
                "USE"
            ],
            "tenant": "urn:storageos:TenantOrg:757053b7-
b952-41b9-83e1-44204b67a368:"
        }
    ]
}

```

Creating a project for a tenant

A project is a container for ViPR resources that are accessible by users in a tenant. You can successfully create a project as root, or you can log in as a Tenant Administrator or Project Administrator to complete this task

Procedure

1. Create a project for a tenant by sending a `POST /tenants/{identifier}/projects` request.

Request

```
POST https://<ViPR_VIP>:4443/tenants/urn:storageos:TenantOrg:  
757053b7-b952-41b9-83e1-44204b67a368:/projects  
{  
    'Content-Type': 'application/json',  
    'X-SDS-AUTH-TOKEN': '{Token_Text}',  
    'ACCEPT': 'application/json'  
}  
  
{  
    "name": "myproject"  
}
```

Response

```
HTTP 200  
{  
    "id":  
        "urn:storageos:Project:cdb26784-9d70-413a-998c-8e6b7cd95719:",  
        "link": {  
            "href": "/projects/  
urn:storageos:Project:cdb26784-9d70-413a-998c-8e6b7cd95719:",  
            "rel": "self"  
        },  
        "name": "myproject"  
}
```

CHAPTER 8

Managing File Systems and Snapshots

This chapter contains the following topics.

◆ File system	92
◆ Creating a file system	92
◆ Expanding a file share	94
◆ Exporting a file system	96
◆ Unexporting a file share	97
◆ Deleting a file system export	99
◆ Deleting a file system	101
◆ Snapshot	102
◆ Creating a file system snapshot	102
◆ Exporting a file system snapshot	104
◆ Restoring a file system snapshot	106
◆ Unexporting a file system snapshot	107
◆ Deleting a file system snapshot	109

File system

A file system is a unit of file storage capacity that has been allocated by a user to a project.

File system provisioning operations are performed after the System Administrator has set up the ViPR virtual data center. File systems are provisioned within the context of a project by users who meet one of the following criteria:

- ◆ have a Tenant Administrator role.
- ◆ are a project owner.
- ◆ have an `ALL` access control list (ACL) permission on the project.

When a file system is created, the user must select the project and virtual array in which the file system will reside, and the virtual pool that will define the file system's storage performance characteristics. Once the file system is created, it can be exported to multiple hosts. In this way, a host cluster can be provisioned with shared file storage.

Unlike block volumes, there is no virtual array restriction when exporting file systems. File systems can be exported from any virtual array specified by the user. If no virtual array is selected for the file system export, the ViPR file store will pick the virtual array.

Creating a file system

When you create a file system you select the project and virtual array in which the file system will reside, and the virtual pool that will define the file system's storage performance characteristics. Once the file system is created, you can provision a host cluster with shared file storage by exporting the file system to multiple hosts.

Before you begin

You need this information:

- ◆ Name of the file system being created
- ◆ Size of the file system
- ◆ URN of the virtual array that contains the file system
- ◆ URN of the virtual pool supporting the file system

Procedure

1. Create a file system by sending a `POST /file/filesystems` request.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/file/filesystems?
project=urn:storageos:Project:d456ae4d-dala-4488-9e59-1b6e9dc59d78:
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN':{token_text},
    'ACCEPT': 'application/json'
}
{
    "name": "fs-cosnetappf-00:50:56:91:58:91",
    "size": "20MB",
    "varray": "urn:storageos:VirtualArray:69df596a-34b3-4d23-
aaea-1da667593b74:",
    "vpool": "urn:storageos:VirtualPool:0d5f110f-af0d-4af8-bd71-
```

```
a04635eba8be:"  
}
```

Response

```
202 Accepted  
{  
    "description": "Filesystem create",  
    "link": {  
        "href": "/file/filesystems/urn:storageos:FileShare:  
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/252396af-e4e5-4ee0-  
a6b5-6606aeeaaf55",  
        "rel": "self"  
    },  
  
    "op_id": "252396af-e4e5-4ee0-a6b5-6606aeeaaf55",  
    "resource": {  
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-  
c6efedc81fc3:",  
        "link": {  
            "href": "/file/filesystems/urn:storageos:FileShare:  
840d01a9-8836-4c53-a95a-c6efedc81fc3:",  
            "rel": "self"  
        },  
        "name": "fs-cosnetappf-00:50:56:91:58:91"  
    },  
    "start_time": 1379958410262,  
    "state": "pending"  
}
```

2. Send a GET request to the task URI that was returned by the POST request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/file/filesystems/  
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/  
tasks/252396af-e4e5-4ee0-a6b5-6606aeeaaf55  
{  
    'Content-Type': 'application/json',  
    'X-SDS-AUTH-TOKEN':{token_text},  
    'ACCEPT': 'application/json'  
}
```

Response

```
200 OK  
{  
    "description": "Filesystem create",  
    "end_time": 1379958413504,  
    "link": {  
        "href": "/file/filesystems/urn:storageos:FileShare:  
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/252396af-e4e5-4ee0-  
a6b5-6606aeeaaf55",  
        "rel": "self"  
    },  
  
    "message": "Operation completed successfully",  
    "op_id": "252396af-e4e5-4ee0-a6b5-6606aeeaaf55",  
  
    "resource": {  
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-  
c6efedc81fc3:",  
        "link": {  
            "href": "/file/filesystems/urn:storageos:FileShare:  
840d01a9-8836-4c53-a95a-c6efedc81fc3:",  
        }  
    }  
}
```

```

        "rel": "self"
    },
    "name": "fs-cosnetappf-00:50:56:91:58:91"
},
"start_time": 1379958410262,
"state": "ready"
}

```

Expanding a file share

A file share may be expanded by sending a POST to /file/filesystems/{id}/expand.

Before you begin

You need this information:

- ◆ The URN of the file system which is being expanded
- ◆ The size to which the file system is being expanded

Procedure

1. Expand a file system by sending a POST /file/filesystems/{id}/expand request.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```

POST https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
expand
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': {token_text},
'ACCEPT': 'application/json'}

{
    "new_size": "26214400"
}

```

Response

```

202 Accepted
{
    "link": {
        "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/2f108ca5-6cb0-44c0-
b4bb-f6aed1324520",
        "rel": "self"
    },
    "op_id": "2f108ca5-6cb0-44c0-b4bb-f6aed1324520",
    "resource": {
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-
c6efedc81fc3:",
        "link": {
            "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91"
    },
    "start_time": 1379958417856,
    "state": "pending"
}

```

```
{
  'op_id': '2f108ca5-6cb0-44c0-b4bb-f6aed1324520',
  'link': {
    'href': '/file/filesystems/urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/2f108ca5-6cb0-44c0-b4bb-f6aed1324520',
    'rel': 'self'
  },
  'state': 'pending',
  'resource': {
    'link':
      {'href': '/file/filesystems/urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:', 'rel': 'self'},
      'name': 'fs-cosnetappf-00:50:56:91:58:91',
      'id': 'urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:',
    },
    'start_time': 1379958417856
  }
}
```

- Send a GET request to the task URI that was returned by the POST request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
tasks/2f108ca5-6cb0-44c0-b4bb-f6aed1324520
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': {token_text},
'ACCEPT': 'application/json'}
```

Response

```
200 OK
{
  "end_time": 1379958418794,
  "link":
  {
    "href": "/file/filesystems/urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/2f108ca5-6cb0-44c0-b4bb-f6aed1324520",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "2f108ca5-6cb0-44c0-b4bb-f6aed1324520",
  "resource": {
    "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:",
    "link": {
      "href": "/file/filesystems/urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:",
      "rel": "self"
    },
    "name": "fs-cosnetappf-00:50:56:91:58:91"
  },
  "start_time": 1379958417856,
  "state": "ready"
}
```

Exporting a file system

Once a file system is created, it can be exported to multiple hosts. In this way, a host cluster can be provisioned with shared file storage.

Before you begin

You need this information:

- ◆ URN of the file system being exported
- ◆ Security type for the file export
- ◆ Permissions for the file export
- ◆ The user with root permissions
- ◆ The file system export protocol
- ◆ Endpoints

Procedure

1. Export the file system by sending a POST /file/filesystems/{id}exports request.

The request returns a task whose URI can be queried to determine the status of the task.

In this example, the `sub_directory` element is not included in the payload. Therefore, the entire file system is exported.

Request

```
POST https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
exports
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': {Token_text}
  'ACCEPT': 'application/json'
}
{
  "endpoints": [
    "www.ford.com",
    "www.gmc.com",
    "www.pontiac.com"
  ],
  "permissions": "rw",
  "protocol": "NFS",
  "root_user": "nobody",
  "type": "sys"
}
```

Response

```
202 Accepted
{
  "link": {
    "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/7770ff2a-433b-480c-
aef6-8c64e924339b",
    "rel": "self"
  },
  "op_id": "7770ff2a-433b-480c-aef6-8c64e924339b",
  "resource": {
    "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-
c6efedc81fc3:",
    "link": {
      "href": "/file/filesystems/urn:storageos:FileShare:
```

```

        "840d01a9-8836-4c53-a95a-c6efedc81fc3:",
        "rel": "self"
    },
    "name": "fs-cosnetappf-00:50:56:91:58:91"
},
"start_time": 1379958422465,
"state": "pending"
}

```

- Send a GET request to the task URI that was returned by the POST request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```

GET https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/7770ff2a-433b-480c-aef6-8c64e924339b
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_text}
    'ACCEPT': 'application/json'
}

```

Response

```

200 OK
{
    "end_time": 1379958422816,
    "link": {
        "href": "/file/filesystems/urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/7770ff2a-433b-480c-aef6-8c64e924339b",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "7770ff2a-433b-480c-aef6-8c64e924339b",
    "resource": {
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:",
        "link": {
            "href": "/file/filesystems/urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91"
    },
    "start_time": 1379958422465,
    "state": "ready"
}

```

Unexporting a file share

Unexport a file system by updating the list of endpoints of the file system export. A PUT request is sent to /file/filesystems/{id}/exports/{protocol}, {secType}, {perm}, {root_mapping}.

Before you begin

You need this information:

Requirement	Description
{id}	URN of the file system export being deleted
{protocol}	protocol of the file system export

Requirement	Description
{secType}	security type of the file system export
{perm}	permissions for the file system export
{root_mapping}	user with root permissions
add or remove	endpoint changes

Note

The permission, security, or root user mapping of an existing export may not be changed. In order to change one of these attributes, you must first delete the export and then create an export with the new value.

Procedure

1. Unexport the file system by removing the required endpoint.

In this example, www.ford.com is being removed as an endpoint. In addition, NFS is the protocol of the endpoint, sys is the security type of the export, rw is export permissions, and nobody is the root user for the export.

Request

```
PUT https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
exports/NFS,sys,rw,nobody
{
    "Content-Type": "application/json",
    "X-SDS-AUTH-TOKEN": {Token_Text},
    "ACCEPT": "application/json"
}

{
    "add": [
        "www.kia.com"
    ],
    "remove": [
        "www.ford.com"
    ]
}
```

Response

```
202 Accepted
{
    "link": {
        "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/
74465c92-4cfa-4a1c-88f0-b63a53fdbbe46",
        "rel": "self"
    },
    "op_id": "74465c92-4cfa-4a1c-88f0-b63a53fdbbe46",
    "resource": {
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-
c6efedc81fc3:",
        "link": {
            "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91"
    },
    "start_time": 1379958426981,
    "state": "pending"
}
```

- Send a `GET` request to the task URN that was returned by the `PUT` request to return the status of the task.

When the task completes successfully, the `message` attribute is set to `Operation completed successfully`.

Request

```
GET https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
tasks/74465c92-4cfa-4a1c-88f0-b63a53fdbe46
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

```
200 OK

{
    "end_time": 1379958427315,
    "link": {
        "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/
74465c92-4cfa-4a1c-88f0-b63a53fdbe46",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "74465c92-4cfa-4a1c-88f0-b63a53fdbe46",
    "resource": {
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-
c6efedc81fc3:",
        "link": {
            "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91"
    },
    "start_time": 1379958426981,
    "state": "ready"
}
```

Deleting a file system export

A file system export is deleted by sending a `DELETE` `file/filesystems/{id}/exports/{protocol},{secType},{perm},{root_mapping}` request.

Before you begin

You need this information:

Requirement	Description
{id}	URN of the file system
{protocol}	protocol of the file system export
{secType}	security type of the file system export
{perm}	permissions for the file system export
{root_mapping}	user with root permissions

Procedure

1. Delete the file system export. .

The request returns a task whose URI can be queried to determine the status of the task

Request

```
DELETE https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
exports/NFS,sys,rw,nobody
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_text}
    'ACCEPT': 'application/json'
}
```

Response

```
202 Accepted
{
    "link": {
        "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/
008f5d02-689d-4953-8b3e-0bd009b0616e",
        "rel": "self"
    },
    "op_id": "008f5d02-689d-4953-8b3e-0bd009b0616e",
    "resource": {
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-
c6efedc81fc3:",
        "link": {
            "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91"
    },
    "start_time": 1379958434711,
    "state": "pending"
}
```

2. Send a GET request to the task URI that was returned by the DELETE request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/
tasks/008f5d02-689d-4953-8b3e-0bd009b0616e
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

```
200 OK
{
    "end_time": 1379958435051,
    "link": {
        "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:/tasks/
008f5d02-689d-4953-8b3e-0bd009b0616e",
        "rel": "self"
    },
    "message": "Operation completed successfully",
}
```

```

    "op_id": "008f5d02-689d-4953-8b3e-0bd009b0616e",
    "resource": {
        "id": "urn:storageos:FileShare:840d01a9-8836-4c53-a95a-
c6efedc81fc3:",
        "link": {
            "href": "/file/filesystems/urn:storageos:FileShare:
840d01a9-8836-4c53-a95a-c6efedc81fc3:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91"
    },
    "start_time": 1379958434711,
    "state": "ready"
}

```

Deleting a file system

Deactivating a file system moves the file system to a marked-for-delete state, and it will be deleted when all references to this file system of type Snapshot are deleted. The optional forceDelete param will delete snapshots and exports when set to true in the case of VNXFile.

Before you begin

You need this information:

- ◆ URN of the file system being deleted
- ◆ value of the forceDelete parameter

Procedure

1. Delete the file system by sending a POST /file/filesystems/{id}/deactivate request.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```

POST https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:2105a51d-c867-4f2c-a779-68ae0c282d03:/deactivate
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<filesystem_deactivate>
    <forceDelete>true</forceDelete>
</filesystem_deactivate>

```

Response

```

202 Accepted
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task>
    <associated_resources/>
    <op_id>6a8b1c38-f60f-4470-a200-3efca9231d4b</op_id>
    <resource>
        <id>urn:storageos:FileShare:2105a51d-c867-4f2c-
a779-68ae0c282d03:</id>
            <link href="/file/filesystems/urn:storageos:FileShare:
2105a51d-c867-4f2c-a779-68ae0c282d03:" rel="self"/>
    <name>prov_NetApp_UnexportFilesystemWithInvalidPermissions_39</
name>
    </resource>
    <link href="/file/filesystems/urn:storageos:FileShare:
2105a51d-c867-4f2c-a779-68ae0c282d03:/tasks/6a8b1c38-f60f-4470-
a200-3efca9231d4b" rel="self"/>
        <start_time>1388787072029</start_time>
        <state>pending</state>
    </task>

```

2. Send a `GET` request to the task URI that was returned by the `POST` request to return the status of the task.

When the task completes successfully, the `message` attribute is set to `Operation completed successfully`.

Request

```
GET https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:2105a51d-c867-4f2c-a779-68ae0c282d03:/
tasks/6a8b1c38-f60f-4470-a200-3efca9231d4b
```

Response

```
200 OK
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task>
  <associated_resources/>
  <end_time>1388787074391</end_time>
  <message>Operation completed successfully</message>
  <op_id>6a8b1c38-f60f-4470-a200-3efca9231d4b</op_id>
  <resource>
    <id>urn:storageos:FileShare:2105a51d-c867-4f2c-
a779-68ae0c282d03:</id>
    <link href="/file/filesystems/urn:storageos:FileShare:
2105a51d-c867-4f2c-a779-68ae0c282d03:" rel="self"/>

    <name>prov_NetApp_UnexportFilesystemWithInvalidPermissions_39</
name>
    </resource>
    <link href="/file/filesystems/urn:storageos:FileShare:
2105a51d-c867-4f2c-a779-68ae0c282d03:/tasks/6a8b1c38-f60f-4470-
a200-3efca9231d4b" rel="self"/>
    <start_time>1388787072029</start_time>
    <state>ready</state>
  </task>
```

Snapshot

A snapshot is a point-in-time copy of a volume or a file system. Snapshots are intended for short-term operational recovery.

Snapshots have the following properties:

- ◆ A snapshot can be exported/unexported to a host, and you can delete it.
- ◆ A snapshot's lifetime is tied to the original volume/file system: when the original volume/file system is deleted, all of its snapshots will also be deleted.
- ◆ A volume/file system may be restored in place based on a snapshot.
- ◆ A snapshot is associated with the same project as the original volume/file system.
- ◆ A new volume/file system may be created using a snapshot as a template.
- ◆ Multi-volume consistent snapshots can be used to snapshot multiple volumes at once.

Creating a file system snapshot

A file system snapshot is created by sending a `POST /file/filesystems/{id}/protection/snapshots` request.

Before you begin

You need this information:

- ◆ Name of the file system snapshot

Procedure

1. Create the file system snapshot.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/file/filesystems/
urn:storageos:FileShare:840d01a9-8836-4c53-a95a-c6efedc81fc3:/protection/snapshots
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': {token_text},
'ACCEPT': 'application/json'}
```

```
{
    "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
}
```

Response

```
202 Accepted
{
    "description": "Filesystem snapshot create",
    "link": {
        "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/8eb0bfc0-c147-4a59-bcd-3b4294b5e02e",
        "rel": "self"
    },
    "op_id": "8eb0bfc0-c147-4a59-bcd-3b4294b5e02e",
    "resource": {
        "id": "urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
        "link": {
            "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
    },
    "start_time": 1379958438617,
    "state": "pending"
}
```

2. Send a GET request to the task URI that was returned by the POST request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/8eb0bfc0-c147-4a59-bcd-3b4294b5e02e
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

```
200 OK
```

```
{
    "description": "Filesystem snapshot create",
    "end_time": 1379958439009,
    "link": {
```

```

        "href": "/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/8eb0bfc0-c147-4a59-
bdcd-3b4294b5e02e",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "8eb0bfc0-c147-4a59-bdcd-3b4294b5e02e",
    "resource": {
        "id": "urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
        "link": {
            "href": "/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
    },
    "start_time": 1379958438617,
    "state": "ready"
}

```

Exporting a file system snapshot

A snapshot of a file system can be exported by sending a `POST /file/snapshots/{id}/exports` request.

Before you begin

You need this information:

- ◆ URN of the file system snapshot being exported
- ◆ Security type for the file snapshot export
- ◆ Permissions for the file snapshot export
- ◆ The user with root permissions
- ◆ The file snapshot export protocol
- ◆ Endpoints

Procedure

1. Export the file system snapshot.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```

POST https://<ViPR_VIP>:4443/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:/exports
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': {token_text},
'ACCEPT': 'application/json'}
{
    "endpoints": [
        "www.endpoint1.com",
        "www.endpoint2.com",
        "www.endpoint3.com"
    ],
    "permissions": "ro",
    "protocol": "NFS",
    "root_user": "nobody",
    "type": "sys"
}

```

Response

```

202 Accepted
{

```

```

    "link": {
      "href": "/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/aac817fa-72d1-4b8c-
ae73-5b02e6553242",
      "rel": "self"
    },
    "op_id": "aac817fa-72d1-4b8c-ae73-5b02e6553242",
    "resource": {
      "id": "urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
      "link": {
        "href": "/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
        "rel": "self"
      },
      "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
    },
    "start_time": 1379958442647,
    "state": "pending"
  }

```

- Send a **GET** request to the task URI that was returned by the **POST** request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```

GET https://<ViPR_VIP>:4443/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/aac817fa-72d1-4b8c-
ae73-5b02e6553242
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': {Token_Text},
  'ACCEPT': 'application/json'
}

```

Response

200 OK

```

{
  "end_time": 1379958443025,
  "link": {
    "href": "/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/aac817fa-72d1-4b8c-
ae73-5b02e6553242",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "aac817fa-72d1-4b8c-ae73-5b02e6553242",
  "resource": {
    "id": "urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
    "link": {
      "href": "/file/snapshots/urn:storageos:Snapshot:
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
      "rel": "self"
    },
    "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
  },
  "start_time": 1379958442647,
  "state": "ready"
}

```

Restoring a file system snapshot

A file system snapshot is restored by sending a `POST /file/snapshots/{id}/restore` request.

Before you begin

You need this information:

- ◆ URN of the file system snapshot which is being restored

Procedure

1. Restore the file system snapshot.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/restore
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

```
202 Accepted
{
    "link": {
        "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/6fcfd9ba-772b-4dc0-a5c6-76f93a5919ef",
        "rel": "self"
    },
    "op_id": "6fcfd9ba-772b-4dc0-a5c6-76f93a5919ef",
    "resource": {
        "id": "urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
        "link": {
            "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
    },
    "start_time": 1379958447488,
    "state": "pending"
}
```

2. Send a `GET` request to the task URI that was returned by the `POST` request to return the status of the task.

When the task completes successfully, the `message` attribute is set to `Operation completed successfully`.

Request

```
GET https://<ViPR_VIP>:4443/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/6fcfd9ba-772b-4dc0-a5c6-76f93a5919ef
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

200 OK

```
{
    "link": {
        "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/6fcfd9ba-772b-4dc0-a5c6-76f93a5919ef",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "6fcfd9ba-772b-4dc0-a5c6-76f93a5919ef",
    "resource": {
        "id": "urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
        "link": {
            "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
    },
    "state": "ready"
}
```

Unexporting a file system snapshot

A file system snapshot export is removed by sending a `DELETE /file/snapshots/{id}/exports/{protocol},{secType},{perm},{rootMapping}` request.

Before you begin

You need this information:

Requirement	Description
{id}	URN of the file system export being deleted
{protocol}	protocol of the file system export
{secType}	security type of the file system export
{perm}	permissions for the file system export
{root_mapping}	user with root permissions

Procedure

1. Remove the file system snapshot export.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
DELETE https://<ViPR_VIP>:4443/file/snapshots/
urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/
exports/NFS,sys,ro,nobody
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

202 Accepted

```
{
  "link": {
    "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/e2c8382b-c5e4-43ac-93dd-549e06ee97cf",
    "rel": "self"
  },
  "op_id": "e2c8382b-c5e4-43ac-93dd-549e06ee97cf",
  "resource": {
    "id": "urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
    "link": {
      "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
      "rel": "self"
    },
    "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
  },
  "start_time": 1379958451444,
  "state": "pending"
}
```

- Send a GET request to the task URI that was returned by the DELETE request to return the status of the task.

When the task completes successfully, the message attribute is set to Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/e2c8382b-c5e4-43ac-93dd-549e06ee97cf
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': {Token_Text},
  'ACCEPT': 'application/json'
}
```

Response

200 OK

```
{
  "end_time": 1379958451797,
  "link": {
    "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/e2c8382b-c5e4-43ac-93dd-549e06ee97cf",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "e2c8382b-c5e4-43ac-93dd-549e06ee97cf",
  "resource": {
    "id": "urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
    "link": {
      "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
      "rel": "self"
    },
    "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
  },
  "start_time": 1379958451444,
  "state": "ready"
}
```

Deleting a file system snapshot

A file system snapshot is deleted by sending a `POST /file/snapshots/{id}/deactivate` request which moves the snapshot to the marked-for-delete state. The snapshot is deleted by the garbage collector on a subsequent iteration.

Before you begin

You need this information:

- ◆ URN of the file system snapshot being deleted

Procedure

1. Delete the file system snapshot.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR VIP>:4443/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/deactivate
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
    'ACCEPT': 'application/json'
}
```

Response

```
202 Accepted
{
    "link": {
        "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/116cccf9-3cd1-4f77-bb44-08d09ea7e772",
        "rel": "self"
    },
    "op_id": "116cccf9-3cd1-4f77-bb44-08d09ea7e772",
    "resource": {
        "id": "urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
        "link": {
            "href": "/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:",
            "rel": "self"
        },
        "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"
    },
    "start_time": 1379958455395,
    "state": "pending"
}
```

2. Send a `GET` request to the task URI that was returned by the `POST` request to return the status of the task.

When the task completes successfully, the `message` attribute is set to `Operation completed successfully`.

Request

```
GET https://<ViPR VIP>:4443/file/snapshots/urn:storageos:Snapshot:24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/116cccf9-3cd1-4f77-bb44-08d09ea7e772
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': {Token_Text},
```

```
'ACCEPT': 'application/json'  
}
```

Response

200 OK

```
{  
    "end_time": 1379958455916,  
    "link": {  
        "href": "/file/snapshots/urn:storageos:Snapshot:  
24c83321-55a4-40d3-8a8f-2f6edbd18d49:/tasks/116cccf9-3cd1-4f77-  
bb44-08d09ea7e772",  
        "rel": "self"  
    },  
    "message": "Operation completed successfully",  
    "op_id": "116cccf9-3cd1-4f77-bb44-08d09ea7e772",  
    "resource": {  
        "id": "urn:storageos:Snapshot:  
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",  
        "link": {  
            "href": "/file/snapshots/urn:storageos:Snapshot:  
24c83321-55a4-40d3-8a8f-2f6edbd18d49:",  
            "rel": "self"  
        },  
        "name": "fs-cosnetappf-00:50:56:91:58:91-092313174649-2"  
    },  
    "start_time": 1379958455395,  
    "state": "ready"  
}
```

CHAPTER 9

Managing Volumes

This chapter contains the following topics.

◆ Create block volume	112
◆ Exporting a volume to a host	114
◆ Creating a block volume snapshot	116
◆ Deleting a block volume snapshot	118
◆ Unexporting a volume from a host	121
◆ Deleting a block volume	123

Create block volume

Creating a block volume requires you to issue a `POST` against the resource `/block/volumes`, and supply information defining the volume.

This API allows a user to create one or more volumes. The volumes are created in the same storage pool. All users can create a block volume. This is an asynchronous request.

Procedure

1. Call the following API to create the block volume.

Request

```
POST https://<ViPR_VIP>:4443/block/volumes

{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}'
  'ACCEPT': 'application/json'
}

{
  "count": 1,
  "name": "myvolume",
  "project": "urn:storageos:Project:cdb26784-9d70-413a-998c-8e6b7cd95719:",
  "size": "1GB",
  "varray": "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
  "vpool": "urn:storageos:VirtualPool:6cd9f843-1b41-4b9f-8eb1-a26797d0a268:"
}
```

Response

Response code 202

```
{
  "task": [
    {
      "link": {
        "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:/tasks/ac441447-3c88-4b62-b8e3-a38fcae4767a",
        "rel": "self"
      },
      "op_id": "ac441447-3c88-4b62-b8e3-a38fcae4767a",
      "resource": {
        "id": "urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
        "link": {
          "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
          "rel": "self"
        },
        "name": "myvolume"
      },
      "start_time": 1379202606192,
      "state": "pending"
    }
  ]
}
RESP = {
'task': [{}{'op_id': 'ac441447-3c88-4b62-b8e3-a38fcae4767a',
'link': {}}
```

```

    'href': '/block/volumes/{Volume_URN}/tasks/ac441447-3c88-4b62-
b8e3-a38fcae4767a',
    'rel': 'self',
    'state': 'pending', 'resource': {'link': {'href': '/block/
volumes/{Volume_URN}'},
    'rel': 'self'}, 'name': 'myvolume', 'id': '{Volume_URN}':
1379202606192]}
[{'op_id': 'ac441447-3c88-4b62-b8e3-a38fcae4767a',
'link': {'href': '/block/volumes/{Volume_URN}/tasks/
ac441447-3c88-4b62-b8e3-a38fcae4767a',
'rel': 'self'}, 'state': 'pending', 'resource': {'link':
{'href': '/block/volumes/{Volume_URN}'},
'rel': 'self'}, 'name': 'myvolume', 'id': '{Volume_URN}':
1379202606192}]

```

2. Monitor the task to see when the volume create operation is complete. To do this, issue a GET on the task ID returned from the Volume Create API call.

Request

```
GET https://<ViPR_VIP>:4443/block/volumes/{Volume_ID}/tasks/
{Task_ID}
```

Headers:

```
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_ID}',
  'ACCEPT': 'application/json'
}
```

Response

HTTP 200

```
{
  "link": {
    "href": "/block/volumes/urn:storageos:Volume:
46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:/tasks/ac441447-3c88-4b62-
b8e3-a38fcae4767a",
    "rel": "self"
  },
  "op_id": "ac441447-3c88-4b62-b8e3-a38fcae4767a",
  "resource": {
    "id": "urn:storageos:Volume:46a3f479-7bb0-446f-
b2b3-6ef77ea27d0d:",
    "link": {
      "href": "/block/volumes/urn:storageos:Volume:
46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
      "rel": "self"
    },
    "name": "myvolume"
  },
  "start_time": 1379202606192,
  "state": "pending"
}
```

3. Call the URL in step 2 periodically.

The task is complete when the message is Operation completed successfully.

Response

```
{
  "end_time": 1379202631981,
  "link": {
    "href": "/block/volumes/urn:storageos:Volume:
46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:/tasks/ac441447-3c88-4b62-
b8e3-a38fcae4767a",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "ac441447-3c88-4b62-b8e3-a38fcae4767a",
```

```

    "resource": {
        "id": "urn:storageos:Volume:46a3f479-7bb0-446f-
b2b3-6ef77ea27d0d:",
        "link": {
            "href": "/block/volumes/urn:storageos:Volume:
46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
            "rel": "self"
        },
        "name": "myvolume"
    },
    "start_time": 1379202606192,
    "state": "ready"
}

```

Exporting a volume to a host

In a ViPR environment, hosts that belong to the same export group as a volume can access that volume.

Before you begin

You need this information:

- ◆ The URN of a host in the ViPR environment.
- ◆ The URN of a project
- ◆ The URN of a virtual array
- ◆ The URN of a volume
- ◆ A valid authentication token

This is an asynchronous task.

Procedure

1. Call the following POST to create an export group and add a host and a volume to that export group.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```

POST https://<ViPR_VIP>:4443/block/exports

{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}

{
  "hosts": [
    "urn:storageos:Host:037aeb4e-c6e4-4fa3-8a67-6db94a93a6c5:",
  ],
  "name": "myhost-exports",
  "project": "urn:storageos:Project:cdb26784-9d70-413a-998c-8e6b7cd95719:",
  "type": "Host",
  "varray": "urn:storageos:VirtualArray:dc09417d-9028-40de-8bff-74574996f2d5:",
  "volumes": [
    {
      "id": "urn:storageos:Volume:46a3f479-7bb0-446f-
b2b3-6ef77ea27d0d:"
    }
  ]
}

```

Response

```

HTTP 202 Accepted
{
    "description": "create export",
    "link": {
        "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:/tasks/1cd9d4c4-8f5a-49a6-91c4-c93c3c058327",
        "rel": "self"
    },
    "message": "create export",
    "op_id": "1cd9d4c4-8f5a-49a6-91c4-c93c3c058327",
    "resource": {
        "id": "urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
        "link": {
            "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
            "rel": "self"
        },
        "name": "myhost-exports"
    },
    "start_time": 1379202635022,
    "state": "pending"
}

```

2. Check the status of the asynchronous task periodically.

The volume export is complete when the message is Operation completed successfully.

Request

```

GET https://<ViPR_VIP>:4443/block/exports/{Export-Group_ID}/tasks/1cd9d4c4-8f5a-49a6-91c4-c93c3c058327
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}

```

Response

```

200 OK
Response
{
    "description": "create export",
    "end_time": 1379202732747,
    "link": {
        "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:/tasks/1cd9d4c4-8f5a-49a6-91c4-c93c3c058327",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "1cd9d4c4-8f5a-49a6-91c4-c93c3c058327",
    "resource": {
        "id": "urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
        "link": {
            "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
            "rel": "self"
        },
        "name": "myhost-exports"
    },
    "start_time": 1379202635022,
    "state": "ready"
}

```

Creating a block volume snapshot

You can archive a copy of a volume at a specific point in time by sending a `POST /block/volumes/{id}/protection/snapshots` request to create a snapshot.

Before you begin

You need the following information:

- ◆ A valid authentication token.
- ◆ The URN of a ViPR volume.

Procedure

1. Call the following POST to create the volume snapshot.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/block/volumes/{Volume_URN}/protection/snapshots
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}

{
    "create_inactive": null,
    "name": "mysnapshot",
    "type": null
}
```

Response

HTTP 202 Accepted

```
{
    "task": [
        {
            "description": "Block snapshot create",
            "link": {
                "href": "/block/snapshots/urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:/tasks/29875573-5d2f-46f9-9522-609311786120",
                "rel": "self"
            },
            "message": "Block snapshot create",
            "op_id": "29875573-5d2f-46f9-9522-609311786120",
            "resource": {
                "id": "urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
                "link": {
                    "href": "/block/snapshots/urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
                    "rel": "self"
                },
                "name": "mysnapshot"
            },
            "start_time": 1379202733863,
            "state": "pending"
        }
    ]
}
```

- Call the following GET on the task ID to check the status of the create snapshot operation.

```
GET https://<ViPR_VIP>:4443/block/snapshots/{Snapshot_URN}/tasks/29875573-5d2f-46f9-9522-609311786120
```

```
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}
```

Response

HTTP 200 OK

```
{
  "description": "Block snapshot create",
  "link": {
    "href": "/block/snapshots/urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:/tasks/29875573-5d2f-46f9-9522-609311786120",
    "rel": "self"
  },
  "message": "Block snapshot create",
  "op_id": "29875573-5d2f-46f9-9522-609311786120",
  "resource": {
    "id": "urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
    "link": {
      "href": "/block/snapshots/urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
      "rel": "self"
    },
    "name": "mysnapshot"
  },
  "start_time": 1379202733863,
  "state": "pending"
}
```

- Check the status of the asynchronous task periodically.

The volume export is complete when the message is Operation completed successfully.

```
GET https://<ViPR_VIP>:4443/block/snapshots/{Snapshot_URN}/tasks/29875573-5d2f-46f9-9522-609311786120
```

```
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}
```

Response

200 OK

```
{
  "description": "Block snapshot create",
  "end_time": 1379202798024,
  "link": {
    "href": "/block/snapshots/urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:/tasks/29875573-5d2f-46f9-9522-609311786120",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "29875573-5d2f-46f9-9522-609311786120",
  "resource": {
    "id": "urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
    "link": {
      "href": "/block/snapshots/urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
```

```

        "rel": "self"
    },
    "name": "mysnapshot"
},
"start_time": 1379202733863,
"state": "ready"
}
}
```

Deleting a block volume snapshot

To delete a block volume snapshot, you must deactivate the snapshot and delete the export group to which it belongs.

Before you begin

You need the following information:

- ◆ The URN of the volume snapshot you want to delete.
- ◆ The URN of the volume from which the snapshot was created.
- ◆ The URN of the export group to which the snapshot belongs.
- ◆ A valid ViPR authentication token.

This is an asynchronous request.

Procedure

1. Call the following POST to deactivate the snapshot.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/block/volumes/{Volume_URN}/protection/
snapshots/{snapshot_urn}/deactivate

{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}
```

Response

```
HTTP 202 Accepted
{
  "link": {
    "href": "/block/snapshots/
urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:/
tasks/c04b87a6-7802-4d81-8081-eb57fce8317a",
    "rel": "self"
  },
  "op_id": "c04b87a6-7802-4d81-8081-eb57fce8317a",
  "resource": {
    "id": "urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-
ac36-01468bf73975:",
    "link": {
      "href": "/block/snapshots/
urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
      "rel": "self"
    },
    "name": "mysnapshot"
  },
  "start_time": 1379202799026,
  "state": "pending"
}
```

- Call the following GET to monitor the delete snapshot operation status.

Request

```
GET https://<ViPR_VIP>:4443/block/snapshots/{snapshot_urn}/tasks/c04b87a6-7802-4d81-8081-eb57fce8317a
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}
```

Response

```
200 OK
{
    "link": {
        "href": "/block/snapshots/
urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:/tasks/c04b87a6-7802-4d81-8081-eb57fce8317a",
        "rel": "self"
    },
    "op_id": "c04b87a6-7802-4d81-8081-eb57fce8317a",
    "resource": {
        "id": "urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
        "link": {
            "href": "/block/snapshots/
urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
            "rel": "self"
        },
        "name": "mysnapshot"
    },
    "start_time": 1379202799026,
    "state": "pending"
}
```

- Call the GET described in step 2 periodically.

Snapshot deactivation is complete when message is Operation completed successfully.

```
{
    "end_time": 1379202843313,
    "link": {
        "href": "/block/snapshots/
urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:/tasks/c04b87a6-7802-4d81-8081-eb57fce8317a",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "c04b87a6-7802-4d81-8081-eb57fce8317a",
    "resource": {
        "id": "urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
        "link": {
            "href": "/block/snapshots/
urn:storageos:BlockSnapshot:e8b3170b-65c3-4d2f-ac36-01468bf73975:",
            "rel": "self"
        },
        "name": "mysnapshot"
    },
    "start_time": 1379202799026,
    "state": "ready"
}
```

- Call the following POST to delete the export group that contains the snapshot.

```
POST https://<ViPR_VIP>:4443/block/exports/{exportgroup_urn}/deactivate
{
    'Content-Type': 'application/json',
}
```

```
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'
}
```

This is an asynchronous request.

5. Call the following GET to monitor the export group delete status.

Request

```
GET https://<ViPR_VIP>:4443/block/exports/
export_groupurn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-
ff63781049dc:/tasks/8eecbaa2-b6b5-4075-b723-9eea93d51e72
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}
```

Response

```
HTTP 200
{
    "description": "delete export",
    "link": {
        "href": "/block/exports/urn:storageos:ExportGroup:
3074f7bd-4c6d-4259-9dc5-ff63781049dc:/tasks/8eecbaa2-b6b5-4075-
b723-9eea93d51e72",
        "rel": "self"
    },
    "message": "delete export",
    "op_id": "8eecbaa2-b6b5-4075-b723-9eea93d51e72",
    "resource": {
        "id": "urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-
ff63781049dc:",
        "link": {
            "href": "/block/exports/urn:storageos:ExportGroup:
3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
            "rel": "self"
        },
        "name": "myhost-exports"
    },
    "start_time": 1379202845761,
    "state": "pending"
}
```

6. Call the GET described in step 5 periodically.

The export group delete is complete when message is Operation completed successfully.

```
{
    {
        "description": "delete export",
        "end_time": 1379202949620,
        "link": {
            "href": "/block/exports/urn:storageos:ExportGroup:
3074f7bd-4c6d-4259-9dc5-ff63781049dc:/tasks/8eecbaa2-b6b5-4075-
b723-9eea93d51e72",
            "rel": "self"
        },
        "message": "Operation completed successfully",
        "op_id": "8eecbaa2-b6b5-4075-b723-9eea93d51e72",
        "resource": {
            "id": "urn:storageos:ExportGroup:
3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
            "link": {
                "href": "/block/exports/urn:storageos:ExportGroup:
3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
                "rel": "self"
            },
            "name": "myhost-exports"
        },
    }
}
```

```

        "start_time": 1379202845761,
        "state": "ready"
    }
]
```

Unexporting a volume from a host

To unexport a volume from a host, deactivate the export group to which the volume and host belong. The host can no longer access the volume.

Before you begin

You need the following information:

- ◆ The URN of the export group to which the host and volume belong.
- ◆ A valid ViPR authentication token.

This is an asynchronous request.

Procedure

1. Call the following POST to deactivate the export group.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/block/exports/{ExportGroup_URN}/deactivate
{
    'Content-Type': 'application/json',
    'X-SDS-AUTH-TOKEN': '{Token_Text}',
    'ACCEPT': 'application/json'
}
```

Response

```
HTTP 202 Accepted
{
    "description": "delete export",
    "link": {
        "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:/tasks/8eecbaa2-b6b5-4075-b723-9eea93d51e72",
        "rel": "self"
    },
    "message": "delete export",
    "op_id": "8eecbaa2-b6b5-4075-b723-9eea93d51e72",
    "resource": {
        "id": "urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
        "link": {
            "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
            "rel": "self"
        },
        "name": "myhost-exports"
    },
    "start_time": 1379202845761,
    "state": "pending"
}
```

2. Monitor the deactivate export group operation by calling the following GET on the task ID.

Request

```
GET https://<ViPR_VIP>:4443/block/exports/{Export_Group_URN}/tasks/8eecbaa2-b6b5-4075-b723-9eea93d51e72
```

```
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
}
```

Response

HTTP 200 OK

```
{
  "description": "delete export",
  "link": {
    "href": "/block/exports/{Export_Group_URN}/tasks/8eecbaa2-b6b5-4075-b723-9eea93d51e72",
    "rel": "self"
  },
  "message": "delete export",
  "op_id": "8eecbaa2-b6b5-4075-b723-9eea93d51e72",
  "resource": {
    "id": "urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
    "link": {
      "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
      "rel": "self"
    },
    "name": "myhost-exports"
  },
  "start_time": 1379202845761,
  "state": "pending"
}
```

- Call the GET request in step 2 periodically. The export group deactivation is complete when the message is Operation completed successfully.

Response

HTTP 200

```
{
  "description": "delete export",
  "end_time": 1379202949620,
  "link": {
    "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:/tasks/8eecbaa2-b6b5-4075-b723-9eea93d51e72",
    "rel": "self"
  },
  "message": "Operation completed successfully",
  "op_id": "8eecbaa2-b6b5-4075-b723-9eea93d51e72",
  "resource": {
    "id": "urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
    "link": {
      "href": "/block/exports/urn:storageos:ExportGroup:3074f7bd-4c6d-4259-9dc5-ff63781049dc:",
      "rel": "self"
    },
    "name": "myhost-exports"
  },
  "start_time": 1379202845761,
  "state": "ready"
}
```

Deleting a block volume

A block volume is deleted by sending a POST `/block/volumes/{Volume_URN}/deactivate` request.

Before you begin

You need the following information:

- ◆ The volume must not be part of any export group.
- ◆ The URN of the volume you want to delete.
- ◆ A valid ViPR authentication token.

This is an asynchronous request.

Procedure

1. Call the following POST to deactivate the volume.

The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/block/volumes/{Volume_URN}/deactivate
Headers:
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
```

Response

```
HTTP 202 Accepted
{
  "link": {
    "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:/tasks/4c9b1cc6-47ce-4158-88df-2ee12322ef59",
    "rel": "self"
  },
  "op_id": "4c9b1cc6-47ce-4158-88df-2ee12322ef59",
  "resource": {
    "id": "urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
    "link": {
      "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
      "rel": "self"
    },
    "name": "myvolume"
  },
  "start_time": 1379202950804,
  "state": "pending"
}
```

2. Monitor the volume delete operation by calling the following GET on the task ID.

Request

```
GET https://<ViPR_VIP>:4443/block/volumes/{Volume_URN}/tasks/4c9b1cc6-47ce-4158-88df-2ee12322ef59
{
  'Content-Type': 'application/json',
  'X-SDS-AUTH-TOKEN': '{Token_Text}',
  'ACCEPT': 'application/json'
```

Response

```
HTTP 200 OK
{
    "link": {
        "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:/tasks/4c9b1cc6-47ce-4158-88df-2ee12322ef59",
        "rel": "self"
    },
    "op_id": "4c9b1cc6-47ce-4158-88df-2ee12322ef59",
    "resource": {
        "id": "urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
        "link": {
            "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
            "rel": "self"
        },
        "name": "myvolume"
    },
    "start_time": 1379202950804,
    "state": "pending"
}
```

3. Call the GET request in step 2 periodically. The volume delete is complete when the message is Operation completed successfully.

Response

```
HTTP 200 OK
{
    "end_time": 1379203014362,
    "link": {
        "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:/tasks/4c9b1cc6-47ce-4158-88df-2ee12322ef59",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "4c9b1cc6-47ce-4158-88df-2ee12322ef59",
    "resource": {
        "id": "urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
        "link": {
            "href": "/block/volumes/urn:storageos:Volume:46a3f479-7bb0-446f-b2b3-6ef77ea27d0d:",
            "rel": "self"
        },
        "name": "myvolume"
    },
    "start_time": 1379202950804,
    "state": "ready"
}
```

CHAPTER 10

Managing Hosts

This chapter contains the following topics:

◆ Host API Overview	126
◆ Adding and discovering a Windows host	127
◆ Adding and discovering a LINUX host	130
◆ Adding and discovering a vCenter Server	131

Host API Overview

Hosts that are part of your data center can be added and managed using the ViPR Controller REST API.

The following table shows some important APIs that are used to manage hosts, vCenters and Windows clusters.

ViPR REST API Call	Description
POST /tenant/{tenant_urn}/hosts	Create a host resource in ViPR.
PUT /compute/hosts/{Host_URN}	Modify parameters for a host.
POST /admin/api/hosts/{{Host_URN}}/discover	Add a host to the discovery queue. Host discovery is performed asynchronously. This call is directed to port 443.
POST /tenants/{Tenant_URN}/vccenters	Add a vCenter resource to the tenant organization.
GET /compute/vcenters/{Vcenter_URN}/clusters	List the clusters in a vCenter.
GET /compute/vcenters/{Vcenter_URN}/hosts	List the hosts in a vCenter.
POST <ViPR_VIP>:443/admin/api/vcenters/{vcenter_URN}/discover	Discover a vCenter. This call is directed to port 443.
GET /compute/clusters/{Cluster_URN}/hosts	Lists the hosts in a cluster.
GET /tenants/{Tenant_URN}/clusters	List all clusters that belong to the specified tenant. Clusters, like hosts and vCenters, are tenant-level resources.
GET /compute/clusters/{Cluster_URN}	Retrieve a cluster resource.
POST /tenants/{id}/clusters	Build a ViPR cluster.
PUT /compute/hosts/{id}	Add a host to a cluster. The payload for this call allows you to specify a cluster name. <pre data-bbox="798 1558 1314 1738"><host_update> <type>Windows</type> <host_name>myhost.corp.com</host_name> <name>myHost</name> <user_name>admin</user_name> <password>password</password> <cluster>vipr_cluster</cluster> </host_update></pre>

Adding and discovering a Windows host

To manage a Windows host in ViPR, you must register and discover that host.

Before you begin

Your Windows host must be running the correct version of Windows, and be properly configured. In particular, your ViPR installation, the Windows host, and the user name you use to authenticate must all be in the same Windows domain.

You must be logged in as a tenant administrator to perform the commands in this procedure.

Note

Some of the API calls in the following procedure are sent to Port 4443, and others are sent to port 443. The port number is included in the examples for clarity.

Procedure

1. Get your tenant's URN.

```
GET https://<ViPR_VIP>:4443/tenant
```

2. Use the tenant URN to create a host.

Request

```
POST https://<ViPR_VIP>:4443/tenants/{Tenant_URN}/hosts
```

```
<host_create>
  <type>Windows</type>
  <host_name>lglbc012.lss.emc.com</host_name>
  <name>lglbc012</name>
  <port_number>5985</port_number>
  <user_name>Administrator</user_name>
  <password>Dangerous01</password>
  <use_ssl>No</use_ssl>
</host_create>
```

Response

```
HTTP 202 Accepted
```

```
<task>
  <associated_resources/>
  <message>Operation completed successfully</message>
  <op_id>f50ca6c4-d66c-4ddf-868f-3703c2a092df</op_id>
  <resource>
    <id>urn:storageos:Host:e34e171a-97b7-4dc9-b3b7-c57fb9c3e45c:</id>
    <link href="/compute/hosts/urn:storageos:Host:e34e171a-97b7-4dc9-b3b7-c57fb9c3e45c:" rel="self"/>
      <name>lglbc012</name>
    </resource>
    <link href="/compute/hosts/urn:storageos:Host:e34e171a-97b7-4dc9-b3b7-c57fb9c3e45c:/tasks/f50ca6c4-d66c-4ddf-868f-3703c2a092df" rel="self"/>
      <start_time>1386365633873</start_time>
      <state>ready</state>
    </task>
```

If the message from the returned task is Operation completed successfully, the host is now added to ViPR. You will not be able to fully manage the host until you discover it.

3. Get the URN of the new host from the response to the `host create` call, and call the following API to discover the host, and its initiator ports.

```
POST https://<ViPR_VIP>:443/admin/api/hosts/
urn:storageos:Host:e34e171a-97b7-4dc9-b3b7-c57fb9c3e45c:/discover
```

4. Check the user interface to see if the host has completed discovery, or check the host resource. The host should contain the status shown in the following example.

Request

```
GET https://<ViPR_VIP>:4443/compute/hosts/
urn:storageos:Host:e34e171a-97b7-4dc9-b3b7-c57fb9c3e45c:
```

Response

```
HTTP 200 OK
<host>
  <creation_time>1386365633865</creation_time>
  <id>urn:storageos:Host:e34e171a-97b7-4dc9-b3b7-c57fb9c3e45c:</id>
  <inactive>false</inactive>
  <link href="/compute/hosts/urn:storageos:Host:e34e171a-97b7-4dc9-
b3b7-c57fb9c3e45c:" rel="self"/>
  <name>lglbc012</name>
  <tags/>
  <native_guid/>
  <compatibility_status>COMPATIBLE</compatibility_status>
  <job_discovery_status>COMPLETE</job_discovery_status>
  <last_discovery_run_time>1386366866868</last_discovery_run_time>
  <last_discovery_status_message/>
  <last_metering_run_time>0</last_metering_run_time>
  <job_metering_status>CREATED</job_metering_status>
  <next_discovery_run_time>0</next_discovery_run_time>
  <next_metering_run_time>0</next_metering_run_time>
  <registration_status>REGISTERED</registration_status>
  <tenant>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:</id>
    <link href="/tenants/
urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
    </tenant>
    <host_name>lglbc012.lss.emc.com</host_name>
    <os_version>6.2.9200 Microsoft Windows Server 2012 Standard</
os_version>
    <port_number>5985</port_number>
    <type>Windows</type>
    <use_ssl>false</use_ssl>
    <user_name>Administrator</user_name>
  </host>
```

Windows clusters

When you discover a Windows host that is part of a windows cluster, that cluster is added to ViPR. ViPR clusters allow you to allocate block storage to a group of hosts.

ViPR does not directly discover Windows clusters. Windows clusters are discovered when you discover a host belonging to that cluster. Windows host resources specify the ID of the cluster when the host belongs to a cluster.

```
<cluster>
  <id>urn:storageos:Cluster:37dcbb9-2158-4976-b937-c167e8e738a2:</
id>
  <link href="/compute/clusters/urn:storageos:Cluster:
37dcbb9-2158-4976-b937-c167e8e738a2:" rel="self"/>
</cluster>
```

If the host does not belong to a cluster, the cluster is omitted from the resource.

The following shows a Windows host that belongs to a cluster.

Request

```
GET https://<ViPR_VIP>:4443/compute/hosts/urn:storageos:Host:0f034ce7-b830-4a47-8cc4-71b6b792589b:
```

Response

```
HTTP 200 OK
<host>
  <creation_time>1386882454404</creation_time>
  <id>urn:storageos:Host:0f034ce7-b830-4a47-8cc4-71b6b792589b:</id>
  <inactive>false</inactive>
  <link href="/compute/hosts/urn:storageos:Host:0f034ce7-b830-4a47-8cc4-71b6b792589b:" rel="self"/>
  <name>lglw7150.lss.emc.com</name>
  <tags/>
  <native_guid/>
  <compatibility_status>COMPATIBLE</compatibility_status>
  <job_discovery_status>COMPLETE</job_discovery_status>
  <last_discovery_run_time>1386882462221</last_discovery_run_time>
  <last_discovery_status_message/>
  <last_metering_run_time>0</last_metering_run_time>
  <job_metering_status>CREATED</job_metering_status>
  <next_discovery_run_time>0</next_discovery_run_time>
  <next_metering_run_time>0</next_metering_run_time>
  <registration_status>REGISTERED</registration_status>
  <success_discovery_time>0</success_discovery_time>
  <success_metering_time>0</success_metering_time>
  <tenant>
    <id>urn:storageos:TenantOrg:7985d438-9980-41df-bba1-29d6a873f811:</id>
    <link href="/tenants/urn:storageos:TenantOrg:7985d438-9980-41df-bba1-29d6a873f811:" rel="self"/>
  </tenant>
  <cluster>
    <id>urn:storageos:Cluster:37dcbb9-2158-4976-b937-c167e8e738a2:</id>
    <link href="/compute/clusters/urn:storageos:Cluster:37dcbb9-2158-4976-b937-c167e8e738a2:" rel="self"/>
  </cluster>
  <discoverable>true</discoverable>
  <host_name>lglw7150.lss.emc.com</host_name>
  <os_version>6.2.9200 Microsoft Windows Server 2012 Standard</os_version>
  <port_number>5985</port_number>
  <type>Windows</type>
  <use_ssl>false</use_ssl>
  <user_name>administrator</user_name>
</host>
```

The following shows a cluster resource.

Request

```
GET https://<ViPR_VIP>:4443/compute/clusters/urn:storageos:Cluster:37dcbb9-2158-4976-b937-c167e8e738a2:
```

Response

```
HTTP 200 OK
<cluster>
  <creation_time>1386882461075</creation_time>
  <id>urn:storageos:Cluster:37dcbb9-2158-4976-b937-c167e8e738a2:</id>
  <inactive>false</inactive>
  <link href="/compute/clusters/urn:storageos:Cluster:37dcbb9-2158-4976-b937-c167e8e738a2:" rel="self"/>
  <name>ViPR-WINDOWS</name>
  <tags/>
  <tenant>
    <id>urn:storageos:TenantOrg:7985d438-9980-41df-bba1-29d6a873f811:</id>
    <link href="/tenants/urn:storageos:TenantOrg:7985d438-9980-41df-bba1-29d6a873f811:" rel="self"/>
```

```
</tenant>
</cluster>
```

Adding and discovering a LINUX host

To manage a LINUX host in ViPR, you have to register the host, then successfully discover it.

Before you begin

Your LINUX host must be running the correct version of LINUX and be properly configured. In particular, your ViPR installation, the LINUX host, and the user name that you use to authenticate must all be in the same network domain. See the *EMC ViPR Installation and Configuration Guide* for more information.

You must be logged in as a tenant administrator to perform the commands in this procedure.

Note

Some of the API calls in the following procedure are sent to port 4443, and others are sent to port 443. The port number is included in the examples for clarity.

Procedure

1. Get your tenant's URN.

```
GET https://<ViPR_VIP>:4443/tenant
```

2. Use the tenant URN to create a host.

Request

```
POST https://<ViPR_VIP>:4443/tenants/{Tenant_URN}/hosts
```

```
<host_create>
  <type>Linux</type>
  <host_name>lglw7141.lss.emc.com</host_name>
  <name>lglw7141</name>
  <port_number>22</port_number>
  <user_name>root</user_name>
  <password>dangerous</password>
  <use_ssl>No</use_ssl>
</host_create>
```

Response

HTTP 202 Accepted

```
<task>
  <associated_resources/>
  <message>Operation completed successfully</message>
  <op_id>90827e3a-ce0f-42b3-9499-d43986dba44b</op_id>
  <resource>
    <id>urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:</id>
    <link href="/compute/hosts/urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:" rel="self"/>
      <name>lglw7141</name>
    </resource>
    <link href="/compute/hosts/urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:/tasks/90827e3a-ce0f-42b3-9499-d43986dba44b" rel="self"/>
      <start_time>1386618442169</start_time>
      <state>ready</state>
  </task>
```

If the message from the returned task is `Operation completed successfully`, the host is now added to ViPR. You will not be able to fully manage the host until you discover it.

3. Get the URN of the new host from the response to the `host create` call, and call the following API to discover the host, and its initiator ports.

```
POST https://<ViPR_VIP>:443/admin/api/hosts/
urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:/discover
```

4. Check the user interface to see if the host has completed discovery, or check the host resource. It should contain the information shown in the following example.

Request

```
GET https://<ViPR_VIP>:4443/compute/hosts/
urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:
```

Response

```
<host>
  <creation_time>1386618442154</creation_time>
  <id>urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:</id>
  <inactive>false</inactive>
  <link href="/compute/hosts/
  urn:storageos:Host:ead39b5a-07f4-4cb4-8124-35aa864fe760:" rel="self"/>
  <name>lglw7141</name>
  <tags/>
  <native_guid/>
  <compatibility_status>COMPATIBLE</compatibility_status>
  <job_discovery_status>COMPLETE</job_discovery_status>
  <last_discovery_run_time>1386619685871</last_discovery_run_time>
  <last_discovery_status_message/>
  <last_metering_run_time>0</last_metering_run_time>
  <job_metering_status>CREATED</job_metering_status>
  <next_discovery_run_time>0</next_discovery_run_time>
  <next_metering_run_time>0</next_metering_run_time>
  <registration_status>REGISTERED</registration_status>
  <tenant>
    <id>urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-
c40b6de2c72f:</id>
    <link href="/tenants/
  urn:storageos:TenantOrg:d61d9fa1-9886-40ef-85d3-c40b6de2c72f:" rel="self"/>
  </tenant>
  <host_name>lglw7141.lss.emc.com</host_name>
  <os_version>RHEL 5.9.0.2</os_version>
  <port_number>22</port_number>
  <type>Linux</type>
  <use_ssl>false</use_ssl>
  <user_name>root</user_name>
</host>
```

Adding and discovering a vCenter Server

To manage a vCenter server in ViPR, you have to add the server to ViPR through the ViPR user interface or through the ViPR REST API. ViPR automatically discovers the hosts, clusters and other resources managed by that server.

Before you begin

Your vCenter server must adhere to the parameters specified in the VMWare Requirements section of the *EMC ViPR Data Sheet and Compatibility Matrix*.

You must be logged in as a system administrator to perform the commands in this procedure.

Note

Some of the API calls in the following procedure are sent to Port 4443, and others are sent to port 443. The port number is included in the examples for clarity.

Procedure

1. Get your tenant's URN.

```
GET https://<ViPR_VIP>:4443/tenant
```

2. Use the tenant URN to create a host.

Request

```
POST https://<ViPR_VIP>:4443/tenants/{Tenant_URN}/vcenters
```

```
<vcenter_create>
  <ip_address>myvcenter.corp.emc.com</ip_address>
  <name>myvcenter.corp.emc.com</name>
  <user_name>domain\username</user_name>
  <password>pa$$words</password>
</vcenter_create>
```

Response

```
HTTP 202 Accepted
```

```
<task>
  <associated_resources/>
  <message>Operation completed successfully</message>
  <op_id>1bbd3c99-5f59-4eef-9152-f149357e7acb</op_id>
  <resource>
    <id>urn:storageos:Vcenter:5bb76aae-1b75-4fca-89ec-e463df0cba13:</id>
    <link href="/compute/vcenters/urn:storageos:Vcenter:5bb76aae-1b75-4fca-89ec-e463df0cba13:" rel="self"/>
    <name>my_vcenter</name>
  </resource>
  <link href="/compute/vcenters/urn:storageos:Vcenter:5bb76aae-1b75-4fca-89ec-e463df0cba13:/tasks/1bbd3c99-5f59-4eef-9152-f149357e7acb" rel="self"/>
  <start_time>1386705168500</start_time>
  <state>ready</state>
</task>
```

If the message from the returned task is Operation completed successfully, the vCenter is now added to ViPR. You will not be able to fully manage the vCenter until you discover it.

3. Get the URN of the new vcenter from the response to the vcenter create call, and call the following API to discover the vcenter.

```
POST <ViPR_VIP>:443/admin/api/vcenters/urn:storageos:Vcenter:5bb76aae-1b75-4fca-89ec-e463df0cba13:/discover
```

4. Check the user interface to see if the host has completed discovery, or check the vcenter resource. It should contain the information shown in the following example.

Request

```
GET https://<ViPR_VIP>:4443/compute/vcenters/urn:storageos:Vcenter:5bb76aae-1b75-4fca-89ec-e463df0cba13:
```

Response

```
HTTP 200 OK
```

```
<vcenter>
  <creation_time>1386860663431</creation_time>
  <id>urn:storageos:Vcenter:5bb76aae-1b75-4fca-89ec-e463df0cba13:</id>
  <inactive>false</inactive>
```

```

<link href="/compute/vcenters/urn:storageos:Vcenter:
5bb76aae-1b75-4fca-89ec-e463df0cba13:" rel="self"/>
<name>Lglba041</name>
<tags/>
<native_guid/>
<compatibility_status>COMPATIBLE</compatibility_status>
<job_discovery_status>COMPLETE</job_discovery_status>
<last_discovery_run_time>1386860840239</last_discovery_run_time>
<last_discovery_status_message/>
<last_metering_run_time>0</last_metering_run_time>
<job_metering_status>CREATED</job_metering_status>
<next_discovery_run_time>0</next_discovery_run_time>
<next_metering_run_time>0</next_metering_run_time>
<registration_status>REGISTERED</registration_status>
<success_discovery_time>0</success_discovery_time>
<success_metering_time>0</success_metering_time>
<tenant>
    <id>urn:storageos:TenantOrg:7985d438-9980-41df-
bbal-29d6a873f811:</id>
    <link href="/tenants/urn:storageos:TenantOrg:
7985d438-9980-41df-bbal-29d6a873f811:" rel="self"/>
</tenant>
    <ip_address>10.247.166.41</ip_address>
    <os_version>5.1.0</os_version>
    <user_name>cc-ro</user_name>
</vcenter>
```

Examples: Accessing vCenter resources

vCenters contain clusters and hosts. The hosts contain initiator ports that allow the host to access storage. All of these resources are accessible from ViPR.

Listing vCenter clusters

Request

```
GET https://<ViPR_VIP>:4443/compute/vcenters/
urn:storageos:Vcenter:ce774753-3cbb-4b3e-845c-feb6ef29428e:/clusters
```

Response

```

HTTP 200 OK
<clusters>
    <cluster>
        <id>urn:storageos:Cluster:c8fe93d7-a7c2-4c93-a338-a31e22aab490:</
id>
        <link href="/compute/clusters/urn:storageos:Cluster:c8fe93d7-
a7c2-4c93-a338-a31e22aab490:" rel="self"/>
        <name>Mgmt-Cluster</name>
    </cluster>
    <cluster>
        <id>urn:storageos:Cluster:37e6b15a-d503-47cc-b18a-6d59594001b9:</
id>
        <link href="/compute/clusters/urn:storageos:Cluster:37e6b15a-
d503-47cc-b18a-6d59594001b9:" rel="self"/>
        <name>CSE-Cluster-CISCO</name>
    </cluster>
</clusters>
```

Listing hosts in a vCenter

Request

```
GET https://<ViPR_VIP>:4443/compute/vcenters/
urn:storageos:Vcenter:ce774753-3cbb-4b3e-845c-feb6ef29428e:/hosts
```

Response

```

HTTP 200 OK
<hosts>
    <host>
        <id>urn:storageos:Host:003422cf-6596-43da-b382-816486fd58e9:</id>
        <link href="/compute/hosts/urn:storageos:Host:003422cf-6596-43da-
```

```
b382-816486fd58e9;" rel="self"/>
    <name>lglba180.lss.emc.com</name>
</host>
<host>
    <id>urn:storageos:Host:492ed254-282f-4cd4-bb76-34777db9152e:</id>
    <link href="/compute/hosts/urn:storageos:Host:492ed254-282f-4cd4-
bb76-34777db9152e;" rel="self"/>
    <name>lglac019.lss.emc.com</name>
</host>
<host>
    <id>urn:storageos:Host:8e854ee0-ca64-4642-9cea-527dfb1b19a8:</id>
    <link href="/compute/hosts/urn:storageos:Host:8e854ee0-
ca64-4642-9cea-527dfb1b19a8;" rel="self"/>
    <name>lglba181.lss.emc.com</name>
</host>
<host>
    <id>urn:storageos:Host:27dab0f2-70cc-4bbe-81ef-a9b66ddb14ca:</id>
    <link href="/compute/hosts/urn:storageos:Host:
27dab0f2-70cc-4bbe-81ef-a9b66ddb14ca;" rel="self"/>
    <name>lglba241.lss.emc.com</name>
</host>
<host>
    <id>urn:storageos:Host:8ea2ef78-a072-41df-ada9-951433cc6464:</id>
    <link href="/compute/hosts/urn:storageos:Host:8ea2ef78-a072-41df-
ada9-951433cc6464;" rel="self"/>
    <name>lglba046.lss.emc.com</name>
</host>
<host>
    <id>urn:storageos:Host:807d346a-3b47-4412-9ab5-8f516e792b71:</id>
    <link href="/compute/hosts/urn:storageos:Host:
807d346a-3b47-4412-9ab5-8f516e792b71;" rel="self"/>
    <name>lglba243.lss.emc.com</name>
</host>
<host>
    <id>urn:storageos:Host:e7f67b93-8090-4316-a800-7f826d415bb8:</id>
    <link href="/compute/hosts/urn:storageos:Host:e7f67b93-8090-4316-
a800-7f826d415bb8;" rel="self"/>
    <name>lglba242.lss.emc.com</name>
</host>
</hosts>
```

Listing initiator ports for a host in a vCenter Request

```
GET https://<ViPR_VIP>:4443/compute/hosts/urn:storageos:Host:8ea2ef78-
a072-41df-ada9-951433cc6464:/initiators
```

Response

```
HTTP 200 OK
<initiators>
    <initiator>
        <id>urn:storageos:Initiator:97a5bfd2-5981-401c-
ba52-3bf002a63470:</id>
        <link href="/compute/initiators/urn:storageos:Initiator:
97a5bfd2-5981-401c-ba52-3bf002a63470;" rel="self"/>
        <name>iqn.1998-01.com.vmware:localhost:1161951551:36</name>
    </initiator>
    <initiator>
        <id>urn:storageos:Initiator:b70c8916-
e415-4f71-9e46-34fd73c9488f:</id>
        <link href="/compute/initiators/urn:storageos:Initiator:b70c8916-
e415-4f71-9e46-34fd73c9488f;" rel="self"/>
        <name>iqn.1998-01.com.vmware:localhost:1161951551:34</name>
    </initiator>
    <initiator>
        <id>urn:storageos:Initiator:ebe54432-3bba-4bf6-
a5b4-0af274d96b53:</id>
        <link href="/compute/initiators/
urn:storageos:Initiator:ebe54432-3bba-4bf6-a5b4-0af274d96b53;" rel="self"/>
```

```
<name>iqn.1998-01.com.vmware:localhost:1161951551:35</name>
</initiator>
<initiator>
  <id>urn:storageos:Initiator:7188e4cb-6b58-4815-
a1c7-9a1b2684f08d:</id>
  <link href="/compute/initiators/urn:storageos:Initiator:
7188e4cb-6b58-4815-a1c7-9a1b2684f08d:" rel="self"/>
    <name>iqn.1998-01.com.vmware:localhost:1161951551:33</name>
  </initiator>
  <initiator>
    <id>urn:storageos:Initiator:ce9cee8f-
fec8-4aac-8233-1256ad34d57c:</id>
    <link href="/compute/initiators/urn:storageos:Initiator:ce9cee8f-
fec8-4aac-8233-1256ad34d57c:" rel="self"/>
      <name>10:00:00:00:C9:B8:2D:30</name>
    </initiator>
    <initiator>
      <id>urn:storageos:Initiator:
514381ab-5555-4183-9467-8172b4a0d7fa:</id>
      <link href="/compute/initiators/urn:storageos:Initiator:
514381ab-5555-4183-9467-8172b4a0d7fa:" rel="self"/>
        <name>10:00:00:00:C9:B8:2D:31</name>
      </initiator>
    </initiator>
  </initiators>
```


CHAPTER 11

Discovering Storage Systems

This chapter contains the following topics

- ◆ [Registering and discovering physical storage systems in VNX Block and VMAX](#).....138
- ◆ [Registering and discovering a VPLEX](#).....139
- ◆ [Registering and discovering an Isilon storage system](#).....140
- ◆ [Registering and discovering a NetApp storage system](#).....142
- ◆ [Registering and discovering a VNX File storage system](#).....143

Registering and discovering physical storage systems in VNX Block and VMAX

For VNX Block and VMAX, registering an SMI-S provider provides access to the storage systems. Once the SMI-S providers are registered, then the underlying storage systems, storage pools, and storage ports are discovered automatically without any further user input.

Before you begin

You need the following information:

- ◆ The IP address of the SMI-S provider.
- ◆ The username and password for connecting to the SMI-S provider.
- ◆ The port number used to connect to the SMI-S provider.

Procedure

1. Register the SMI-S Provider by sending a POST to /vdc/smisi-providers.

The request returns a task whose URI can be queried to determine the status of the task such as completion state, any failures, and the start time of the task request.

In this example, the SMI-S Provider with the IP address of 192.168.0.0 is being registered.

Request

```
POST https://<ViPR_VIP>:4443/vdc/smisi-providers
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}' ,
'ACCEPT': 'application/json'}
{
    "ip_address": "192.168.0.0",
    "name": "mysmis",
    "password": "#1Password",
    "port_number": "5988",
    "use_ssl": "false",
    "user_name": "admin"
}
```

Response

```
HTTP 202 Accepted
{
    "link": {
        "href": "/vdc/smisi-providers/
urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:/tasks/e393ece4-44eb-42d2-9d8a-33e28b436daf",
        "rel": "self"
    },
    "op_id": "e393ece4-44eb-42d2-9d8a-33e28b436daf",
    "resource": {
        "id": "urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
        "link": {
            "href": "/vdc/smisi-providers/
urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
            "rel": "self"
        },
        "name": "mysmis"
    },
    "start_time": 1379202542423,
    "state": "pending"
}
```

2. Repeat the query of the SMI-S registration task, using the task URL from the response body of the POST request, until the message attribute of the task is Operation completed successfully.

Request

```
GET https://<ViPR_URL>:4443/vdc/smis-providers/
urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:/tasks/e393ece4-44eb-42d2-9d8a-33e28b436daf
```

```
{'Content-Type': 'application/json',
'X-SDS-AUTH-TOKEN': '{Token_Text}',
'ACCEPT': 'application/json'}
```

Response

```
HTTP 200 OK
{
    "end_time": 1379202543299,
    "link": {
        "href": "/vdc/smis-providers/
urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:/tasks/e393ece4-44eb-42d2-9d8a-33e28b436daf",
        "rel": "self"
    },
    "message": "Operation completed successfully",
    "op_id": "e393ece4-44eb-42d2-9d8a-33e28b436daf",
    "resource": {
        "id": "urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
        "link": {
            "href": "/vdc/smis-providers/
urn:storageos:SMISProvider:dce4a4c4-334b-4353-9484-c7b225f413d5:",
            "rel": "self"
        },
        "name": "mysmis"
    },
    "start_time": 1379202542423,
    "state": "ready"
}
```

Registering and discovering a VPLEX

When you add a VPLEX to ViPR, the array is automatically added to the ViPR discovery queue. If the credentials are correct, the array is automatically discovered.

Before you begin

You need the following information:

- ◆ The IP address of the storage system.
- ◆ The username and password for connecting to the storage system.
- ◆ The port number used to connect to the storage system.

Procedure

1. Register a VPLEX by sending a POST /vdc/storage-systems request.

The request returns a task whose URI can be queried to determine the status of the task such as completion, any failures, time of task request, and time of task completion.

Request

```
POST https://<ViPR_VIP>:4443/vdc/storage-systems
{
    "ip_address": "192.168.0.0",
    "name": "VPLEX_MET",
    "password": "Mi@Dim7T",
```

```

        "port_number": "443",
        "system_type": "vplex",
        "user_name": "service"
    }
}

Response
202 Accepted
<task>
    <op_id>d6860996-be16-42f8-8d81-e35b0ba63fc9</op_id>
    <resource>
        <id>urn:storageos:StorageSystem:75b2adfb-47c9-45d6-
a5be-76e400e328fb:</id>
        <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:75b2adfb-47c9-45d6-a5be-76e400e328fb:" rel="self" />
        <name>VPLEX_MET</name>
    </resource>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
75b2adfb-47c9-45d6-a5be-76e400e328fb:/tasks/d6860996-
be16-42f8-8d81-e35b0ba63fc9" rel="self" />
    <start_time>1386301587343</start_time>
    <state>pending</state>
</task>

```

2. Repeat the query of the VPLEX create task, using the task URL from the response body of the POST request, until the message attribute of the task is Operation completed successfully.

Request

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems/
urn:storageos:StorageSystem:75b2adfb-47c9-45d6-a5be-76e400e328fb:/
tasks/d6860996-be16-42f8-8d81-e35b0ba63fc9
```

Response

```

<task>
    <end_time>1386301599628</end_time>
    <message>Operation completed successfully</message>
    <op_id>d6860996-be16-42f8-8d81-e35b0ba63fc9</op_id>
    <resource>
        <id>urn:storageos:StorageSystem:75b2adfb-47c9-45d6-
a5be-76e400e328fb:</id>
        <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:75b2adfb-47c9-45d6-a5be-76e400e328fb:" rel="self" />
        <name>VPLEX_MET</name>
    </resource>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
75b2adfb-47c9-45d6-a5be-76e400e328fb:/tasks/d6860996-
be16-42f8-8d81-e35b0ba63fc9" rel="self" />
    <start_time>1386301587343</start_time>
    <state>ready</state>
</task>
```

Registering and discovering an Isilon storage system

When you add an Isilon to ViPR, the array is automatically added to the ViPR discovery queue. If the credentials are correct, the array is automatically discovered.

Before you begin

You need the following information:

- ◆ System Type - Isilon.
- ◆ SmartConnect IP Address - Direct the discovery request to the Isilon SmartConnect host IP.

- ◆ Port Number - 8080.
- ◆ user_name - name of the root account on the Isilon array.
- ◆ password - password of the root account on the Isilon array.

Procedure

1. Register an Isilon by sending a POST /vdc/storage-systems request.

The request returns a task whose URI can be queried to determine the status of the task such as completion state, any failures, and the time of the task request.

Request

```
POST https://<ViPR_VIP>:4443/vdc/storage-systems
```

```
<storage_system_create>
  <name>isilon1</name>
  <system_type>isilon</system_type>
  <ip_address>192.168.0.0</ip_address>
  <port_number>8080</port_number>
  <user_name>rootid</user_name>
  <password>password</password>
</storage_system_create>
```

Response

202 Accepted

```
<task>
  <associated_resources/>
  <op_id>63719ec8-211e-41ae-a0a6-2a7a9c3a97e8</op_id>
  <resource>
    <id>urn:storageos:StorageSystem:473d0990-742b-4035-
b360-933f3f189ba8:</id>
    <link href="/vdc/storage-systems/{StorageSystem_urn}">
      <rel="self"/>
      <name>isilon1</name>
    </resource>
    <link href="/vdc/storage-systems/{StorageSystem_urn}/tasks/
63719ec8-211e-41ae-a0a6-2a7a9c3a97e8" rel="self"/>
      <start_time>1387486656968</start_time>
      <state>pending</state>
    </link>
  </resource>
</task>
```

2. Query the Isilon create task, using the task URL from the response body of the POST request, until the state attribute of the task is ready which indicates that the operation has completed.

Request

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems/
{StorageSystem_urn}/tasks/63719ec8-211e-41ae-a0a6-2a7a9c3a97e8
```

Response

```
<task>
  <associated_resources/>
  <end_time>1387486658727</end_time>
  <message>Operation completed successfully</message>
  <op_id>63719ec8-211e-41ae-a0a6-2a7a9c3a97e8</op_id>
  <resource>
    <id>urn:storageos:StorageSystem:473d0990-742b-4035-
b360-933f3f189ba8:</id>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
473d0990-742b-4035-b360-933f3f189ba8:" rel="self"/>
      <name>isilon1</name>
    </resource>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
473d0990-742b-4035-b360-933f3f189ba8:/tasks/63719ec8-211e-41ae-
a0a6-2a7a9c3a97e8" rel="self"/>
      <start_time>1387486656968</start_time>
```

```

<state>ready</state>
</task>
```

Once the task is complete, this storage system is automatically discovered by ViPR.

Registering and discovering a NetApp storage system

When you add an NetApp to ViPR, the array is automatically added to the ViPR discovery queue. If the credentials are correct, the array is automatically discovered.

Before you begin

To successfully complete this procedure, you need the following information:

- ◆ System Type - netapp.
- ◆ IP Address - Direct the discovery request to the NetApp OnTap IP.
- ◆ Port Number - 443.
- ◆ user_name - name of the root account on the NetApp array.
- ◆ password - password of the root account on the NetApp array.

Procedure

1. Register the NetApp by sending a POST /vdc/storage-systems request. The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/vdc/storage-systems
```

```

<storage_system_create>
  <name>NetApp2</name>
  <system_type>netapp</system_type>
  <ip_address>192.168.0.0</ip_address>
  <port_number>443</port_number>
  <user_name>root</user_name>
  <password>dangerous1</password>
</storage_system_create>
```

Response

202 Accepted

```

<task>
  <op_id>937200f1-16ef-4bd3-8c13-b79e6651f967</op_id>
  <resource>
    <id>urn:storageos:StorageSystem:b4c6efa0-
bf8b-416a-9c01-5fc0a19f27a8:</id>
    <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:b4c6efa0-bf8b-416a-9c01-5fc0a19f27a8:" rel="self" />
    <name>ISI_RAH</name>
  </resource>
  <link href="/vdc/storage-systems/
urn:storageos:StorageSystem:b4c6efa0-
bf8b-416a-9c01-5fc0a19f27a8:/tasks/937200f1-16ef-4bd3-8c13-
b79e6651f967" rel="self" />
  <start_time>1386303328118</start_time>
  <state>pending</state>
</task>
```

2. Query the NetApp create task, using the task URL from the response body of the POST request. When the state attribute of the task is ready, the operation has completed.

Request

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems/
{StorageSystem_urn}/tasks/937200f1-16ef-4bd3-8c13-b79e6651f967
```

Response

```
<task>
  <associated_resources/>
  <end_time>1387489906247</end_time>
  <message>Operation completed successfully</message>
  <op_id>937200f1-16ef-4bd3-8c13-b79e6651f967</op_id>
  <resource>
    <id>urn:storageos:StorageSystem:0dfd2dfc-
e097-4af4-94f2-5e08f41a6fad:</id>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
0dfd2dfc-e097-4af4-94f2-5e08f41a6fad:" rel="self"/>
      <name>NetApp2</name>
    </resource>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
0dfd2dfc-e097-4af4-94f2-5e08f41a6fad:/tasks/ba2b50b3-b5ad-44a7-
bbfd-bd51dc58f9d9" rel="self"/>
      <start_time>1387489905274</start_time>
      <state>ready</state>
  </task>
```

Registering and discovering a VNX File storage system

When you add an VNX File storage system to ViPR, the array is automatically added to the ViPR discovery queue. If the credentials are correct, the array is automatically discovered.

Before you begin

VNX File arrays are discovered by providing two sets of credentials:

- ◆ A user name and password for the VNX File Control Station
- ◆ A user name and password for the on-board SMI-S provider.

The following table shows the complete set of information you need to successfully discover a VNX File array.

Requirement	Description
System type	vnxfile
IP address	The IP address of the VNX File control station.
Port number	443
user_name	An admin user for the VNX File control station
password	A password for the VNX File control station admin user
SMI-S provider IP address	The IP address of the on-board SMI-S provider
SMI-S user_name	An admin user for the SMI-S provider
SMI-S password	A password for the SMI-S provider
smi-s port number	5989
smi-s_user_ssl	true

Procedure

1. Register a VNX File system by sending a POST /vdc/storage-systems request.
The request returns a task whose URI can be queried to determine the status of the task.

Request

```
POST https://<ViPR_VIP>:4443/vdc/storage-systems
```

```
<storage_system_create>
  <name>VNXFile1</name>
  <system_type>vnxfile</system_type>
  <ip_address>192.168.0.0</ip_address>
  <port_number>443</port_number>
  <user_name>nasadmin</user_name>
  <password>nasadmin</password>
  <smis_provider_ip>10.247.71.61</smis_provider_ip>
  <smis_port_number>5989</smis_port_number>
  <smis_user_name>admin</smis_user_name>
  <smis_password>#1Password</smis_password>
  <smis_use_ssl>true</smis_use_ssl>
</storage_system_create>
```

Response

202 Accepted

```
<task>
  <associated_resources/>
  <op_id>d036da87-2a93-4db2-9c34-c971945264ed</op_id>
  <resource>
    <id>urn:storageos:StorageSystem:1cab66bf-46fd-4953-
a527-2dc80797bf7d:</id>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
1cab66bf-46fd-4953-a527-2dc80797bf7d:" rel="self"/>
    <name>VNXFile1</name>
  </resource>
  <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
1cab66bf-46fd-4953-a527-2dc80797bf7d:/tasks/
d036da87-2a93-4db2-9c34-c971945264ed" rel="self"/>
    <start_time>1387568454683</start_time>
    <state>pending</state>
  </task>
```

2. Query the VNX File create task, using the task URL from the response body of the POST request. When the state attribute of the task is ready, the operation has completed.

Request

```
GET https://<ViPR_VIP>:4443/vdc/storage-systems/
{StorageSystem_urn}/d036da87-2a93-4db2-9c34-c971945264ed
```

Response

```
<task>
  <associated_resources/>
  <end_time>1387568504400</end_time>
  <message>Operation completed successfully</message>
  <op_id>d036da87-2a93-4db2-9c34-c971945264ed</op_id>
  <resource>
    <id>urn:storageos:StorageSystem:1cab66bf-46fd-4953-
a527-2dc80797bf7d:</id>
    <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
1cab66bf-46fd-4953-a527-2dc80797bf7d:" rel="self"/>
    <name>VNXFile1</name>
  </resource>
  <link href="/vdc/storage-systems/urn:storageos:StorageSystem:
1cab66bf-46fd-4953-a527-2dc80797bf7d:/tasks/
d036da87-2a93-4db2-9c34-c971945264ed" rel="self"/>
    <start_time>1387568454683</start_time>
    <state>ready</state>
  </task>
```

CHAPTER 12

User Interface Services

This chapter contains the following topics.

◆ Catalog, Services, Orders, and Approvals.....	146
◆ Accessing UI services.....	148
◆ UI Services response formats.....	148
◆ Approval requests.....	149
◆ Approving or rejecting.....	149
◆ Retrieving a list of pending approvals.....	150
◆ URL notification.....	150
◆ Asset options.....	151
◆ Upstream values.....	151
◆ Executing a service.....	153
◆ Service catalog.....	154
◆ Service descriptors.....	156
◆ Field descriptors.....	156
◆ Field validation.....	157
◆ Type field values.....	157
◆ Tracking orders.....	158

Catalog, Services, Orders, and Approvals

You can perform self-service provisioning by executing one of the services in the service catalog.

Using the ViPR UI, an administrator can customize the services available in the service catalog, or create new services based on the existing services. New services, such as Create Block Storage, can be added to existing categories or new categories can be defined, such as Block Storage Services. The administrator can assign ACLs to the categories and to the services themselves in order to restrict the ability of users to view services and execute them. The administrator can also set up the service so that all requests must be approved prior to processing.

Once the Service Catalog has been set up, clients can perform self-service provisioning by executing one of the services in the service catalog. A client can browse the service catalog to find services. Once the appropriate service has been found, the service descriptor for that service is retrieved. The client can then send a request, including the necessary form-encoded parameters from the service descriptor, to execute the service. When a service execution is requested, an Order is created. If the administrator has configured the service so that it requires approval, then an approval request is sent to the appropriate approvers. Once approval is granted, the order proceeds and is executed.

The following table shows some API calls that are used to browse the catalog, request services, and track orders and approvals.

Note

These calls are all accessed on HTTP port 443.

ViPR REST API Call	Description
GET /api/catalog	Browse the service catalog from the root (or Home).
GET /api/catalog/{sp1}/{sp2}/{sp3}/{sp4}/{sp5}	Show the details of the service which matches the specified path. The path can be up to five catalog or service names. For example, /api/catalog/BlockStorageServices/RemoveBlockStorageForHost
GET /api/categories/{categoryID}	Show the contents of the category.
GET /api/services/{serviceID}	Show the contents of a service.
POST /api/services/{serviceID}/descriptor	Retrieve the service descriptor for the catalog service. A JSON file is returned. It contains the information required to create a form that specifies the payload parameters to be sent with a request to execute the service.
POST /api/services/{serviceID}	Execute a service by ID. The request includes parameters defined in the service descriptor. For example, to create a volume you would include parameters similar to: virtualArray:urn:storageos:VirtualArray: 58528ef1-149e-40c4-8732-

ViPR REST API Call	Description
	<pre>df3ddf108a22: virtualPool:urn:storageos:VirtualPool: 7aa0c6f0-5f47-423b-9bleeda98c838bc7: project:urn:storageos:Project:b161965b-4fb1-439e-8224- ed47840487e0: name:ViPR-Test2 consistencyGroup: numberOfVolumes:1 size:5</pre> <p>The service parameters in the payload must be form encoded using the content type: application/x-www-form-urlencoded.</p>
POST /api/catalog/{sp1}/{sp2}/{sp3}/{sp4}/{sp5}	<p>Execute the service matching the specified path.</p> <p>The request includes parameters defined in the service descriptor. For example, to create a volume you would include parameters similar to:</p> <pre>virtualArray:urn:storageos:VirtualArray: 58528ef1-149e-40c4-8732- df3ddf108a22: virtualPool:urn:storageos:VirtualPool: 7aa0c6f0-5f47-423b-9bleeda98c838bc7: project:urn:storageos:Project:b161965b-4fb1-439e-8224- ed47840487e0: name:ViPR-Test2 consistencyGroup: numberOfVolumes:1 size:5</pre> <p>The service parameters in the payload must be form encoded using the content type: application/x-www-form-urlencoded.</p>
GET /api/orders	Returns a list of IDs for orders which you own.
GET /api/orders/all	Returns a list of IDs for all orders in the current tenant. You must have the Tenant Administrator role.
GET /api/orders/{orderID}	Show the details of a specific order. You must be the owner of the order or have the Tenant Administrator or Tenant Approval role.
GET /api/orders/{orderId}/execution	Show the execution information for an order, including all log information.
GET /api/approvals	List all approvals for the tenant of which you are a member.
GET /api/approvals/pending	List all approvals that are in the pending state.
GET /api/approvals/{approvalID}	Show the details for the specified approval request.
POST /api/approvals/{approvalID}/approve	Approve the specified approval request
POST /api/approvals/	Reject the specified approval request

ViPR REST API Call	Description
{approvalID}/reject	
GET /api/options/{asset}	<p>Retrieve options for an asset type.</p> <p>An asset type is specified in a service descriptor field as "type": "assetType.xxxx.xxxx" and indicates that this is not a fixed value but something that must be retrieved from the server.</p> <p>For example, if the URN of the project is required to execute a service request, the form would show "type": "assetType.bourne.project".</p> <p>Sending GET /api/options/bourne.project returns the IDs of all ViPR projects.</p>
GET /api/options/{asset}/dependencies	Retrieve asset dependencies.

Accessing UI services

All UI Services are available under the /api context. To access them the client must be authenticated against ViPR and an X-SDS-AUTH-TOKEN provided.

[Authentication on page 7](#) describes how to obtain a token.

Note

Unlike other parts of the ViPR API, the UI services are accessed on HTTP port 443, and not 4443.

UI Services response formats

The UI Services use both XML and JSON payloads. If no format is specified, the JSON format is used by default.

Note that Service Descriptors are always in JSON format.

The HTTP Accept header is used to determine the response format with a value of either application/xml or application/json for both xml and json respectively.

```
GET https://<ViPR_VIP>:443/api/orders
X-SDS-AUTH-TOKEN: <AUTH_TOKEN>
Accept: application/json
```

returns

```
[
  {
    "id": "urn:storageos:Order:3184068d-0b88-447e-a899-16d412c21309:",
    "href": "/api/orders/urn:storageos:Order:3184068d-0b88-447e-
a899-16d412c21309:"
  },
  {
    "id": "urn:storageos:Order:34a028a8-b2b4-43e8-81af-5442ba97dc6b:",
    "href": "/api/orders/urn:storageos:Order:34a028a8-
b2b4-43e8-81af-5442ba97dc6b:"}
```

```

        }
    ]

GET https://<ViPR_VIP>:443/api/orders
X-SDS-AUTH-TOKEN: <AUTH_TOKEN>
Accept: application/xml

returns

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<list>
    <reference>
        <href>/api/orders/urn:storageos:Order:3184068d-0b88-447e-
a899-16d412c21309:</href>
        <id>urn:storageos:Order:3184068d-0b88-447e-
a899-16d412c21309:</id>
    </reference>
    <reference>
        <href>/api/orders/urn:storageos:Order:34a028a8-
b2b4-43e8-81af-5442ba97dc6b:</href>
        <id>urn:storageos:Order:34a028a8-
b2b4-43e8-81af-5442ba97dc6b:</id>
    </reference>
</list>

```

Approval requests

You can set up a service in the Service Catalog to require approval before being processed. These Approval Requests can be listed and actioned using either the UI or the Approval Service.

Note that to make any calls against the Approval Service, the authenticated user must have the TENANT_APPROVER role.

Approving or rejecting

Approval Requests can be either approved or rejected using POST /api/approvals/<ApprovalRequest_ID>/approve and /reject.

```
POST https://<ViPR_VIP>:443/api/approvals/<ApprovalRequest_ID>/
approve
POST https://<ViPR_VIP>:443/api/approvals/<ApprovalRequest_ID>/reject
```

The following shows a response to an Approval Request being rejected.

```
{
    "dateActioned": "Oct 2, 2013 4:42:55 PM",
    "status": "REJECTED",
    "approvedBy": "administrator@corp.sean.com",
    "order": {
        "id": "urn:storageos:Order:7f9e7e1f-d765-498c-a320-5840ade14dal:",
        "href": "/api/orders/urn:storageos:Order:7f9e7e1f-d765-498c-
a320-5840ade14dal:",
    },
    "tenant": "urn:storageos:TenantOrg:66e1c6ce-d133-42bb-b90f-
f7c3334599bb:",
    "id": "urn:storageos:ApprovalRequest:ad0f1973-514f-42e9-8186-
ebd121198e16:",
    "inactive": false,
    "link": {
        "rel": "self",
        "href": "/api/approvals/
urn:storageos:ApprovalRequest:ad0f1973-514f-42e9-8186-ebd121198e16:"
    }
}
```

Retrieving a list of pending approvals

A list of pending approvals for the authenticated user can be retrieved with GET /api/approvals/pending.

```
GET https://<ViPR_VIP>:443/api/approvals/pending
```

returns a list of ids for approval requests that require approval:

```
[
  {
    "id": "urn:storageos:ApprovalRequest:ad0f1973-514f-42e9-8186-
ebd121198e16:",
    "href": "/api/approvals/
urn:storageos:ApprovalRequest:ad0f1973-514f-42e9-8186-ebd121198e16:"
  }
]
```

An individual Approval Request can then be retrieved with

```
GET https://<ViPR_VIP>:443/api/approvals/<ApprovalRequest_ID>
```

which returns

```
{
  "status": "PENDING",
  "order": {
    "id": "urn:storageos:Order:7f9e7elf-d765-498c-a320-5840ade14dal:",
    "href": "/api/orders/urn:storageos:Order:7f9e7elf-d765-498c-
a320-5840ade14dal:"
  },
  "tenant": "urn:storageos:TenantOrg:66e1c6ce-d133-42bb-b90f-
f7c3334599bb:",
  "id": "urn:storageos:ApprovalRequest:ad0f1973-514f-42e9-8186-
ebd121198e16:",
  "inactive": false,
  "link": {
    "rel": "self",
    "href": "/api/approvals/
urn:storageos:ApprovalRequest:ad0f1973-514f-42e9-8186-ebd121198e16:"
  }
}
```

URL notification

Ordinarily when an Approval Request is created ViPR will send an email to the approvers. You can however also setup ViPR to call a URL by providing a URL in the UI **Admin > Tenant > Approval Settings** page. The URL can be any address visible to the ViPR cluster.

The following shows an example of the JSON Approval Information resource that ViPR would send to the specified URL.

```
{
  "message":null,
  "dateActioned":null,
  "status":"PENDING",
  "approvedBy":null,
  "order":{
    "id": "urn:storageos:Order:2792d568-034a-42d0-b0da-f39b32fa9074:",
    "href": "/api/orders/urn:storageos:Order:2792d568-034a-42d0-b0da-
f39b32fa9074:"
  },
  "tenant": "urn:storageos:TenantOrg:66e1c6ce-d133-42bb-b90f-
f7c3334599bb:",
  "id": "urn:storageos:ApprovalRequest:739ff36b-ce89-41ff-
a22e-5c572f861da7:",
```

```

    "inactive":false,
    "link":{
      "rel":"self",
      "href":"/api/approvals/urn:storageos:ApprovalRequest:739ff36b-
ce89-41ff-a22e-5c572f861da7:"
    }
  }
}

```

Asset options

There is a special field type that starts with AssetType.XXX. This indicates that options for that field are not supplied statically with the form, but should be retrieved from the server.

To retrieve values for an assetType you call the AssetOption API using the name of the asset (the part after "AssetType."). So for example, to get the name of all hosts for the type AssetType.bourne.hosts, you would call:

```
GET https://<ViPR_VIP>:443/api/options/bourne.hosts
```

The response returned contains a JSON array of key value pairs; the key being the value which should be sent to ViPR as the field value when executing the service, and the value being the display label for that option.

```
[
  {
    "key": "urn:storageos:Host:4c4b7da7-61ee-4384-
afb9-87e1fa3d045b:",
    "value": "linux2"
  },
  {
    "key": "urn:storageos:Host:4c4b7da7-61ee-9783-
afb9-87e1fa3d045b:",
    "value": "linux1"
  }
]
```

Upstream values

Some asset option fields require previous options in order to determine possible valid options. For example, the "Create Block Volume for a Host" service requires that the user select a protocol so that it can display only the hosts that support that particular protocol.

In order to support this, the Asset Option API requires that you pass all current field values that have been selected when asking for the options. This is why the ordering of the field descriptors in the Service Descriptor is important.

For example, the "Export Volume To Host" service includes the following fields :

- ◆ Block Protocol (bourne.blockProtocol)
- ◆ Project (bourne.project)
- ◆ Host (bourne.host)
- ◆ Volume (bourne.unassignedBlockVolume)

The first call will be made against the first field, Block Protocol:

```
GET https://<ViPR_VIP>:443/api/options/bourne.blockProtocol
```

which returns

```
[
  {
    "key": "FC",
```

```

        "value": "FC"
    },
{
    "key": "iSCSI",
    "value": "iSCSI"
}
]
```

The next call will be to retrieve options for the `bourne.Project` field. Note that we must pass in all proceeding values that have been chosen so far.

```
GET https://<ViPR_VIP>:443/api/options/bourne.project?
bourne.blockProtocol=FC
```

Returns the following options

```
[
{
    "key": "urn:storageos:Project:b161965b-4fb1-439e-8224-
ed47840487e0:",
    "value": "ViPR Project 1"
}]
```

We then pass the two values so far when getting the `bourne.Host` field:

```
GET https://<ViPR_VIP>:443/api/options/bourne.host?
bourne.blockProtocol=FC&bourne.project=urn:storageos:Project:b161965b-
4fb1-439e-8224-ed47840487e0:
```

Returns the following options:

```
[
{
    "key": "urn:storageos:Host:4c4b7da7-61ee-4384-afb9-87e1fa3d045b:",
    "value": "windows-vipr1"
},
{
    "key": "urn:storageos:Host:b8006b82-b60c-433b-abda-ef533e60c7cb:",
    "value": "suse-vipr1"
}]
```

Lastly we can get the options for `unassignedBlockVolume`. Note that this example is on separate lines for readability, however it should all be submitted as a single line.

```
GET https://<ViPR_VIP>:443/api/options/bourne.unassignedBlockVolume
?bourne.host=urn:storageos:Host:4c4b7da7-61ee-4384-
afb9-87e1fa3d045b:
&bourne.project=urn:storageos:Project:b161965b-4fb1-439e-8224-
ed47840487e0:
&bourne.blockProtocol=FC
```

Which returns

```
[
{
    "key": "urn:storageos:Volume:39149c88-46ea-4aec-
ac51-46bd58a01e5c:",
    "value": "ViPR-TestVolume-1"
}]
```

If you attempt to get the `bourne.unassignedBlockVolume` option without passing in the previous options, a 400 Bad Request is returned indicating that more options are required before the list of options can be calculated :

```
400 : Bad Request : Query for Asset 'bourne.unassignedBlockVolume'
requires additional asset dependencies. Supplied: []
```

Executing a service

Services are executed by POSTing the required service values to the service as form-urlencoded.

Use the following URL path:

```
POST https://<ViPR_VIP>:443/api/services/<serviceId>
```

For example, in order to execute the Create Volume service you would post the following parameters:

```
virtualArray:urn:storageos:VirtualArray:58528ef1-149e-40c4-8732-
df3ddf108a22:
virtualPool:urn:storageos:VirtualPool:7aa0c6f0-5f47-423b-9b1e-
eda98c838bc7:
project:urn:storageos:Project:b161965b-4fb1-439e-8224-ed47840487e0:
name:ViPR-Test2
consistencyGroup:
numberOfVolumes:1
size:5
```

These would be URL encoded as the body of the request, with the following being sent:

```
POST https://<ViPR_VIP>:443/api/services/
urn:storageos:CatalogService:672fa6b6-b17e-49f3-ad11-b6c43c3680cb:
Accept: application/json
Content-Length: 420
Content-Type: application/x-www-form-urlencoded
X-SDS-AUTH-TOKEN: <AUTH-TOKEN>

virtualArray=urn%3Astorageos%3AVirtualArray%3A58528ef1-149e-40c4-8732-
df3ddf108a22%3A&virtualPool=urn%3Astorageos%3AVirtualPool
%3A7aa0c6f0-5f47-423b-9b1e-eda98c838bc7%3A&project=urn%3Astorageos
%3AProject%3Ab161965b-4fb1-439e-8224-ed47840487e0%3A&name=ViPR-
Test3&consistencyGroup=&numberOfVolumes=1&size=5
size:5
```

If a 200 OK response is returned, the submission has been accepted and a link to the order is returned:

```
{
  "orderNumber": "5",
  "service": {
    "id": "urn:storageos:CatalogService:672fa6b6-b17e-49f3-ad11-
b6c43c3680cb:",
    "href": "/api/services/urn:storageos:CatalogService:672fa6b6-
b17e-49f3-ad11-b6c43c3680cb:"
  },
  "summary": "Create a block volume",
  "message": "",
  "createdDate": "Sep 10, 2013 1:26:24 PM",
  "status": "PENDING",
  "execution": {
    "id": "urn:storageos:Order:34a028a8-b2b4-43e8-81af-5442ba97dc6b:",
    "href": "/api/orders/urn:storageos:Order:34a028a8-
b2b4-43e8-81af-5442ba97dc6b:/execution"
  },
  "id": "urn:storageos:Order:34a028a8-b2b4-43e8-81af-5442ba97dc6b:",
  "inactive": false,
  "link": {
    "rel": "self",
    "href": "/api/orders/urn:storageos:Order:34a028a8-
b2b4-43e8-81af-5442ba97dc6b:"
  }
}
```

Orders are visible in the ViPR Portal when submitted via the API as if they had been submitted directly via the catalog.

If any of the required fields are missing, the Portal API will reject the submission, and return a 400 BAD REQUEST with a JSON resource explaining what's wrong. For example, if the previous request were submitted with the size parameter is missing, the response would be:

```
400 BAD_REQUEST
[
  {
    "field": "size",
    "error": "Required"
  }
]
```

The Portal API will also validate all parameters against the Service Descriptor requirements, and any errors could lead to multiple errors for the same field being returned. If the following example, were submitted the `numberOfVolumes` parameter with a non numeric value of `asd` the response would be:

```
400 BAD_REQUEST
[
  {
    "field": "numberOfVolumes",
    "error": "Must be a valid integer"
  },
  {
    "field": "numberOfVolumes",
    "error": "Cannot be lower than 1"
  },
  {
    "field": "numberOfVolumes",
    "error": "Cannot be greater than 2"
  }
]
```

Service catalog

The Service Catalog API allows a REST client to browse the ViPR Self Service Catalog configured in the ViPR Portal. The catalog itself is structured as a tree of Categories, each Category can contain a list of Sub-Categories and a list of Services in that category.

Information about a category can be retrieved by path using the category titles (with spaces removed) to form a path. For example, the following will retrieve the Block Storage Services → Protection Services category:

```
GET https://<ViPR_VIP>:443/api/catalog/BlockStorageServices/
ProtectionServices
```

The Root (or 'Home') category is retrieved using the following URL (note that there is no / on the end of the URL)

```
GET https://<ViPR_VIP>:443/api/catalog
```

The following shows an example category descriptor returned by ViPR.

```
{
  "name": "BlockStorageServices",
  "title": "Block Storage Services",
  "description": "Block storage services for fibre channel and iSCSI",
  "image": "icon_host.png",
  "subCategories": [
    {
      "name": "Protection",
      "id": "urn:storageos:CatalogCategory:67218dcf-9330-4822-9b67-137f3a0de2b5:",
      "subCategories": [
        {
          "name": "Encryption"
        }
      ]
    }
  ]
}
```

```

        "href": "/api/categories/urn:storageos:CatalogCategory:
67218dcf-9330-4822-9b67-137f3a0de2b5:"
    },
],
"services": [
{
    "name": "CreateBlockVolumeforaHost",
    "title": "Create Block Volume for a Host",
    "description": "Create block volume and export it for a host",
    "image": "icon_array_host_add.png",
    "approvalRequired": false,
    "executionWindowRequired": false,
    "defaultExecutionWindowId":
"urn:storageos:ExecutionWindow:NEXT::NEXT",
    "baseService": "CreateBlockStorageForHost",
    "maxSize": 10,
    "id": "urn:storageos:CatalogService:5ee6e282-bf83-457f-
bc4f-48c3b94e0648:",
    "inactive": false,
    "link": {
        "rel": "self",
        "href": "/api/services/urn:storageos:CatalogService:5ee6e282-
bf83-457f-bc4f-48c3b94e0648:"
    }
},
{
    "name": "RemoveBlockStorageForHost",
    "title": "Remove Volume by Host",
    "description": "Removes an unmounted block volume assigned to a
host and all of its exports. The volume will no longer be available
from any host.",
    "image": "icon_array_host_remove.png",
    "approvalRequired": false,
    "executionWindowRequired": false,
    "baseService": "RemoveBlockStorageForHost",
    "id": "urn:storageos:CatalogService:aef84673-2f98-41f9-
b74b-32c6e75ee7d5:",
    "inactive": false,
    "link": {
        "rel": "self",
        "href": "/api/services/
urn:storageos:CatalogService:aef84673-2f98-41f9-b74b-32c6e75ee7d5:"
    }
}
],
"id": "urn:storageos:CatalogCategory:1ee6d153-5809-45c5-962f-
fc91f255a0d9:",
"inactive": false,
"link": {
    "rel": "self",
    "href": "/api/categories/urn:storageos:CatalogCategory:
1ee6d153-5809-45c5-962f-fc91f255a0d9:"
}
}
]

```

The id and link fields at the end of the descriptor provide identification information about this category.

ViPR also allows you to retrieve the category descriptor directly with the Category ID (shown as the "id":"urn:CatalogCategory:XXXX" values in the above example), using the following URL:

```
GET https://<ViPR_VIP>:443/api/categories/<Catalog_ID>
```

Service descriptors

Each service in the catalog is represented by a service descriptor which provides configuration information about the service as well as a list of field descriptors. The field descriptors tell clients (such as the portal) what fields values are required and any constraints on those fields values.

Service descriptors can be retrieved with the ID of the service (id's starting with urn:storageos:CatalogService:XXXX), using the following URL

```
GET https://<ViPR_VIP>:443/api/services/<service_id>/descriptor
```

The following shows a basic Service descriptor with one field of Virtual Array.

```
{
  "serviceId": "CreateVolume",
  "category": "Block Services",
  "title": "Create Block Volume",
  "description": "Create a Block Volume",
  "roles": [],
  "destructive": false,
  "fields": {
    "virtualArray": {
      "name": "virtualArray",
      "label": "Virtual Array",
      "type": "assetType.bourne.virtualArray",
      "required": true,
      "select": "one",
      "lockable": true,
      "validation": {
        "min": 0,
        "max": 2147483647,
        "regEx": "",
        "failureMessage": ""
      },
      "options": {}
    }
  }
}
```

Field descriptors

Field descriptors describe the fields required by the service in order to perform the requested operation. For example, the Export Volume to Host service requires the user select a host and a target volume.

Each descriptor provides information such as the label, the type of the field, and then any constraints on that field which a UI can use to provide further hints to a user. The following is an example of a field descriptor:

```
"host": {
  "name": "host",
  "label": "Host",
  "type": "assetType.bourne.host",
  "description": "If the host is in a cluster, storage will be allocated to all hosts in the cluster.",
  "required": true,
  "select": "one",
  "lockable": false,
  "validation": {
    "min": 0,
    "max": 2147483647,
    "regEx": "",
    "failureMessage": ""
  }
},
```

```
"options": {}
```

A field descriptor contains the following values:

name	The symbolic name of the field. This is the name by which the field is referenced when executing the service
label	The user readable label for the field
type	The type of the field (see section on field types)
required	True if a value MUST be supplied for the field
select	The type of selection for field types that support it. A value of one indicates only one value can be selected from the options, a value of many indicates one or more values can be selected
lockable	Indicates to the user should have the option of locking this field to a particular value when editing the service
validation	Validation rules that should be applied to the field. See section on validation
options	If the field type is choice, this provides a map of value to label for all the possible choices

Field validation

For text and number field types, the validation resource provides information on the constraints around acceptable values.

Note that the validation resource will always contain all the fields, but only values that make sense against the type should be used.

Field	Applies to type	Description
min	number	The minimum allowed value
number	number	A numerical value
regEx	text	A regular expression which the value must match in order to be considered valid
failureMessage	A message that can be displayed to users if validation fails.	

Type field values

The type field can have one of several possible values.

text	A plain text value
number	A numerical value
choice	A choice of one of the options from the options field of the descriptor
storageSize	A value that represents a storage size

assetType.XXXXX	If the field type begins with assetType, this indicates a dynamic value. See Asset Options for how to retrieve the possible options.
-----------------	--

A field type can have a type that begins with assetType, for example assetType.vipr.host. If the field type begins with assetType it indicates that the field is dynamic and a further call should be made back to ViPR to get possible values. See section on retrieving Asset Type Values, or the AssetOptions API on how to retrieve these values.

Tracking orders

Once a service has been submitted, a link to the order ID is returned, which can be used to retrieve the status of the order.

For example, calling:

```
GET https://<ViPR_VIP>:443/api/orders/urn:storageos:Order:34a028a8-b2b4-43e8-81af-5442ba97dc6b:  
X-SDS-AUTH-TOKEN: <AUTH_TOKEN>
```

would return the summary information for order urn:storageos:Order:34a028a8-b2b4-43e8-81af-5442ba97dc6b::

```
{
    "orderNumber": "5",
    "service": {
        "id": "urn:storageos:CatalogService:672fa6b6-b17e-49f3-ad11-b6c43c3680cb:",
        "href": "/api/services/urn:storageos:CatalogService:672fa6b6-b17e-49f3-ad11-b6c43c3680cb:"
    },
    "summary": "Create a block volume",
    "message": "",
    "createdDate": "Sep 10, 2013 1:26:24 PM",
    "status": "SUCCESS",
    "execution": {
        "id": "urn:storageos:Order:3184068d-0b88-447e-a899-16d412c21309:",
        "href": "/api/orders/urn:storageos:Order:3184068d-0b88-447e-a899-16d412c21309:/execution"
    },
    "id": "urn:storageos:Order:3184068d-0b88-447e-a899-16d412c21309:",
    "inactive": false,
    "link": {
        "rel": "self",
        "href": "/api/orders/urn:storageos:Order:3184068d-0b88-447e-a899-16d412c21309:"
    }
}
```

The status of an order will be one of the following:

PENDING	Order has been accepted, but hasn't started executing.
EXECUTING	The order is currently being processed.
SUCCESS	The order has completed successfully.
ERROR	The order has completed, but with an error.
SCHEDULED	The order has been scheduled in an Execution Window.
APPROVAL	The order requires, and is awaiting approval.
APPROVED	The order has been approved and is awaiting execution.
REJECTED	The order has been rejected by an approver and will not be executed.

CHAPTER 13

Error Codes and HTML Return Codes

This chapter contains the following topics.

- ◆ [Error code descriptions](#) 162
- ◆ [HTTP return codes](#) 163

Error code descriptions

Whenever a HTTP method request encounters an error, the response includes an HTTP return code, an error code, and a description of the error.

Error code	Description	Retryable	HTTP return code
1003	Some parameters are missing or are incorrect	No	400 Bad Request
1004	This operation is not authorized for this resource using the specified credentials	No	401 Unauthorized
1005	This operation in this resource is forbidden	No	403 Forbidden
1006	Unable to find a suitable placement to handle the request	No	500 Internal Server Error
1007	Method not allowed	No	405 Method not allowed
1008	An error occurred while finding a suitable placement to handle the request	No	500 Internal Server Error
2000	An error occurred in the database	No	500 Internal Server Error
2001	Unable to find the object with the given id	No	404 Not Found
2002	The object has been marked as inactive for deletion	No	500 Internal Server Error
2003	Unable to connect to the database service	Yes	503 Service Unavailable
4000	An error occurred in the controller	No	500 Internal Server Error
4001	error occurred in the network controller	No	500 Internal Server Error
4002	An error occurred in the storage controller	No	500 Internal Server Error
4003	An error occurred in the object controller	No	500 Internal Server Error
4004	Unable to find a suitable controller to handle the request	No	500 Internal Server Error
4005	An error occurred executing a step in the controller workflow	No	500 Internal Server Error
5000	A general error occurred in the Isilon node	No	500 Internal Server Error
5001	An error occurred communicating with the Isilon node using REST	No	500 Internal Server Error
5002	Unable to contact Isilon node	Yes	503 Service Unavailable

Error code	Description	Retryable	HTTP return code
6000	An error occurred in the coordinator	No	500 Internal Server Error
6001	The coordinator was unable to locate the service	Yes	503 Service Unavailable
6002	Unable to queue the job, we have reached the maximum capacity for this queue	Yes	503 Service Unavailable
6003	Unable to queue the job	No	500 Internal Server Error
10000	Unable to connect to the service. The service is unavailable, try again later	Yes	503 Service Unavailable
10001	An unexpected error occurred, please check the ViPR logs for more information	No	500 Internal Server Error

HTTP return codes

Whenever a HTTP method request is sent to the API, the response includes one of the following HTTP codes.

Response code	Description
200 OK	Standard response for successful HTTP requests. The actual response depends on the request method used: <ul style="list-style-type: none"> In a GET request, the response contains an entity corresponding to the requested resource. In a POST request, the response contains an entity describing or containing the result of the action.
201 Created	The requested entity has been created and can be found at the URL specified in the Location header.
202 Accepted	The request has been accepted for processing, but the processing has not been completed. A task has been created for the request. The response provides a task URI in the Location header and usually has a task document in the response body.
204 No Content	The request is valid and has been completed. The response includes no document body.
301 Moved Permanently	This and all future requests should be directed to the given URL.
302 Found	The redirected URL has been found.
400 Bad Request	The request contains bad syntax or cannot be fulfilled.
401 Unauthorized	An Authorization header was expected, but not found. The response includes a WWW-Authenticate header field containing a challenge applicable to the requested resource.
403 Forbidden	The client does not have adequate privileges to access one or more resources specified in the request.

Response code	Description
404 Not Found	The requested resource could not be found.
405 Method Not Allowed	A request was made of a resource using a request method not supported by that resource. For example, a PUT on a read-only resource.
406 Non Acceptable	The requested resource is incapable of generating content as specified in the Accept headers sent in the request, or the version set in the X-UIM-Accept-Version header of the request is not supported by the resource.
500 Internal Server Error	The request could not be completed due to an internal error at the server.
501 Not Implemented	The server either does not recognize the request method, or it lacks the ability to fulfill the request.
503 Service Unavailable	Services that are needed to complete the request on the server are currently unavailable. This state is usually temporary.