Red Hat Enterprise Virtualization 3.0

REST API Guide

Using the Red Hat Enterprise Virtualization Representational State Transfer Application Programming Interface

Red Hat Documentation Team

Daniel Macpherson

David Jorm

Mark McLoughlin

Eoghan Glynn

Red Hat Enterprise Virtualization 3.0 REST API Guide
Using the Red Hat Enterprise Virtualization Representational State Transfer Application Programming Interface
Edition 1

Author	Red Hat Documentation Team	
Author	Daniel Macpherson	*dmacpher@redhat.com*
Author	David Jorm	*djorm@redhat.com*
Author	Mark McLoughlin	*markmc@redhat.com*
Author	Eoghan Glynn	*eglynn@redhat.com*

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701

This document describes the Representational State Transfer (REST) API for Red Hat Enterprise Virtualization.

Preface **vii**
1. About This Guide ... vii
2. Audience ... vii
3. The Red Hat Enterprise Virtualization Documentation Suite ... vii
4. Document Conventions ... viii
4.1. Typographic Conventions ... viii
4.2. Pull-quote Conventions ... ix
4.3. Notes and Warnings ... x
5. Getting Help and Giving Feedback ... xi
5.1. Do You Need Help? ... xi
5.2. We Need Feedback! ... xi
6. Backwards Compatibility Statement ... xi

1. Introduction **1**
1.1. Representational State Transfer ... 1
1.2. Prerequisites ... 1

2. Authentication and Security **3**
2.1. TLS/SSL Certification ... 3
2.2. HTTP Authentication ... 4

3. Beginner Example **7**
3.1. Access API Entry Point ... 7
3.2. List Data Center Collection ... 9
3.3. List Host Cluster Collection ... 10
3.4. List Logical Networks Collection ... 11
3.5. List Host Collection ... 12
3.6. Approve Host ... 13
3.7. Create NFS Data Storage ... 14
3.8. Create NFS ISO Storage ... 15
3.9. Attach Storage Domains to Data Center ... 16
3.10. Activate Storage Domains ... 17
3.11. Create Virtual Machine ... 17
3.12. Create Virtual Machine NIC ... 19
3.13. Create Virtual Machine Storage Disk ... 19
3.14. Attach ISO Image to Virtual Machine ... 20
3.15. Start Virtual Machine ... 21
3.16. Check System Events ... 22
3.17. Example Completion ... 23

4. Entry Point **25**
4.1. Product Information ... 26
4.2. Link elements ... 26
4.3. Special object elements ... 28
4.4. Summary element ... 28

5. Compatibility Level Versions **29**

6. Capabilities **31**
6.1. Version-Dependent Capabilities ... 31
6.1.1. Current Version ... 32
6.1.2. Features ... 32
6.1.3. CPUs ... 32
6.1.4. Power Managers ... 33
6.1.5. Fence Types ... 34
6.1.6. Storage Types ... 34
6.1.7. Storage Domain Types ... 34

6.1.8. Virtual Machine Types ... 35
6.1.9. Boot Devices ... 35
6.1.10. Display Types ... 35
6.1.11. NIC Interface Types ... 35
6.1.12. Disk Types ... 36
6.1.13. OS Types ... 36
6.1.14. Disk Formats ... 37
6.1.15. Disk Interfaces ... 37
6.1.16. Virtual Machine Affinities ... 37
6.1.17. Custom Properties ... 38
6.1.18. Boot Protocols ... 38
6.1.19. Error Handling ... 38
6.1.20. Storage Formats ... 39
6.1.21. Resource Status States ... 39
6.2. Permits ... 39
6.3. Scheduling Policies ... 39
6.4. Capabilities Example ... 40

7. Common Features 45
7.1. Representations ... 45
7.2. Collections ... 46
7.2.1. Listing All Resources in a Collection ... 46
7.2.2. Listing Extended Resource Sub-Collections ... 46
7.2.3. Searching Collections with Queries ... 47
7.2.4. Creating a Resource in a Collection ... 49
7.3. Resources ... 51
7.3.1. Retrieving a Resource ... 51
7.3.2. Updating a Resource ... 51
7.3.3. Deleting a Resource ... 52
7.3.4. Sub-Collection Relationships ... 52
7.3.5. XML Element Relationships ... 53
7.3.6. Actions ... 53
7.3.7. Permissions ... 55
7.3.8. Handling Errors ... 56

8. Data Centers 57
8.1. Storage Domains Sub-Collection ... 59
8.1.1. Attaching a Storage Domain ... 59
8.1.2. Actions ... 60
8.2. Force Remove Action ... 60

9. Host Clusters 63
9.1. Networks Sub-Collection ... 66

10. Networks 69

11. Storage Domains 71
11.1. Storage types ... 73
11.1.1. NFS ... 73
11.1.2. iSCSI and FCP ... 74
11.1.3. LocalFS ... 74
11.2. Export Storage Domains ... 75
11.3. Files Sub-Collection ... 76
11.4. Import Existing Storage Domain ... 77
11.5. Delete Storage Domain ... 77

12. Hosts 79

12.1. Power Management 82
12.2. Memory Management 83
12.3. Network Interface Sub-Collection 84
12.3.1. Bonded Interfaces 86
12.3.2. Network Interface Statistics 87
12.3.3. Attach Action 88
12.3.4. Detach Action 89
12.4. Storage Sub-Collection 89
12.5. Statistics Sub-Collection 90
12.6. Actions 91
12.6.1. Install Action 91
12.6.2. Activate Action 92
12.6.3. Fence Action 92
12.6.4. Deactivate Action 92
12.6.5. Approve Action 93
12.6.6. iSCSI Login Action 93
12.6.7. iSCSI Discover Action 93
12.6.8. Commit Network Configuration Action 94

13. Virtual Machines 95
13.1. Disks Sub-Collection 100
13.1.1. Disk Cloning 102
13.1.2. Disk Statistics 103
13.2. Network Interfaces Sub-Collection 104
13.2.1. Network Interface Statistics 105
13.3. CD-ROMs Sub-Collection 106
13.4. Snapshots Sub-Collection 107
13.5. Statistics Sub-Collection 108
13.6. Actions 110
13.6.1. Start Action 110
13.6.2. Stop Action 111
13.6.3. Shutdown Action 111
13.6.4. Suspend Action 111
13.6.5. Detach Action 112
13.6.6. Migrate Action 112
13.6.7. Export Action 112
13.6.8. Move Action 113
13.6.9. Ticket Action 113
13.6.10. Force Remove Action 114

14. Templates 115
14.1. Export Action 118

15. Virtual Machine Pools 121

16. Domains 123
16.1. Domain Users Sub-Collection 123
16.2. Domain Groups Sub-Collection 124

17. Groups 125

18. Roles 127
18.1. Permits Sub-Collection 128

19. Users 131

20. Tags 135
20.1. Associating Tags With a Host, User or VM 135

20.2. Parent Tags 137

21. Events **139**

21.1. Searching Events 140

21.2. Paginating Events 141

A. API Usage with cURL **143**

B. Java Keystores **147**

C. Event Codes **149**

D. Timezones **163**

E. Revision History **167**

Preface

The Red Hat Enterprise Virtualization platform is a richly featured virtualization that provides fully integrated management across virtual machines with multiple hosts. It is based on Kernel-based Virtual Machine (KVM), the leading open source virtualization platform, and provides superior technical capabilities. The platform offers scalability in the management of a large number of physical hosts and virtual machines.

1. About This Guide

This guide explains how to use Red Hat Enterprise Virtualization Manager's Representational State Transfer (REST) API. This document covers the fundamentals of the REST architectural concepts in the context of a virtualization environment and provides examples of the API in operation.

2. Audience

The target audience for this guide includes **developers** and **system administrators** who aim to integrate their Red Hat Enterprise Virtualization environment with third-party applications and scripts.

3. The Red Hat Enterprise Virtualization Documentation Suite

The Red Hat Enterprise Virtualization documentation suite provides information on installation, development of applications, configuration and usage of the Red Hat Enterprise Virtualization platform and its related products.

The Red Hat Enterprise Virtualization documentation suite

- *Red Hat Enterprise Virtualization Hypervisor Release Notes* contain release specific information for Red Hat Enterprise Virtualization Hypervisors.

- *Red Hat Enterprise Virtualization Manager Release Notes* contain release specific information for Red Hat Enterprise Virtualization Managers.

- *Red Hat Enterprise Virtualization Installation Guide* describes the installation prerequisites and procedures. Read this if you need to install Red Hat Enterprise Virtualization. The installation of hosts, manager and storage are covered in this guide. You will need to refer to the *Red Hat Enterprise Virtualization Administration Guide* to configure the system before you can start using the platform.

- *Red Hat Enterprise Virtualization Administration Guide* describes how to setup, configure and manage Red Hat Enterprise Virtualization. It assumes that you have successfully installed the Red Hat Enterprise Virtualization manager and hosts.

- *Red Hat Enterprise Virtualization User Portal Guide* describes how users of the Red Hat Enterprise Virtualization system can access and use virtual desktops.

- *Red Hat Enterprise Virtualization Storage and Networking Guide* provides detailed information on the storage and networking subsystems of Red Hat Enterprise Virtualization.

- *Red Hat Enterprise Linux Hypervisor Deployment Guide* describes how to deploy and install the hypervisor. Read this guide if you need advanced information about installing and deploying Hypervisors. The basic installation of Hypervisor hosts is also described in the *Red Hat Enterprise Virtualization Installation Guide.*

- *Red Hat Enterprise Linux V2V Guide* describes importing virtual machines from KVM, Xen and VMware ESX to Red Hat Enterprise Virtualization and KVM managed by libvirt.
- *Red Hat Enterprise Virtualization REST API Guide* (the book you are reading) describes how to use the REST API to set up and manage virtualization tasks. Use this guide if you wish to develop systems which integrate with Red Hat Enterprise Virtualization, using an open and platform independent API.
- *Red Hat Enterprise Virtualization Security Guide* describes security concepts, features and best practices surrounding Red Hat Enterprise Virtualization.
- *Red Hat Enterprise Virtualization Technical Reference Guide* describes the technical architecture of Red Hat Enterprise Virtualization and its interactions with existing infrastructure.

4. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

4.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

[1] https://fedorahosted.org/liberation-fonts/

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

4.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

4.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

5. Getting Help and Giving Feedback

5.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at *http://access.redhat.com*. Through the customer portal, you can:

- search or browse through a knowledgebase of technical support articles about Red Hat products.
- submit a support case to Red Hat Global Support Services (GSS).
- access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at *https://www.redhat.com/mailman/listinfo*. Click on the name of any mailing list to subscribe to that list or to access the list archives.

5.2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: *http://bugzilla.redhat.com/* against the product **Red Hat Enterprise Virtualization Manager.**

When submitting a bug report, be sure to mention the manual's identifier: *Guides-REST API*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

6. Backwards Compatibility Statement

Red Hat will make commercially reasonable efforts to ensure that applications developed for a certain version of Red Hat Enterprise Virtualization (RHEV) will work unmodified with future versions of RHEV within the same major release. Compatibility between different major releases is not guaranteed.

Introduction

Red Hat Enterprise Virtualization Manager provides a **Representational State Transfer (REST)** API. The API provides software developers and system administrators with control over their Red Hat Enterprise Virtualization environment outside of the standard web interface. The REST API is useful for developers and administrators who aim to integrate the functionality of a Red Hat Enterprise Virtualization environment with custom scripts or external applications that access the API via the standard Hypertext Transfer Protocol (HTTP).

The benefits of the REST API are:

- Broad client support - Any programming language, framework, or system with support for HTTP protocol can use the API;
- Self descriptive - Client applications require minimal knowledge of the virtualization infrastructure as many details are discovered at runtime;
- Resource-based model - The resource-based REST model provides a natural way to manage a virtualization platform.

This provides developers and administrators with the ability to:

- Integrate with enterprise IT systems.
- Integrate with third-party virtualization software.
- Perform automated maintenance or error checking tasks.
- Automate repetitive tasks in a Red Hat Enterprise Virtualization environment with scripts.

This documentation acts as a reference to the Red Hat Enterprise Virtualization Manager REST API. It aims to provide developers and administrators with instructions and examples to help harness the functionality of their Red Hat Enterprise Virtualization environment through the REST API.

1.1. Representational State Transfer

Representational State Transfer (REST) is a design architecture that focuses on resources for a specific service and their representations. A resource representation is a key abstraction of information that corresponds to one specific managed element on a server. A client sends a request to a server element located at a Uniform Resource Identifier (URI) and performs operations with standard HTTP methods, such as **`GET`**, **`POST`**, **`PUT`**, and **`DELETE`**. This provides a stateless communication between the client and server where each request acts independent of any other request and contains all necessary information to complete the request.

1.2. Prerequisites

This guide requires the following:

- A networked installation of Red Hat Enterprise Virtualization Manager 3.0, which includes the REST API;
- A client or programming library that initiates and receives HTTP requests from the REST API. As an example, this guide includes basic instructions on use with **cURL** in *Appendix A, API Usage with cURL*;

- Knowledge of Hypertext Transfer Protocol (HTTP), which is the protocol used for REST API interactions. The Internet Engineering Task Force provides a Request for Comments (RFC) explaining the Hypertext Transfer Protocol at *http://www.ietf.org/rfc/rfc2616.txt*; and,
- Knowledge of Extensible Markup Language (XML), which the API uses to construct resource representations. The W3C provides a full specification on XML at *http://www.w3.org/TR/xml/*.

Authentication and Security

This chapter provides information on authorization through Red Hat Enterprise Virtualization Manager's security.

2.1. TLS/SSL Certification

The API requires Hypertext Transfer Protocol Secure (HTTPS) [1] for secure transport-level encryption of requests. This involves a process of attaining a certificate from your Red Hat Enterprise Virtualization Manager server and importing it into your client's certificate store.

Procedure 2.1. Attain a certificate

This process helps a user attain a certificate from the Red Hat Enterprise Virtualization Manager and transfer it to the client machine. A user achieves this using one of three methods:

1. **Method 1 -** Use a command line tool to download the certificate from the server. Examples of command line tools include **cURL** and **Wget**; both are available for multiple platforms.

   a. If using **cURL**:

   ```
   curl -o rhevm.cer http://[rhevm-server]:8080/ca.crt
   ```

   b. If using **Wget**:

   ```
   wget -O rhevm.cer http://[rhevm-server]:8080/ca.crt
   ```

2. **Method 2 -** Use a web browser to navigate to the certificate located at:

   ```
   http://[rhevm-server]:8080/ca.crt
   ```

   Depending on the chosen browser, the certificate either downloads or imports into the browser's keystore.

   - **If the browser downloads the certificate:** save the file as **`rhevm.cer`**.

     If the browser imports the certificate: export it from the browser's certification options and save it as **`rhevm.cer`**.

3. **Method 3 -** Access your Red Hat Enterprise Virtualization Manager server either physically or through a secure shell (SSH) client, export the certificate from the server's keystore and copy it to your client machine.

   a. Access your Red Hat Enterprise Virtualization Manager server as the **`root`** user.

   b. Export a certificate from the server's keystore using the Java **keytool** management utility:

   ```
   keytool -exportcert -keystore /etc/pki/rhevm/.keystore -alias rhevm -storepass
    mypass -file rhevm.cer
   ```

   This creates a certificate file called **`rhevm.cer`**.

[1] HTTPS is described in *RFC 2818 HTTP Over TLS* [http://tools.ietf.org/html/rfc2818].

c. Copy the certificate to the client machine using the **scp** command:

```
scp rhevm.cer [username]@[client-machine]:[directory]
```

Each of the three methods results in a certificate file named **rhevm.cer** on your client machine. An API user imports this file into the client's certificate store.

Procedure 2.2. Import a certificate to your client

- A certificate import for your client relies on how the client itself stores and interprets certificates. This guide contains an example on importing to a Java keystore in *Appendix B, Java Keystores*. For other clients, please refer to your client documentation for more information on importing a certificate.

2.2. HTTP Authentication

An API user submits a mandatory Red Hat Enterprise Virtualization Manager username and password with all requests to the API and uses HTTP Basic Authentication [2] to encode these credentials. If a request does not include an appropriate **Authorization** header, the API sends a **401 Authorization Required** as a result:

Example 2.1. Access to the REST API without appropriate credentials

```

HEAD [base] HTTP/1.1
Host: [host]

HTTP/1.1 401 Authorization Required
```

Request are issued with an **Authorization** header for the specified realm. An API user encodes an appropriate Red Hat Enterprise Virtualization Manager domain and user in the supplied credentials with the **username@domain:password** convention.

Table 2.1. Encoding credentials for access to the API

Item	Value
username	rhevmadmin
domain	domain.example.com
password	123456
unencoded credentials	rhevmadmin@domain.example.com:123456
base64 encoded credentials	cmhldm1hZG1pbkBibGFjay5xdW1yYW5ldC5jb206MTIzNDU2

This table shows the process for encoding credentials in base64.

An API user provides the base64 encoded credentials as shown:

Example 2.2. Access to the REST API with appropriate credentials

[2] Basic Authentication is described in *RFC 2617 HTTP Authentication: Basic and Digest Access Authentication* [http://tools.ietf.org/html/rfc2617].

```
HEAD [base] HTTP/1.1
Host: [host]
Authorization: Basic cmhldm1hZG1pbkBibGFjay5xdW1yYW5ldC5jb206MTIzNDU2

HTTP/1.1 200 OK
...
```

Important

Basic authentication involves potentially sensitive information, such as passwords, sent as plain text. REST API requires Hypertext Transfer Protocol Secure (HTTPS) for transport-level encryption of plain-text requests. See *Section 2.1, “TLS/SSL Certification”* for more information.

Important

Some base64 libraries break the result into multiple lines and terminate each line with a newline character. This breaks the header and causes a faulty request. The Authorization header requires the encoded credentials on a single line within the header.

Beginner Example

This chapter provides an example to demonstrate the REST API's ability to create a virtual machine within a basic Red Hat Enterprise Virtualization environment.

In addition to the standard prerequisites (see *Section 1.2, “Prerequisites”*), this example requires the following:

- A networked and configured host containing Red Hat Enterprise Virtualization Hypervisor;
- An ISO file containing a desired virtual machine operating system to install. This chapter uses Red Hat Enterprise Linux Server 6 for our installation ISO example; and
- Red Hat Enterprise Virtualization Platform's uploader tool to upload your chosen operating system ISO file.

This example uses **cURL** to demonstrate REST requests with a client application. Find out more about **cURL** use with the REST API in *Appendix A, API Usage with cURL*. Note that any application capable of HTTP requests can substitute for **cURL**.

Important

For simplicity, the HTTP request headers in this example omit the **`Host:`** and **`Authorization:`** fields. However, these fields are mandatory and require data specific to your installation of Red Hat Enterprise Virtualization Manager.

Important

All **cURL** examples include placeholders for authentication details (**`USER:PASS`**) and certificate location (**`CERT`**). Ensure all requests performed with **cURL** fulfil certification and authentication requirements. See *Chapter 2, Authentication and Security* and *Appendix A, API Usage with cURL* for more information.

Note

Red Hat Enterprise Virtualization Manager generates a globally unique identifier (GUID) for the **`id`** attribute for each resource. Identifier codes in this example might appear different to the identifier codes in your Red Hat Enterprise Virtualization environment.

3.1. Access API Entry Point

The following request retrieves a representation of the main entry point of the API.

Example 3.1. Access the API entry point

```
GET /api HTTP/1.1
Accept: application/xml
```

cURL command:

curl -X GET -H "Accept: application/xml" -u *[USER:PASS]* --cacert *[CERT]* https://*[RHEVM Host]*:8443/api

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<api>
    <link rel="capabilities" href="/api/capabilities"/>
    <link rel="clusters" href="/api/clusters"/>
    <link rel="clusters/search" href="/api/clusters?search={query}"/>
    <link rel="datacenters" href="/api/datacenters"/>
    <link rel="datacenters/search" href="/api/datacenters?search={query}"/>
    <link rel="events" href="/api/events"/>
    <link rel="events/search" href="/api/events?search={query}"/>
    <link rel="hosts" href="/api/hosts"/>
    <link rel="hosts/search" href="/api/hosts?search={query}"/>
    <link rel="networks" href="/api/networks"/>
    <link rel="roles" href="/api/roles"/>
    <link rel="storagedomains" href="/api/storagedomains"/>
    <link rel="storagedomains/search" href="/api/storagedomains?search={query}"/>
    <link rel="tags" href="/api/tags"/>
    <link rel="templates" href="/api/templates"/>
    <link rel="templates/search" href="/api/templates?search={query}"/>
    <link rel="users" href="/api/users"/>
    <link rel="groups" href="/api/groups"/>
    <link rel="domains" href="/api/domains"/>
    <link rel="vmpools" href="/api/vmpools"/>
    <link rel="vmpools/search" href="/api/vmpools?search={query}"/>
    <link rel="vms" href="/api/vms"/>
    <link rel="vms/search" href="/api/vms?search={query}"/>
    <special_objects>
        <link rel="templates/blank"
          href="/api/templates/00000000-0000-0000-0000-000000000000"/>
        <link rel="tags/root"
          href="/api/tags/00000000-0000-0000-0000-000000000000"/>
    </special_objects>
    <product_info>
        <name>Red Hat Enterprise Virtualization</name>
        <vendor>Red Hat</vendor>
        <version revision="0" build="0" minor="0" major="3"/>
    </product_info>
    <summary>
        <vms>
            <total>5</total>
            <active>0</active>
        </vms>
        <hosts>
            <total>1</total>
            <active>1</active>
        </hosts>
        <users>
            <total>1</total>
            <active>1</active>
        </users>
        <storage_domains>
            <total>2</total>
            <active>2</active>
        </storage_domains>
    </summary>
</api>
```

The entry point provides a user with links to the collections in a virtualization environment. The **`rel=`** attribute of each collection link provides a reference point for each link. The next step in this example examines the **`datacenter`** collection, which is available through the **`rel="datacenter"`** link.

The entry point also contains other data such as **`product_info`**, **`special_objects`** and **`summary`**. This data is covered in chapters outside this example.

3.2. List Data Center Collection

Red Hat Enterprise Virtualization Manager creates a **`Default`** data center on installation. This example uses the **`Default`** data center as the basis for our virtual environment.

The following request retrieves a representation of the data center collection:

Example 3.2. List data center collection

```
GET /api/datacenters HTTP/1.1
Accept: application/xml
```

cURL command:

```
curl -X GET -H "Accept: application/xml" -u [USER:PASS] --cacert [CERT]
https://[RHEVM Host]:8443/api/datacenters
```

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<data_centers>
    <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"
      href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988">
        <name>Default</name>
        <description>The default Data Center</description>
        <link rel="storagedomains"
          href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/
          storagedomains"/>
        <link rel="permissions"
          href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/permissions"/>
        <storage_type>nfs</storage_type>
        <storage_format>v1</storage_format>
        <version minor="0" major="3"/>
        <supported_versions>
            <version minor="0" major="3"/>
        </supported_versions>
        <status>
            <state>up</state>
        </status>
    </data_center>
```

```
</data_centers>
```

Note the **`id`** code of your **`Default`** data center. This code identifies this data center in relation to other resources of your virtual environment.

The data center also contains a link to the **`storagedomains`** sub-collection. The data center uses this sub-collection to attach storage domains from the **`storagedomains`** main collection, which this example covers later.

3.3. List Host Cluster Collection

Red Hat Enterprise Virtualization Manager creates a **`Default`** host cluster on installation. This example uses the **`Default`** cluster to group resources in your Red Hat Enterprise Virtualization environment.

The following request retrieves a representation of the cluster collection:

Example 3.3. List host clusters collection

```
GET /api/clusters HTTP/1.1
Accept: application/xml
```

cURL command:

curl -X GET -H "Accept: application/xml" -u *[USER:PASS]* --cacert *[CERT]* https://*[RHEVM Host]*:8443/api/clusters

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<clusters>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95">
        <name>Default</name>
        <description>The default server cluster</description>
        <link rel="networks"
          href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks"/>
        <link rel="permissions"
          href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/permissions"/>
        <cpu id="Intel Penryn Family"/>
        <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"
          href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988"/>
        <memory_policy>
            <overcommit percent="100"/>
            <transparent_hugepages>
                <enabled>false</enabled>
            </transparent_hugepages>
        </memory_policy>
        <scheduling_policy/>
        <version minor="0" major="3"/>
        <error_handling>
            <on_error>migrate</on_error>
        </error_handling>
    </cluster>
</clusters>
```

Note the **id** code of your **Default** host cluster. This code identifies this host cluster in relation to other resources of your virtual environment.

The **Default** cluster is associated with the **Default** data center through a relationship using the **id** and **href** attributes of the **data_center** element.

The **networks** sub-collection contains a list of associated network resources for this cluster. The next section examines the **networks** collection in more detail.

3.4. List Logical Networks Collection

Red Hat Enterprise Virtualization Manager creates a default **rhevm** network on installation. This network acts as the management network for Red Hat Enterprise Virtualization Manager to access hypervisor hosts.

This network is associated with our **Default** cluster and is a member of the **Default** data center. This example uses the **rhevm** network to connect our virtual machines.

The following request retrieves a representation of the logical networks collection:

Example 3.4. List logical networks collection

```
GET /api/networks HTTP/1.1
Accept: application/xml
```

cURL command:

```
curl -X GET -H "Accept: application/xml" -u [USER:PASS] --cacert [CERT]
https://[RHEVM Host]:8443/api/networks
```

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<networks>
    <network id="00000000-0000-0000-0000-000000000009"
      href="/api/networks/00000000-0000-0000-0000-000000000009">
        <name>rhevm</name>
        <description>Management Network</description>
        <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"
          href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988"/>
        <stp>false</stp>
        <status>
            <state>operational</state>
        </status>
        <display>false</display>
    </network>
</networks>
```

The **rhevm** network is attached to the **Default** data center through a relationship using the data center's **id** code.

The **rhevm** network is also attached to the **Default** cluster through a relationship in the cluster's **network** sub-collection.

3.5. List Host Collection

This example uses a Red Hat Enterprise Virtualization Hypervisor host. Red Hat Enterprise Virtualization Manager automatically registers any configured Red Hat Enterprise Virtualization Hypervisor. This example retrieves a representation of the hosts collection and shows a Red Hat Enterprise Virtualization Hypervisor host named **`hypervisor`** registered with the virtualization environment.

Example 3.5. List hosts collection

```
GET /api/hosts HTTP/1.1
Accept: application/xml
```

cURL command:

```
curl -X GET -H "Accept: application/xml" -u [USER:PASS] --cacert [CERT]
https://[RHEVM Host]:8443/api/hosts
```

The API returns the following representation:

```
HTTP/1.1 200 OK
Accept: application/xml

<hosts>
    <host id="0656f432-923a-11e0-ad20-5254004ac988"
      href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988">
        <name>hypervisor</name>
        <actions>
            <link rel="install"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/install"/>
            <link rel="activate"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/activate"/>
            <link rel="fence"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/fence"/>
            <link rel="deactivate"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/deactivate"/>
            <link rel="approve"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/approve"/>
            <link rel="iscsilogin"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/iscsilogin"/>
            <link rel="iscsidiscover"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/iscsidiscover"/>
            <link rel="commitnetconfig"
              href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/
              commitnetconfig"/>
        </actions>
        <link rel="storage"
          href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/storage"/>
        <link rel="nics"
          href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/nics"/>
        <link rel="tags"
          href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/tags"/>
        <link rel="permissions"
          href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/permissions"/>
        <link rel="statistics"
          href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/statistics"/>
        <address>10.64.14.110</address>
        <status>
            <state>non_operational</state>
        </status>
        <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
```

```
          href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
        <port>54321</port>
        <storage_manager>true</storage_manager>
        <power_management>
            <enabled>false</enabled>
            <options/>
        </power_management>
        <ksm>
            <enabled>false</enabled>
        </ksm>
        <transparent_hugepages>
            <enabled>true</enabled>
        </transparent_hugepages>
        <iscsi>
            <initiator>iqn.1994-05.com.example:644949fe81ce</initiator>
        </iscsi>
        <cpu>
            <topology cores="2"/>
            <name>Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz</name>
            <speed>2993</speed>
        </cpu>
        <summary>
            <active>0</active>
            <migrating>0</migrating>
            <total>0</total>
        </summary>
    </host>
</hosts>
```

Note the **`id`** code of your **`Default`** host. This code identifies this host in relation to other resources of your virtual environment.

This host is a member of the **`Default`** cluster and accessing the **`nics`** sub-collection shows this host has a connection to the **`rhevm`** network.

3.6. Approve Host

The **`hypervisor`** host resource contains an **`approve`** action. A user accesses this action's URI with a **`POST`** request.

Example 3.6. Approve a pre-configured Red Hat Enterprise Virtualization Hypervisor host

```
POST /api/hosts/0656f432-923a-11e0-ad20-5254004ac988/approve HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/xml" -u [USER:PASS] --cacert [CERT] -d "<action/>" https://[RHEVM Host]:8443/api/hosts/0656f432-923a-11e0-ad20-5254004ac988/approve
```

The POST request requires a body for the message entities to initiate an action. Since the action does not require additional parameters, the body contains an empty **`action`** element.

Use the **`approve`** action only for Red Hat Enterprise Virtualization Hypervisor hosts. Red Hat Enterprise Linux hosts require a different process to connect to the virtualization environment.

This approves and activates the host for use in your virtual environment. The **`status`** for **`hypervisor`** changes from **`non_operational`** to **`up`**.

3.7. Create NFS Data Storage

An NFS data storage domain is an exported NFS share attached to a data center and provides storage for virtualized guest images. Creation of a new storage domain requires a **`POST`** request, with the storage domain representation included, sent to the URL of the storage domain collection.

Example 3.7. Create an NFS data storage domain

```
POST /api/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain>
  <name>data1</name>
  <type>data</type>
  <storage>
    <type>nfs</type>
    <address>192.168.0.10</address>
    <path>/data1</path>
  </storage>
  <host>
    <name>hypervisor</name>
  </host>
</storage_domain>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<storage_domain><name>data1</
name><type>data</type><storage><type>nfs</type><address>192.168.0.10</
address><path>/data1</path></storage><host><name>hypervisor</name></
host></storage_domain>" https://[RHEVM Host]:8443/api/storagedomains
```

The API creates a NFS data storage domain called **`data1`** with an export path of `192.168.0.10:/data1` and sets access to the storage domain through the **`hypervisor`** host. The API also returns the following representation of the newly created storage domain resource:

```
HTTP/1.1 200 OK
Accept: application/xml

<storage_domain id="9ca7cb40-9a2a-4513-acef-dc254af57aac"
  href="/api/storagedomains/9ca7cb40-9a2a-4513-acef-dc254af57aac">
    <name>data1</name>
    <link rel="permissions"
      href="/api/storagedomains/9ca7cb40-9a2a-4513-acef-dc254af57aac/
      permissions"/>
    <link rel="files"
      href="/api/storagedomains/9ca7cb40-9a2a-4513-acef-dc254af57aac/files"/>
    <type>data</type>
    <master>false</master>
    <storage>
        <type>nfs</type>
        <address>192.168.0.10</address>
        <path>/data1</path>
    </storage>
    <available>175019917312</available>
    <used>27917287424</used>
    <committed>10737418240</committed>
    <storage_format>v1</storage_format>
    <host id="0656f432-923a-11e0-ad20-5254004ac988"
      href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988">
</storage_domain>
```

3.8. Create NFS ISO Storage

An NFS ISO storage domain is a mounted NFS share attached to a data center and provides storage for DVD/CD-ROM ISO and virtual floppy disk (VFD) image files. Creation of a new storage domain requires a **POST** request, with the storage domain representation included, sent to the URL of the storage domain collection.

Example 3.8. Create an NFS ISO storage domain

```
POST /api/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain>
  <name>iso1</name>
  <type>iso</type>
  <storage>
    <type>nfs</type>
    <address>192.168.0.10</address>
    <path>/iso1</path>
  </storage>
  <host>
    <name>hypervisor</name>
  </host>
</storage_domain>
```

cURL command:

curl -X POST -H "Accept: application/xml" -H "Content-Type: application/xml" -u *[USER:PASS]* --cacert *[CERT]* -d "<storage_domain><name>iso1</name><type>iso</type><storage><type>nfs</type><address>192.168.0.10</address><path>/iso1</path></storage><host><name>hypervisor</name></host></storage_domain>" https://*[RHEVM Host]*:8443/api/storagedomains

The API creates a NFS iso storage domain called **iso1** with an export path of `192.168.0.10:/iso1` and gets access to the storage domain through the **hypervisor** host. The API also returns the following representation of the newly created storage domain resource:

```
HTTP/1.1 200 OK
Accept: application/xml

<storage_domain id="00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"
  href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da">
    <name>iso1</name>
    <link rel="permissions"
      href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/
      permissions"/>
    <link rel="files"
      href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/files"/>
    <type>iso</type>
    <host id="" href="">
    <master>false</master>
    <storage>
        <type>nfs</type>
        <address>192.168.0.10</address>
        <path>/iso1</path>
    </storage>
    <available>82678120448</available>
    <used>18253611008</used>
    <committed>0</committed>
    <storage_format>v1</storage_format>
    <host id="0656f432-923a-11e0-ad20-5254004ac988"
      href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988">
</storage_domain>
```

3.9. Attach Storage Domains to Data Center

The following example attaches the **`data1`** and **`iso1`** storage domains to the **`Default`** data center.

Example 3.9. Attach data1 storage domain to the Default data center

```
POST /api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain>
  <name>data1</name>
</storage_domain>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<storage_domain><name>data1</
name></storage_domain>" https://[RHEVM Host]:8443/api/
datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/storagedomains
```

Example 3.10. Attach iso1 storage domain to the Default data center

```
POST /api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain>
  <name>iso1</name>
</storage_domain>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<storage_domain><name>iso1</
name></storage_domain>" https://[RHEVM Host]:8443/api/
datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/storagedomains
```

These **`POST`** requests place our two new **`storage_domain`** resources in the **`storagedomains`** sub-collection of the **`Default`** data center. This means the **`storagedomain`** sub-collection contains attached storage domains of the data center.

3.10. Activate Storage Domains

This example activates the **`data1`** and **`iso1`** storage domains for the Red Hat Enterprise Virtualization Manager's use.

Example 3.11. Activate data1 storage domain

```
POST /api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains/
9ca7cb40-9a2a-4513-acef-dc254af57aac/activate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<action/>" https://[RHEVM
Host]:8443/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/
storagedomains/9ca7cb40-9a2a-4513-acef-dc254af57aac/activate
```

Example 3.12. Activate iso1 storage domain

```
POST /api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains/
00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/activate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<action/>" https://[RHEVM
Host]:8443/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/
storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/activate
```

This activates both storage domains for use with the data center.

3.11. Create Virtual Machine

The following example creates a virtual machine called **`vm1`** on the **`Default`** cluster using the virtualization environment's **`Blank`** template as a basis. The request also defines the virtual machine's **`memory`** as 512 MB and sets the **`boot`** device to a virtual hard disk.

Example 3.13. Create a virtual machine

```
POST /api/vms HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vm>
  <name>vm1</name>
  <cluster>
    <name>default</name>
  </cluster>
  <template>
    <name>Blank</name>
  </template>
  <memory>536870912</memory>
  <os>
    <boot dev="hd"/>
  </os>
</vm>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type:
application/xml" -u [USER:PASS] --cacert [CERT] -d "<vm><name>vm1</
name><cluster><name>default</name></cluster><template><name>Blank</
name></template><memory>536870912</memory><os><boot dev='hd'/></os></vm>"
https://[RHEVM Host]:8443/api/vms
```

The API returns the following representation of the newly created virtual machine resource:

```

HTTP/1.1 200 OK
Accept: application/xml

<vm id="6efc0cfa-8495-4a96-93e5-ee490328cf48"
  href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48">
    <name>vm1</name>
    <actions>
        <link rel="shutdown"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/shutdown"/>
        <link rel="start"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/start"/>
        <link rel="stop"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/stop"/>
        <link rel="suspend"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/suspend"/>
        <link rel="detach"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/detach"/>
        <link rel="export"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/export"/>
        <link rel="move"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/move"/>
        <link rel="ticket"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/ticket"/>
        <link rel="migrate"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/migrate"/>
    </actions>
    <link rel="disks"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/disks"/>
    <link rel="nics"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/nics"/>
    <link rel="cdroms"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/cdroms"/>
    <link rel="snapshots"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/snapshots"/>
    <link rel="tags"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/tags"/>
    <link rel="permissions"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/permissions"/>
    <link rel="statistics"
      href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/statistics"/>
    <type>desktop</type>
    <status>
        <state>down</state>
    </status>
    <memory>536870912</memory>
    <cpu>
        <topology cores="1" sockets="1"/>
    </cpu>
    <os type="Unassigned">
        <boot dev="cdrom"/>
    </os>
    <high_availability>
        <enabled>false</enabled>
        <priority>0</priority>
    </high_availability>
    <display>
        <type>spice</type>
        <monitors>1</monitors>
    </display>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <template id="00000000-0000-0000-0000-000000000000"
      href="/api/templates/00000000-0000-0000-0000-000000000000"/>
    <start_time>2011-06-15T04:48:02.167Z</start_time>
    <creation_time>2011-06-15T14:48:02.078+10:00</creation_time>
    <origin>rhev</origin>
    <stateless>false</stateless>
    <placement_policy>
        <affinity>migratable</affinity>
    </placement_policy>
    <memory_policy>
        <guaranteed>536870912</guaranteed>
    </memory_policy>
</vm>
```

3.12. Create Virtual Machine NIC

The following example creates a virtual network interface to connect the example virtual machine to the **rhevm** network.

Example 3.14. Create a virtual machine NIC

```
POST /api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/nics HTTP/1.1
Accept: application/xml
Content-type: application/xml

<nic>
  <interface>virtio</interface>
  <name>nic1</name>
  <network>
    <name>rhevm</name>
  </network>
</nic>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/xml" -u [USER:PASS] --cacert [CERT] -d "<nic><name>nic1</name><network><name>rhevm</name></network></nic>" https://[RHEVM Host]:8443/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/nics
```

3.13. Create Virtual Machine Storage Disk

The following example creates an 8 GB Copy-On-Write storage disk for the example virtual machine.

Example 3.15. Create a virtual machine storage disk

```
POST /api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/disks HTTP/1.1
Accept: application/xml
Content-type: application/xml

<disk>
    <storage_domains>
        <storage_domain id="9ca7cb40-9a2a-4513-acef-dc254af57aac"/>
    </storage_domains>
    <size>8589934592</size>
    <type>system</type>
    <interface>virtio</interface>
    <format>cow</format>
    <bootable>true</bootable>
</disk>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/xml" -u [USER:PASS] --cacert [CERT] -d "<disk><storage_domains><storage_domain id='9ca7cb40-9a2a-4513-acef-dc254af57aac'/></storage_domains><size>8589934592</size><type>system</type><interface>virtio</interface><format>cow</format><bootable>true</bootable></disk>" https://[RHEVM Host]:8443/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/disks
```

The **`storage_domain`** element tells the API to store the disk on the **`data1`** storage domain.

3.14. Attach ISO Image to Virtual Machine

The boot media for our example virtual machine requires an CD-ROM or DVD ISO image for an operating system installation. This example uses a Red Hat Enterprise Server 6 ISO image for installation.

ISO images must be available in the **`iso1`** ISO domain for the virtual machines to use. Red Hat Enterprise Virtualization Platform provides an uploader tool that ensures that the ISO images are uploaded into the correct directory path with the correct user permissions.

Once the ISO is uploaded, an API user requests the ISO storage domain's **`files`** sub-collection to view the file resource:

Example 3.16. View the files sub-collection in an ISO storage domain

```
GET /api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/files HTTP/1.1
Accept: application/xml
```

cURL command:

```
curl -X GET -H "Accept: application/xml" -u [USER:PASS] --cacert [CERT] https://[RHEVM Host]:8443/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/files
```

The API returns the following representation of the files sub-collection:

```
<files>
    <file id="rhel-server-6.0-x86_64-dvd.iso"
      href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/
      files/rhel-server-6.0-x86_64-dvd.iso.iso">
        <name>rhel-server-6.0-x86_64-dvd.iso.iso</name>
        <storage_domain id="00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"
          href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"/>
    </file>
</files>
```

An API user attaches the **`rhel-server-6.0-x86_64-dvd.iso`** to our example virtual machine.

Example 3.17. Attach an ISO image to the virtual machine

```
POST /api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/cdroms HTTP/1.1
Accept: application/xml
Content-type: application/xml

<cdrom>
  <file id="rhel-server-6.0-x86_64-dvd.iso"/>
</cdrom>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<cdrom><file id='rhel-
server-6.0-x86_64-dvd.iso'/></cdrom>" https://[RHEVM Host]:8443/api/
vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/cdroms
```

3.15. Start Virtual Machine

The virtual environment is complete and the virtual machine contains all necessary components to function. This example starts the virtual machine using the **`start`** action.

Example 3.18. Start the virtual machine

```
POST /api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/start HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
  <vm>
    <os>
      <boot dev="cdrom"/>
    </os>
  </vm>
</action>
```

cURL command:

```
curl -X POST -H "Accept: application/xml" -H "Content-Type: application/
xml" -u [USER:PASS] --cacert [CERT] -d "<action><vm><os><boot
dev='cdrom'/></os></vm></action>" https://[RHEVM Host]:8443/api/
vms/6efc0cfa-8495-4a96-93e5-ee490328cf48/start
```

The additional message entity sets the virtual machine's boot device to CD-ROM for this boot only. This enables the virtual machine to install Red Hat Enterprise Server 6 from the attached ISO image. The boot device reverts back to **`disk`** for all future boots.

3.16. Check System Events

The **`start`** action for the **`vm1`** creates several entries in the **`events`** collection. This example lists the events collection and identifies events specific to the API starting a virtual machine.

Example 3.19. List the events collection

```
GET /api/events HTTP/1.1
Accept: application/xml
```

cURL command:

```
curl -X GET -H "Accept: application/xml" -u [USER:PASS] --cacert [CERT]
https://[RHEVM Host]:8443/api/events
```

The API returns a representation that includes the following:

```
<events>
    ...
    <event id="103" href="/api/events/103">
        <description>User admin logged out.</description>
        <code>31</code>
        <severity>normal</severity>
        <time>2011-06-29T17:42:41.544+10:00</time>
        <user id="80b71bae-98a1-11e0-8f20-525400866c73"
          href="/api/users/80b71bae-98a1-11e0-8f20-525400866c73"/>
    </event>
    <event id="102" href="/api/events/102">
        <description>vm1 was started by admin (Host: hypervisor).</description>
        <code>153</code>
        <severity>normal</severity>
        <time>2011-06-29T17:42:41.499+10:00</time>
        <user id="80b71bae-98a1-11e0-8f20-525400866c73"
          href="/api/users/80b71bae-98a1-11e0-8f20-525400866c73"/>
        <vm id="6efc0cfa-8495-4a96-93e5-ee490328cf48"
          href="/api/vms/6efc0cfa-8495-4a96-93e5-ee490328cf48"/>
        <host id="0656f432-923a-11e0-ad20-5254004ac988"
          href="/api/hosts/0656f432-923a-11e0-ad20-5254004ac988"/>
    </event>
    <event id="101" href="/api/events/101">
        <description>User admin logged in.</description>
        <code>30</code>
        <severity>normal</severity>
        <time>2011-06-29T17:42:40.505+10:00</time>
        <user id="80b71bae-98a1-11e0-8f20-525400866c73"
          href="/api/users/80b71bae-98a1-11e0-8f20-525400866c73"/>
    </event>
    ...
</events>
```

The following events occur:

- **`id="101"`** - The API authenticates with the **`admin`** user's username and password.
- **`id="102"`** - The API, acting as the **`admin`** user, starts **`vm1`** on the **`hypervisor`** host.

- **`id="103"`** - The API logs out of the **`admin`** user account.

3.17. Example Completion

This example demonstrates how the API creates a virtual machine within a basic virtualization environment. The remainder of this guide provides specific details on the complete REST API featureset, including virtualization tasks of a higher complexity.

Chapter 4.

Entry Point

A user begins interacting with the API through a **GET** request on the entry point URI consisting of a **host** and **base**.

Example 4.1. Accessing the API Entry Point

If the **host** is `www.example.com` and the **base** is `/api`, the entry point appears with the following request:

```
GET /api HTTP/1.1
Accept: application/xml
Host: www.example.com
Authorization: [base64 encoded credentials]

HTTP/1.1 200 OK
Content-Type: application/xml

<api>
    <link rel="hosts" href="/api/hosts"/>
    <link rel="vms" href="/api/vms"/>
    ...
    <product_info>
        <name>Red Hat Enterprise Virtualization</name>
        <vendor>Red Hat</vendor>
        <version revision="0" build="0" minor="0" major="3"/>
    </product_info>
    <special_objects>
        <link rel="templates/blank" href="..."/>
        <link rel="tags/root" href="..."/>
    </special_objects>
    <summary>
        <vms>
            <total>10</total>
            <active>3</active>
        </vms>
        <hosts>
            <total>2</total>
            <active>2</active>
        </hosts>
        <users>
            <total>8</total>
            <active>2</active>
        </users>
        <storage_domains>
            <total>2</total>
            <active>2</active>
        </storage_domains>
    </summary>
</api>
```

Note

For simplicity, all other examples omit the **Host:** and **Authorization:** request headers and assume the `base` is the default **/api** path. This base path differs depending on your implementation.

4.1. Product Information

The entry point contains a **`product_info`** element to help an API user determine the legitimacy of the Red Hat Enterprise Virtualization environment. This includes the **`name`** of the product, the **`vendor`** and the **`version`**.

Example 4.2. Verify a genuine Red Hat Enterprise Virtualization environment

The follow elements identify a genuine Red Hat Enterprise Virtualization 3.0 environment:

```
<api>
    ...
    <product_info>
        <name>Red Hat Enterprise Virtualization</name>
        <vendor>Red Hat</vendor>
        <version revision="0" build="0" minor="0" major="3"/>
    </product_info>
    ...
</api>
```

4.2. Link elements

Access to the Entry Point provides **`link`** elements and URIs for all of the resource collections the API exposes. Each collection uses a relation type to identify the URI a client needs.

Table 4.1. Available Relationship Types

Relationship	Description
`capabilities`	Supported capabilities of the Red Hat Enterprise Virtualization Manager.
`datacenters`	Data centers.
`clusters`	Host clusters.
`networks`	Virtual networks.
`storagedomains`	Storage domains.
`hosts`	Hosts.
`vms`	Virtual machines.
`templates`	Templates.
`vmpools`	Virtual machine pools.
`domains`	Identity service domains.
`groups`	Imported identity service groups.
`roles`	Roles.
`users`	Users.
`tags`	Tags.
`events`	Events.

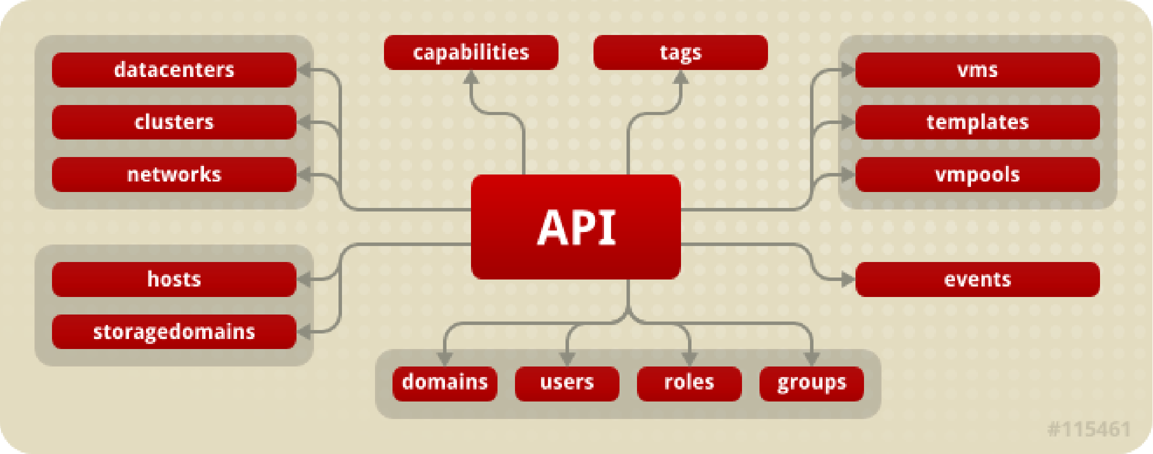

Figure 4.1. The relationship between the API entry point and the resource collections exposed by the API

Note

All URIs shown in example responses are illustrative. The format of all URIs returned by the server is opaque. Clients navigate to specific resources through the entry point URI and use the relationship types to access the URIs.

The server chooses to include absolute URIs or absolute paths [1] in the **`link`** element's **`href`** attribute, so clients are required to handle either form.

The **`link`** elements also contain a set of **`search`** URIs for certain collections. These URIs use URI templates [2] to integrate search queries. The purpose of the URI template is to accept a search expression using the natural HTTP pattern of a query parameter. The client does not require prior knowledge of the URI structure. Thus clients should treat these templates as being opaque and access them with a URI template library.

Each search query URI template is identified with a relation type using the convention **`"collection/search"`**.

Table 4.2. Relationships associated with search query URIs

Relationship	Description
`datacenters/search`	Query data centers.
`clusters/search`	Query host clusters.
`storagedomains/search`	Query storage domains.
`hosts/search`	Query hosts.
`vms/search`	Query virtual machines.
`templates/search`	Query templates.
`vmpools/search`	Query virtual machine pools.

[1] The RFC describing Uniform Resource Locator Generic Syntax provides a *Collected ABNF for URI* [http://tools.ietf.org/html/rfc3986#appendix-A] that explains the difference between these forms.
[2] The Internet-Draft describing the format of a URI Template is available at *http://tools.ietf.org/html/draft-gregorio-uritemplate-03*.

Relationship	Description
`events/search`	Query events.
`users/search`	Query users.

4.3. Special object elements

Special object elements define relationships to special fixed resources within the virtualization environment.

Table 4.3. Special Objects

Relationship	Description
`templates/blank`	The default **`blank`** virtual machine template for your virtualization environment. This template exists in every cluster as opposed to a standard template, which only exists in a single cluster.
`tags/root`	The **`root`** tag that acts a base for tag hierarchy in your virtualization environment.

4.4. Summary element

The summary element shows a high level summary of the system's statistics.

Table 4.4. Summary Elements

Element	Description
`vms`	Total number of vms and total number of active vms.
`hosts`	Total number of hosts and total number of active hosts.
`users`	Total number of users and total number of active users.
`storage_domains`	Total number of storage domains and total number of active storage domains.

Compatibility Level Versions

Each host connected to Red Hat Enterprise Virtualization Manager contains a version of VDSM. VDSM is the agent within the virtualization infrastructure that runs on a hypervisor or host and provides local management for virtual machines, networks and storage. Red Hat Enterprise Virtualization Manager controls hypervisors and hosts using current or older versions of VDSM.

The Manager migrates virtual machines from host to host within a cluster. This means the Manager excludes certain features from a current version of VDSM until all hosts within a cluster have the same VDSM version, or more recent, installed.

The API represents this concept as a **`compatibility level`** for each host, corresponding to the version of VDSM installed. A **`version`** element contains **`major`** and **`minor`** attributes, which describe the compatibility level.

When an administrator upgrades all hosts within a cluster to a certain level, the **`version`** level appears under a **`supported_versions`** element. This indicates the cluster's **`version`** is now updatable to that level. Once the administrator updates all clusters within a data center to a given level, the data center is updatable to that level.

Example 5.1. Upgrading compatibility levels

The API reports the following compatibility levels for Red Hat Enterprise Virtualization Manager 2.2 instance:

```
<host ...>
    ...
    <version major="2" minor="2"/>
    ...
</host>

<cluster ...>
    ...
    <version major="2" minor="2"/>
    <supported_versions/>
    ...
</cluster>

<data_center ...>
    ...
    <version major="2" minor="2"/>
    <supported_versions/>
    ...
</data_center>
```

All hosts within a cluster are updated to VDSM **`3.0`** and the API reports:

```
<host ...>
    ...
    <version major="3" minor="0"/>
    ...
</host>

<cluster ...>
    ...
    <version major="2" minor="2"/>
    <supported_versions>
        <version major="3" minor="0"/>
    </supported_versions>
    ...
</cluster>

<data_center ...>
    ...
    <version major="2" minor="2"/>
    <supported_versions/>
    ...
</data_center>
```

The cluster is now updatable to **`3.0`**. When the cluster is updated, the API reports:

```

<cluster ...>
    ...
    <version major="3" minor="0"/>
    <supported_versions/>
    ...
</cluster>

<data_center ...>
    ...
    <version major="2" minor="2"/>
    <supported_versions>
        <version major="3" minor="0"/>
    </supported_versions>
    ...
</data_center>
```

The API user updates the data center to **`3.0`**. Once upgraded, the API exposes features available in Red Hat Enterprise Virtualization 3.0 for this data center.

Capabilities

The **capabilities** collection provides information about the supported capabilities of a Red Hat Enterprise Virtualization Manager version. These capabilities include active features and available enumerated values for specific properties. An API user accesses this information through the **rel="capabilities"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

6.1. Version-Dependent Capabilities

The **capabilities** element contains any number of **version** elements that describe capabilities dependent on a compatibility level.

The **version** element includes attributes for **major** and **minor** version numbers. This indicates the current version level, which this document discusses in *Chapter 5, Compatibility Level Versions*.

The following representation shows capabilities specific to Red Hat Enterprise Virtualization Manager **3.0** and **2.2** respectively:

```
<capabilities>
    <version major="3" minor="0">
        ...
    </version>
    <version major="2" minor="2">
        ...
    </version>
    ...
</capabilities>
```

Each **version** contains a series of capabilities dependent on the version specified.

Table 6.1. Version-Dependent Capabilities

Capability Element	Description
current	Signifies if this **version** most recent supported compatibility level.
features	The list of features the version supports.
cpus	List of supported CPU types.
power_managers	List of supported fence agents and their configuration options.
fence_types	Supported fence actions for hosts.
storage_types	Supported storage types for a version.
storage_domain_types	List of all domain types.
vm_types	List of all virtual machine types.
boot_devices	Boot devices for a virtual machine.
display_types	Available display protocols for a virtual machine.
nic_interfaces	Network interface types for a virtual machine.
disk_types	List of storage device types.
os_types	List of operating system types.
disk_formats	List of storage device formats.
disk_interfaces	List of interfaces for storage devices.
vm_affinities	Affinities for a host's placement policy.

Capability Element	Description
custom_properties	Environment variables for a virtual machine's custom scripts.
boot_protocols	List of available protocols for IP assignment.
error_handling	Options to determine virtual machine handling when a host in a cluster becomes non-operational.
storage_formats	The format versions for storage meta-data.

6.1.1. Current Version

The **current** element signifies if the **version** specified is the most recent supported compatibility level. The value is either a Boolean **true** or **false**.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <current>true</current>
        ...
    </version>
</capabilities>
```

6.1.2. Features

Each **version** contains a list of compatible **features**.

Table 6.2. Feature Types

Feature Element	Type	Description
transparent_hugepages	Boolean: true or false	Defines the availability of Transparent Hugepages for hosts.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <features>
            <transparent_hugepages>true</transparent_hugepages>
        </features>
        ...
    </version>
</capabilities>
```

6.1.3. CPUs

Each cluster is configured with a minimal CPU type that all hosts in that cluster must support. Guests running on hosts within the cluster all run on this CPU type, ensuring that every guest is migratable to any host within the cluster.

Red Hat Enterprise Virtualization has a set of supported CPU types to choose from when configuring a cluster.

Table 6.3. CPU properties

Element	Description
id	An opaque identifier for the CPU model.

Element	Description
`level`	An indication as to the relative priority/preference for the CPUs in the list.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <cpus>
            <cpu id="Intel Conroe Family">
                <level>3</level>
            </cpu>
            <cpu id="Intel Penryn Family">
                <level>4</level>
            </cpu>
            ...
        </cpus>
        ...
    </version>
</capabilities>
```

6.1.4. Power Managers

Red Hat Enterprise Virtualization platform provides a list of supported **`power_managers`** for host fencing configuration. Each **`power_management`** option contains a set of properties.

Table 6.4. Power Management Properties

Element	Description
`type`	The supported fencing device type.
`options`	A list of options available to the supported fencing device. Options include a **`name`** and a value **`type`**.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <power_managers>
            <power_management type="alom">
                <options>
                    <option type="bool" name="secure"/>
                    <option type="int" name="port"/>
                </options>
            </power_management>
            <power_management type="apc">
                <options>
                    <option type="bool" name="secure"/>
                    <option type="int" name="port"/>
                    <option type="int" name="slot"/>
                </options>
            </power_management>
            <power_management type="bladecenter">
                <options>
                    <option type="bool" name="secure"/>
                    <option type="int" name="port"/>
                    <option type="int" name="slot"/>
                </options>
            </power_management>
            ...
        </power_managers>
        ...
    </version>
</capabilities>
```

6.1.5. Fence Types

The **`fence_types`** element defines **`fence_type`** options to fence a host and reduce power usage.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <fence_types>
            <fence_type>manual</fence_type>
            <fence_type>restart</fence_type>
            <fence_type>start</fence_type>
            <fence_type>stop</fence_type>
            <fence_type>status</fence_type>
        </fence_types>
        ...
    </version>
</capabilities>
```

6.1.6. Storage Types

The **`storage_types`** element defines the available physical **`storage_type`** options.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <storage_types>
            <storage_type>iscsi</storage_type>
            <storage_type>fcp</storage_type>
            <storage_type>nfs</storage_type>
            <storage_type>localfs</storage_type>
        </storage_types>
        ...
    </version>
</capabilities>
```

6.1.7. Storage Domain Types

The **`storage_domain_types`** element shows the available **`storage_domain_type`** options a user creates in a virtualization environment.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <storage_domain_types>
            <storage_domain_type>data</storage_domain_type>
            <storage_domain_type>iso</storage_domain_type>
            <storage_domain_type>export</storage_domain_type>
        </storage_domain_types>
        ...
    </version>
</capabilities>
```

6.1.8. Virtual Machine Types

The **`vm_types`** element defines each virtual machine type (**`vm_type`**) available for creation in a virtual environment.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <vm_types>
            <vm_type>desktop</vm_type>
            <vm_type>server</vm_type>
        </vm_types>
        ...
    </version>
</capabilities>
```

6.1.9. Boot Devices

Each virtual machine boots from a selected device. The **`boot_devices`** element lists such **`boot_device`** options.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <boot_devices>
            <boot_device>cdrom</boot_device>
            <boot_device>hd</boot_device>
            <boot_device>network</boot_device>
        </boot_devices>
        ...
    </version>
</capabilities>
```

6.1.10. Display Types

Access to a virtual machine through Red Hat Enterprise Virtualization Manager requires a display protocol. The **`display_types`** element lists each **`display_type`** protocol.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <display_types>
            <display_type>vnc</display_type>
            <display_type>spice</display_type>
        </display_types>
        ...
    </version>
</capabilities>
```

6.1.11. NIC Interface Types

A virtual machine requires a network interface to connect to a network. The **`nic_interfaces`** element defines the supported NIC types available. Each **`nic_interface`** depends on the drivers available for different types of virtual machines. VirtIO drivers are available for Red Hat Enterprise Linux 4.8 and above, and for Windows virtual machines. Windows supports rtl8139 without the need for drivers. Other Linux machines, or earlier versions of Red Hat Enterprise Linux, use e1000 or rtl8139.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <nic_interfaces>
            <nic_interface>e1000</nic_interface>
            <nic_interface>virtio</nic_interface>
            <nic_interface>rtl8139</nic_interface>
            <nic_interface>rtl8139_virtio</nic_interface>
        </nic_interfaces>
        ...
    </version>
</capabilities>
```

6.1.12. Disk Types

Each virtual machine requires a virtual disk for storage. The **`disk_types`** element lists the available virtual **`disk_type`** options.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <disk_types>
            <disk_type>data</disk_type>
            <disk_type>system</disk_type>
        <disk_types
        ...
    </version>
</capabilities>
```

6.1.13. OS Types

Each virtual machine contains an **`os_type`** value to define the virtual machine operating system. The default is **`unassigned`**.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <os_types>
            <os_type>unassigned</os_type>
            <os_type>windows_xp</os_type>
            <os_type>windows_2003</os_type>
            <os_type>windows_2008</os_type>
            <os_type>other_linux</os_type>
            <os_type>other</os_type>
            <os_type>rhel_5</os_type>
            <os_type>rhel_4</os_type>
            <os_type>rhel_3</os_type>
            <os_type>windows_2003x64</os_type>
            <os_type>windows_7</os_type>
            <os_type>windows_7x64</os_type>
            <os_type>rhel_5x64</os_type>
            <os_type>rhel_4x64</os_type>
            <os_type>rhel_3x64</os_type>
            <os_type>windows_2008x64</os_type>
            <os_type>windows_2008r2x64</os_type>
            <os_type>rhel_6</os_type>
            <os_type>rhel_6x64</os_type>
        <os_types>
        ...
    </version>
</capabilities>
```

6.1.14. Disk Formats

An API user selects the format for virtual disks. The **`disk_formats`** element defines the format types. The **`disk_format`** types include pre-allocated (**`raw`**) or thin-provisioned (Copy-On-Write or **`cow`**).

```

<capabilities>
    <version major="3" minor="0">
        ...
        <disk_formats>
            <disk_format>cow</disk_format>
            <disk_format>raw</disk_format>
        </disk_formats>
        ...
    </version>
</capabilities>
```

6.1.15. Disk Interfaces

The **`disk_interfaces`** element lists **`disk_interface`** options for emulated protocols to interface with virtual disks.

```

<capabilities>
    <version major="3" minor="0">
        ...
        <disk_interfaces>
            <disk_interface>ide</disk_interface>
            <disk_interface>virtio</disk_interface>
        </disk_interfaces>
        ...
    </version>
</capabilities>
```

6.1.16. Virtual Machine Affinities

Virtual machines migrate between hosts in a cluster. The **`vm_affinities`** element defines each available migration **`affinity`** for virtual machines.

```

<capabilities>
    <version major="3" minor="0">
        ...
        <vm_affinities>
            <affinity>migratable</affinity>
            <affinity>user_migratable</affinity>
            <affinity>pinned</affinity>
        </vm_affinities>
        ...
    </version>
</capabilities>
```

6.1.17. Custom Properties

The **`custom_properties`** element lists a set of environment variables for a virtual environment. The virtual environment uses these variables as parameters for event-triggered VDSM scripts. Each **`custom_property`** includes attributes for a property **`name`** and a regular expression (**`regexp`**) to define the format of the property value.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <custom_properties>
            <custom_property name="sap_agent" regexp="^(true|false)$"/>
            <custom_property name="sndbuf" regexp="^[0-9]+$"/>
            <custom_property name="vhost"
              regexp="^(([a-zA-Z0-9_]*):(true|false))
              (,(([a-zA-Z0-9_]*):(true|false)))*$"/>
            <custom_property name="viodiskcache"
              regexp="^(none|writeback|writethrough)$"/>
        </custom_properties>
        ...
    </version>
</capabilities>
```

6.1.18. Boot Protocols

The **`boot_protocol`** element lists each possible IP assignment **`boot_protocol`** for hosts when booting.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <boot_protocols>
            <boot_protocol>dhcp</boot_protocol>
            <boot_protocol>static</boot_protocol>
        </boot_protocols>
        ...
    </version>
</capabilities>
```

6.1.19. Error Handling

A Red Hat Enterprise Virtualization Manager determines whether to migrate virtual machines on a non-operational host using one of the **`on_error`** options the in the **`error_handling`** element.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <error_handling>
            <on_error>migrate</on_error>
            <on_error>do_not_migrate</on_error>
            <on_error>migrate_highly_available</on_error>
        </error_handling>
        ...
    </version>
</capabilities>
```

6.1.20. Storage Formats

The **storage_formats** element lists the available **format** versions for storage meta-data.

```
<capabilities>
    <version major="3" minor="0">
        ...
        <storage_formats>
            <format>v1</format>
            <format>v2</format>
        </storage_formats>
        ...
    </version>
</capabilities>
```

6.1.21. Resource Status States

Each **version** contains a set of states for resource statuses. These resource status elements include **creation_states**, **power_management_states**, **host_states**, **host_non_operational_details**, **network_states**, **storage_domain_states**, **template_states**, **vm_states**, **vm_pause_details**, **disk_states**, **host_nic_states**, and **data_center_states**.

6.2. Permits

Permits are allowable actions a user assigns to a role. Each **permit** contains a set of properties.

Table 6.5. Permit properties

Element	Type	Description
id=	integer	The opaque identifier for a permit.
name	string	The name of the permit.
administrative	Boolean: true or false	The permit is assigned to only administrative roles.

```
<capabilities>
    ...
    <permits>
        <permit id="1">
            <name>create_vm</name>
            <administrative>false</administrative>
        </permit>
        <permit id="2">
            <name>delete_vm</name>
            <administrative>false</administrative>
        </permit>
        ...
    </permits>
    ...
</capabilities>
```

6.3. Scheduling Policies

The **scheduling_policies** element defines the load-balancing and power sharing modes for hosts in the cluster.

```
<capabilities>
    ...
    <scheduling_policies>
        <policy>evenly_distributed</policy>
        <policy>power_saving</policy>
    </scheduling_policies>
    ...
</capabilities>
```

6.4. Capabilities Example

The following example demonstrates a sample representation of **`capabilities`**.

Example 6.1. XML representation of capabilities

An API user performs the following request:

```
GET /api/capabilities HTTP/1.1
Accept: application/xml
```

The API returns the following representation:

```
HTTP/1.1 200 OK
Content-Type: application/xml

<capabilities>
    <version minor="0" major="3">
        <current>true</current>
        <features>
            <transparent_hugepages/>
        </features>
        <cpus>
            <cpu id="Intel Conroe Family">
                <level>3</level>
            </cpu>
            <cpu id="Intel Penryn Family">
                <level>4</level>
            </cpu>
            <cpu id="Intel Nehalem Family">
                <level>5</level>
            </cpu>
            <cpu id="Intel Westmere Family">
                <level>6</level>
            </cpu>
            <cpu id="AMD Opteron G1">
                <level>2</level>
            </cpu>
            <cpu id="AMD Opteron G2">
                <level>3</level>
            </cpu>
            <cpu id="AMD Opteron G3">
                <level>4</level>
            </cpu>
        </cpus>
        <power_managers>
            <power_management type="alom">
                <options>
                    <option type="bool" name="secure"/>
                    <option type="int" name="port"/>
                </options>
            </power_management>
            <power_management type="apc">
                <options>
                    <option type="bool" name="secure"/>
                    <option type="int" name="port"/>
                    <option type="int" name="slot"/>
                </options>
            </power_management>
            <power_management type="bladecenter">
                <options>
                    <option type="bool" name="secure"/>
                    <option type="int" name="port"/>
                    <option type="int" name="slot"/>
                </options>
            </power_management>
            ...
        </power_managers>
        <fence_types>
            <fence_type>manual</fence_type>
            <fence_type>restart</fence_type>
            <fence_type>start</fence_type>
            <fence_type>stop</fence_type>
            <fence_type>status</fence_type>
        </fence_types>
        <storage_types>
            <storage_type>iscsi</storage_type>
            <storage_type>fcp</storage_type>
            <storage_type>nfs</storage_type>
            <storage_type>localfs</storage_type>
        </storage_types>
        <storage_domain_types>
            <storage_domain_type>data</storage_domain_type>
            <storage_domain_type>iso</storage_domain_type>
            <storage_domain_type>export</storage_domain_type>
        </storage_domain_types>
        <vm_types>
            <vm_type>desktop</vm_type>
            <vm_type>server</vm_type>
        </vm_types>
        <boot_devices>
            <boot_device>cdrom</boot_device>
            <boot_device>hd</boot_device>
            <boot_device>network</boot_device>
        </boot_devices>
        <display_types>
            <display_type>vnc</display_type>
            <display_type>spice</display_type>
        </display_types>
        <nic_interfaces>
            <nic_interface>e1000</nic_interface>
            <nic_interface>virtio</nic_interface>
            <nic_interface>rtl8139</nic_interface>
            <nic_interface>rtl8139_virtio</nic_interface>
        </nic_interfaces>
        <disk_types>
            <disk_type>data</disk_type>
            <disk_type>system</disk_type>
        </disk_types>
        <os_types>
            <os_type>unassigned</os_type>
            <os_type>windows_xp</os_type>
            <os_type>windows_2003</os_type>
            <os_type>windows_2008</os_type>
            <os_type>other_linux</os_type>
            <os_type>other</os_type>
            <os_type>rhel_5</os_type>
            <os_type>rhel_4</os_type>
            <os_type>rhel_3</os_type>
            <os_type>windows_2003x64</os_type>
            <os_type>windows_7</os_type>
            <os_type>windows_7x64</os_type>
            <os_type>rhel_5x64</os_type>
            <os_type>rhel_4x64</os_type>
            <os_type>rhel_3x64</os_type>
            <os_type>windows_2008x64</os_type>
            <os_type>windows_2008r2x64</os_type>
            <os_type>rhel_6</os_type>
            <os_type>rhel_6x64</os_type>
        </os_types>
        <disk_formats>
            <disk_format>cow</disk_format>
            <disk_format>raw</disk_format>
        </disk_formats>
        <disk_interfaces>
            <disk_interface>ide</disk_interface>
            <disk_interface>virtio</disk_interface>
        </disk_interfaces>
        <vm_affinities>
            <affinity>migratable</affinity>
            <affinity>user_migratable</affinity>
            <affinity>pinned</affinity>
        </vm_affinities>
        <custom_properties>
            <custom_property regexp="^(true|false)$" name="sap_agent"/>
            <custom_property regexp="^[0-9]+$" name="sndbuf"/>
            <custom_property
              regexp="^(([a-zA-Z0-9_]*):(true|false))(,(([a-zA-Z0-9_]*):(true|false)))*$"
 name="vhost"/>
            <custom_property
              regexp="^(none|writeback|writethrough)$" name="viodiskcache"/>
        </custom_properties>
        <boot_protocols>
            <boot_protocol>dhcp</boot_protocol>
            <boot_protocol>static</boot_protocol>
        </boot_protocols>
        <error_handling>
            <on_error>migrate</on_error>
            <on_error>do_not_migrate</on_error>
            <on_error>migrate_highly_available</on_error>
        </error_handling>
        <storage_formats>
            <format>v1</format>
            <format>v2</format>
        </storage_formats>
    </version>
    ...
    <permits>
        <permit id="1">
            <name>create_vm</name>
            <administrative>false</administrative>
        </permit>
        <permit id="2">
            <name>delete_vm</name>
            <administrative>false</administrative>
        </permit>
        <permit id="3">
            <name>edit_vm_properties</name>
            <administrative>false</administrative>
        </permit>
        <permit id="4">
            <name>vm_basic_operations</name>
            <administrative>false</administrative>
        </permit>
        ...
    </permits>
    <scheduling_policies>
        <policy>evenly_distributed</policy>
        <policy>power_saving</policy>
    </scheduling_policies>
</capabilities>
```

Chapter 7.

Common Features

This chapter examines features common to resources and collections.

Note

Throughout this guide, the elements of each resource are detailed in tables. These tables include a properties column, displaying icons depicting element properties. The meaning of these icons is shown in *Table 7.1, “Element property icons”*

Table 7.1. Element property icons

Property	Description	Icon
Required for creation	These elements must be included in the client-provided representation of a resource on creation, but are not mandatory for an update of a resource.	⚠
Non-updateable	These elements cannot have their value changed when updating a resource. Include these elements in a client-provided representation on update only if their values are not altered by the API user. If altered, the API reports an error.	🔒
Read-only	These elements are read-only. Values for read-only elements are not created or modified.	⊘

7.1. Representations

The API structures resource representations in the following XML document structure:

```
<resource id="resource_id" href="/api/collection/resource_id">
    <name>Resource-Name</name>
    <description>A description of the resource</description>
    ...
</resource>
```

In the context of a virtual machine, the representation appears as follows:

```
<vm id="5b9bbce5-0d72-4f56-b931-5d449181ee06"
  href="/api/vms/5b9bbce5-0d72-4f56-b931-5d449181ee06">
    <name>RHEL6-Machine</name>
    <description>Red Hat Enterprise Linux 6 Virtual Machine</description>
    ...
</vm>
```

All resource representations contain a set of common attributes

Table 7.2. Common attributes to resource representations

Attribute	Type	Description	Properties
id	GUID	Each resource in the virtualization infrastructure contains an **id**, which	⚠

Attribute	Type	Description	Properties
		acts as a globally unique identifier (GUID). The GUID is the primary method of resource identification.	
`href`	string	The canonical location of the resource as an absolute path.	⚠

All resource representations contain a set of common elements.

Table 7.3. Common elements to resource representations

Element	Type	Description	Properties
`name`	string	A user-supplied human readable name for the resource. The **`name`** is unique across all resources of its type.	⚠
`description`	string	A free-form user-supplied human readable description of the resource.	

7.2. Collections

This section examines common features for collections.

7.2.1. Listing All Resources in a Collection

A listing of the resources in a collection is obtained by issuing a **`GET`** request on the collection URI obtained from the entry point.

```
GET /api/collection HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<collection>
    <resource id="resource_id" href="/api/collection/resource_id">
        <name>Resource-Name</name>
        <description>A description of the resource</description>
        ...
    </resource>
    ...
</collection>
```

7.2.2. Listing Extended Resource Sub-Collections

The API extends collection representations to include sub-collections when the **`Accept`** header includes the **`detail`** parameter.

```
GET /api/collection HTTP/1.1
Accept: application/xml; detail=subcollection
```

This includes multiple sub-collection requests using either separated **`detail`** parameters:

```
GET /api/collection HTTP/1.1
Accept: application/xml; detail=subcollection1; detail=subcollection2
```

Or one **detail** parameter that separates the sub-collection with the **+** operator:

```
GET /api/collection HTTP/1.1
Accept: application/xml; detail=subcollection1+subcollection2+subcollection3
```

The API supports extended sub-collections for the following main collections.

Table 7.4. Collections that use extended sub-collections

Collection	Extended Sub-Collection Support
hosts	**statistics**
vms	**statistics**, **nics**, **disks**

Example 7.1. An request for extended statistics, nics and disks sub-collections in the vms collection

```
GET /api/vms HTTP/1.1
Accept: application/xml; detail=statistics+nics+disks
```

7.2.3. Searching Collections with Queries

A **GET** request on a **"collection/search"** link results in a search query of that collection. The API only returns resources within the collection that satisfy the search query constraints.

```
GET /api/collection?search={query} HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<collection>
    <resource id="resource_id" href="/api/collection/resource_id">
        ...
    </resource>
    ...
</collection>
```

7.2.3.1. Query Syntax

The API uses the URI templates to perform a search **query** with a **GET** request:

```
GET /api/collection?search={query} HTTP/1.1
Accept: application/xml
```

The **query** template value refers to the search query the API directs to the **collection**. This **query** uses the same format as Red Hat Enterprise Virtualization Manager search query language:

(criteria) [sortby (element) asc|desc]

The **sortby** clause is optional and only needed when ordering results.

Table 7.5. Example search queries

Collection	Criteria	Result
`hosts`	`vms.status=up`	Displays a list of all hosts running virtual machines that are **`up`**.
`vms`	`domain=qa.company.com`	Displays a list of all virtual machines running on the specified domain.
`vms`	`users.name=mary`	Displays a list of all virtual machines belonging to users with the username **`mary`**.
`events`	`severity>normal sortby time`	Displays the list of all **`events`** with severity higher than **`normal`** and sorted by the **`time`** element values.
`events`	`severity>normal sortby time desc`	Displays the list of all **`events`** with severity higher than **`normal`** and sorted by the **`time`** element values in descending order.

The API requires the **`query`** template to be URL-encoded to translate reserved characters, such as operators and spaces.

Example 7.2. URL-encoded search query

```
GET /api/vms?search=name%3Dvm1 HTTP/1.1
Accept: application/xml
```

Important

All search queries are case-sensitive.

7.2.3.2. Wildcards

Search queries substitute part of a value with an asterisk as a wildcard.

Example 7.3. Wildcard search query for name=vm*

```
GET /api/vms?search=name%3Dvm* HTTP/1.1
Accept: application/xml
```

This query would result in all virtual machines with names beginning with **`vm`**, such as **`vm1`**, **`vm2`**, **`vma`** or **`vm-webserver`**.

Example 7.4. Wildcard search query for name=v*1

```
GET /api/vms?search=name%3Dv*1 HTTP/1.1
Accept: application/xml
```

This query would result in all virtual machines with names beginning with **`v`** and ending with **`1`**, such as **`vm1`**, **`vr1`** or **`virtualmachine1`**.

7.2.3.3. Pagination

Some Red Hat Enterprise Virtualization environments contain large collections of resources. However, the API only displays a default number of resources for one search query to a collection. To display more than the default, the API separates collections into pages via a search query containing the **`page`** command.

Example 7.5. Paginating resources

This example paginates resources in a collection. The URL-encoded request is:

```
GET /api/collection?search=page%201 HTTP/1.1
Accept: application/xml
```

Increase the **`page`** value to view the next page of results.

```
GET /api/collection?search=page%202 HTTP/1.1
Accept: application/xml
```

Use the **`page`** command also in conjunction with other commands in a search query. For example:

```
GET /api/collection?search=sortby%20element%20asc%20page%202 HTTP/1.1
Accept: application/xml
```

This query displays the second page in a collection listing ordered by a chosen element.

7.2.4. Creating a Resource in a Collection

The API creates a new resource with a **`POST`** request to the collection URI containing a representation of the new resource.

A **`POST`** request requires a **`Content-Type: application/xml`** header. This informs the API of the XML representation in the body content as part of the request.

Each resource type has its own specific required properties. The client supplies these properties as XML elements when creating a new resource. Refer to the individual resource type documentation for more details. If a required property is absent, the creation fails with a **`fault`** representation indicating the missing elements.

```
POST /api/collection HTTP/1.1
Accept: application/xml
Content-Type: application/xml
```

```
<resource>
    <name>Resource-Name</name>
</resource>

HTTP/1.1 201 Created
Content-Type: application/xml

<resource id="resource_id" href="/api/collection/resource_id">
    <name>Resource-Name</name>
    ...
</resource>
```

The **`Location`** header in the response gives the URI of the queried resource. The response body contains either a complete representation, partial representation or no representation of the resource. It is recommended that clients rely only on fetching the representation via the URI in the response header.

7.2.4.1. Asynchronous Requests

The API performs asynchronous **`POST`** requests unless the user overrides them with a **`Expect: 201-created`** header.

For example, certain resources, such as Virtual Machines, Disks, Snapshots and Templates, are created asynchronously. A request to create an asynchronous resource results in a **`202 Accepted`** status. The initial document structure for a **`202 Accepted`** resource also contains a **`creation_status`** element and link for creation status updates. For example:

```

POST /api/collection HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<resource>
    <name>Resource-Name</name>
</resource>

HTTP/1.1 202 Accepted
Content-Type: application/xml

<resource id="resource_id" href="/api/collection/resource_id">
    <name>Resource-Name</name>
    <creation_status>
        <state>pending</state>
    </creation status>
    <link rel="creation_status"
      href="/api/collection/resource_id/creation_status/creation_status_id"/>
      ...
</resource>
```

A **`GET`** request to the **`creation_status`** link provides a creation status update:

```

GET /api/collection/resource_id/creation_status/creation_status_id HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<creation id="creation_status_id"
  href="/api/collection/resource_id/creation_status/creation_status_id">
    <status>
```

```
        <state>complete</state>
    </status>
</creation>
```

Overriding the asynchronous resource creation requires an **`Expect: 201-created`** header:

```
POST /api/collection HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Expect: 201-created

<resource>
    <name>Resource-Name</name>
</resource>
```

7.3. Resources

This section examines common features for resources.

7.3.1. Retrieving a Resource

The API retrieves the state of a resource with a **`GET`** request on a URI obtained from a collection listing.

```
GET /api/collection/resource_id HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<resource id="resource_id" href="/api/collection/resource_id">
    ...
</resource>
```

7.3.2. Updating a Resource

The API modifies resource properties with a **`PUT`** request containing an updated description from a previous **`GET`** request for the resource URI. Details on modifiable properties are found in the individual resource type documentation.

A **`PUT`** request requires a **`Content-Type: application/xml`** header. This informs the API of the XML representation in the body content as part of the request.

```
PUT /api/collection/resource_id HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<resource>
    <name>New-Resource-Name</name>
</resource>

HTTP/1.1 200 OK
Content-Type: application/xml

<resource id="resource_id" href="/api/collection/resource_id">
    <name>New-Resource-Name</name>
    ...
</resource>
```

This does not include immutable resource properties that an API user has attempted to modify. If an attempt is made to modify a *strictly* immutable resource property, the API reports a **`409 Conflict`** error with a **`fault`** representation in the response body.

Properties omitted from the representation are ignored and not changed.

7.3.3. Deleting a Resource

The API deletes a resource with a **`DELETE`** request sent to its URI.

```

DELETE /api/collection/resource_id HTTP/1.1
Accept: application/xml

HTTP/1.1 204 No Content
```

Some cases require optional body content in the **`DELETE`** request to specify additional properties. A **`DELETE`** request with optional body content requires a **`Content-Type: application/xml`** header to inform the API of the XML representation in the body content. If a **`DELETE`** request contains no body content, omit the **`Content-Type: application/xml`** header.

7.3.4. Sub-Collection Relationships

A sub-collection relationship defines a hierarchical link between a resource and a sub-collection. The sub-collection exists or has some meaning in the context of a parent resource. For example, a virtual machine contains network interfaces, which means the API maps the relationship between the virtual machine resource and the network interfaces sub-collection.

Sub-collections are used to model the following relationships types:

- 1:N mappings, where mapped resources are dependent on a parent resources. Without the parent resource, the dependent resource cannot exist. For example, the link between a virtual machine and its disk resources.

- 1:N mappings, where mapped resources exist independently from parent resources but data is still associated with the relationship. For example, the link between a network and a cluster.

- N:M mappings, where one mapped resources only belongs to one parent resource. For example, the link between a storage domain and a data center.

The API defines a relationship between a resource and a sub-collection using the **`link rel=`** attribute:

```

GET /api/collection/resource_id HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<resource id="resource_id" href="/api/collection/resource_id">
    ...
    <link rel="subcollection"
      href="/api/collection/resource_id/subcollection"/>
    ...
```

```
</resource>
```

The API user now queries the sub-collection.

```
GET /api/collection/resource_id/subcollection HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<subcollection>
    <subresource id="subresource_id"
      href="/api/collection/resource_id/subcollection/subresource_id">
        ...
    </subresource>
    ...
</subcollection>
```

7.3.5. XML Element Relationships

XML element links act as an alternative to sub-collections to express relationships between resources. XML element links are simply elements with a "href" attribute that points to the linked element.

XML element links are used to model simple 1:N mappings between resources without a dependency and without data associated with the relationship. For example, the relationship between a host and a cluster.

Examples of such relationships include:

- Backlinks from a resource in a sub-collection to a parent resource; or
- Links between resources with an arbitrary relationship.

Example 7.6. Backlinking from a sub-collection resource to a resource using an XML element

```
GET /api/collection/resource_id/subcollection/subresource_id HTTP/1.1

HTTP/1.1 200 OK
Content-Type: application/xml

<subcollection>
    <subresource id="subresource_id"
      href="/api/collection/resource_id/subcollection/subresource_id">
        <resource id="resource_id" href="/api/collection/resource_id"/>
        ...
    </subresource>
</subcollection>
```

7.3.6. Actions

Most resources include a list of action links to provide functions not achieved through the standard HTTP methods.

```
<resource>
    ...
    <actions>
```

```
        <link rel="start" href="/api/collection/resource_id/start"/>
        <link rel="stop" href="/api/collection/resource_id/stop"/>
        ...
    </actions>
    ...
</resource>
```

The API invokes an action with a **`POST`** request to the supplied URI. The body of the **`POST`** requires an **`action`** representation encapsulating common and task-specific parameters.

Table 7.6. Common action parameters

Element	Description
`async`	**`true`** if the server responds immediately with **`202 Accepted`** and an action representation contains a **`href`** link to be polled for completion.
`grace_period`	a grace period in milliseconds, which must expire before the action is initiated.

Individual actions and their parameters are documented in the individual resource type's documentation. Some parameters are mandatory for specific actions and their absence is indicated with a **`fault`** response.

An action also requires a **`Content-Type: application/xml`** header since the **`POST`** request requires an XML representation in the body content.

When the action is initiated asynchronously, the immediate **`202 Accepted`** response provides a link to monitor the status of the task:

```
POST /api/collection/resource_id/action HTTP/1.1
Content-Type: application/xml
Accept: application/xml

<action>
    <async>true</async>
</action>

HTTP/1.1 202 Accepted
Content-Type: application/xml

<action id="action_id"
  href="/api/collection/resource_id/action/action_id">
    <async>true</async>
    ...
<action>
```

A subsequent **`GET`** on the action URI provides an indication of the status of the asynchronous task.

Table 7.7. Action statuses

Status	Description
`pending`	Task has not yet started.
`in_progress`	Task is in operation.
`complete`	Task completed successfully.
`failed`	Task failed. The returned **`action`** representation would contain a **`fault`** describing the failure.

Once the task has completed, the action is retained for an indeterminate period. Once this has expired, subsequent **`GET`**s are **`301 Moved Permanently`** redirected back to the target resource.

```
GET /api/collection/resource_id/action/action_id HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<action id="action_id"
  href="/api/collection/resource_id/action/action_id">
    <status>
        <state>pending</state>
    </status>
    <link rel="parent" /api/collection/resource_id"/>
    <link rel="replay" href="/api/collection/resource_id/action"/>
<action>
```

An action representation also includes some links that are identified by the **rel** attribute:

Table 7.8. Action relationships

Type	Description
parent	A link back to the resource of this action
replay	A link back to the original action URI. POSTing to this URI causes the action to be re-initiated

7.3.7. Permissions

Each resource contains a **permissions** sub-collection. Each **permission** contains a **user**, an assigned **role** and the specified resource. For example:

```
GET /api/collection/resource_id/permissions HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<permissions>
    <permission id="permission-id"
      href="/api/collection/resource_id/permissions/permission_id">
        <role id="role_id" href="/api/roles/role_id"/>
        <user id="user_id" href="/api/users/user_id"/>
        <resource id="resource_id" href="/api/collection/resource_id"/>
    </permission>
    ...
</permissions>
```

A resource acquires a new permission when an API user sends a **POST** request with a **permission** representation and a **Content-Type: application/xml** header to the resource's **permissions** sub-collection. Each new permission requires a **role** and a **user**:

```
POST /api/collection/resource_id/permissions HTTP/1.1
Content-Type: application/xml
Accept: application/xml

<permission>
    <role id="role_id"/>
    <user id="user_id"/>
</permission>
```

```
HTTP/1.1 201 Created
Content-Type: application/xml

<permission id="permission_id"
  href="/api/resources/resource_id/permissions/permission_id">
    <role id="role_id" href="/api/roles/role_id"/>
    <user id="user_id" href="/api/users/user_id"/>
    <resource id="resource_id" href="/api/collection/resource_id"/>
</permission>
```

7.3.8. Handling Errors

Some errors require further explanation beyond a standard HTTP status code. For example, the API reports an unsuccessful resource state update or action with a **`fault`** representation in the response entity body. The fault contains a **`reason`** and **`detail`** strings. Clients must accomodate failed requests via extracting the **`fault`** *or* the expected resource representation depending on the response status code. Such cases are clearly indicated in the individual resource documentation.

```

PUT /api/collection/resource_id HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<resource>
    <id>id-update-test</id>
</resource>

HTTP/1.1 409 Conflict
Content-Type: application/xml

<fault>
    <reason>Broken immutability constraint</reason>
    <detail>Attempt to set immutable field: id</detail>
</fault>
```

Chapter 8.

Data Centers

The **datacenters** collection provides information about the data centers in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="datacenters"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a data center resource representation.

> **Note**
>
> The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 8.1. Data center elements

Element	Type	Description	Properties
link rel="storagedomains"	relationship	A link to the sub-collection for storage domains attached to this data center.	
link rel="permissions"	relationship	A link to the sub-collection for data center permissions. See *Section 7.3.7, “ Permissions ”*.	
storage_type	enumerated	Describes the storage type in this datacenter. A list of enumerated values is available in **capabilities**. See *Section 6.1.6, “Storage Types”*.	⚠ 🔒
storage_format	enumerated	Describes the storage format version for the data center. A list of enumerated values are available in **capabilities**. See *Section 6.1.20, “Storage Formats”*.	⚠ 🔒
version major= minor=	complex	The compatibility level of the data center. See *Chapter 5, Compatibility Level Versions*.	⚠ 🔒
supported_versions	complex	A list of possible **version** levels for the data center. See *Chapter 5, Compatibility Level Versions*.	⊘
status	see below	The data center status.	⊘

The **status** contains one of the following enumerative values: **uninitialized**, **up**, **maintenance**, **not_operational**, **problematic** and **contend**. These states are listed in **data_center_states** under **capabilities** (See *Section 6.1.21, “Resource Status States”*).

Example 8.1. An XML representation of a data center

```
<data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"
  href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988">
    <name>Default</name>
    <description>The default Data Center</description>
    <link rel="storagedomains"
      href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/
      storagedomains"/>
    <link rel="permissions"
      href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988/permissions"/>
    <storage_type>nfs</storage_type>
    <storage_format>v1</storage_format>
    <version minor="0" major="3"/>
    <supported_versions>
        <version minor="0" major="3"/>
    </supported_versions>
    <status>
        <state>up</state>
    </status>
</data_center>
```

Creation of a new data center requires the **name**, **storage_type** and **version** elements. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 8.2. Creating a data center

```
POST /api/datacenters HTTP/1.1
Accept: application/xml
Content-type: application/xml

<data_center>
    <name>NewDatacenter</name>
    <storage_type>nfs</storage_type>
    <version minor="0" major="3"/>
</data_center>
```

The **name** and **description** elements are updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 8.3. Updating a data center

```
PUT /api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<data_center>
    <name>UpdatedName</name>
    <description>An updated description for the data center</description>
</data_center>
```

Removal of a data center requires a **DELETE** request.

Example 8.4. Removing a data center

```
DELETE /api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988 HTTP/1.1
```

```
HTTP/1.1 204 No Content
```

8.1. Storage Domains Sub-Collection

Each data center contains a sub-collection for attached storages domain. An API user interacts with this sub-collection using the standard REST methods as per *Chapter 7, Common Features*.

An attached storage domain has a similar representation to a top-level storage domain, with the exception that it has a data center specific **status** and set of actions. States for the **status** element are listed in **storage_domain_states** under **capabilities** (See *Section 6.1.21, “Resource Status States”*)

Important

The API as documented in this section is experimental and subject to change. It is not covered by the backwards compatibility statement in *Section 6, “Backwards Compatibility Statement”*.

8.1.1. Attaching a Storage Domain

A data center is only ready for use when at least one storage domain is attached, which an API user **POST**s to the data center's storage domains sub-collection.

When attaching a storage domain, its **id** or **name** must be supplied. An example of attaching a storage domain to a data center:

Example 8.5. Attach a storage domain to a data center

```
POST /api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"/>

HTTP/1.1 201 Created
Location: /datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains/
fabe0451-701f-4235-8f7e-e20e458819ed
Content-Type: application/xml

<storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"
  href="/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains/
  fabe0451-701f-4235-8f7e-e20e458819ed">
    <name>images0</name>
    <type>data</type>
    <status>
        <state>inactive</state>
    </status>
    <master>true</master>
    <storage>
        <type>nfs</type>
        <address>172.31.0.6</address>
        <path>/exports/RHEVX/images/0</path>
    </storage>
    <data_center id="d70d5e2d-b8ad-494a-a4d2-c7a5631073c4"
      href="/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4"/>
    <actions>
        <link rel="activate"
```

```
          href="/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/
          storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/activate"/>
        <link rel="deactivate"
          href="/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/
          storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/deactivate"/>
    </actions>
</storage_domain>
```

8.1.2. Actions

There are two possible actions for attached storage domains: **`activate`** and **`deactivate`**.

8.1.2.1. Activate Action

An attached storage domain requires activation on a data center before use. The activate action does not take any action specific parameters.

Example 8.6. Action to active a storage domain on a datacenter

```
POST /api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains/
fabe0451-701f-4235-8f7e-e20e458819ed/activate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

8.1.2.2. Deactivate Action

An attached storage domain is deactivated on a data center before removal. The deactivate action does not take any action specific parameters.

Example 8.7. Action to deactivate a storage domain on a datacenter

```
POST /api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4/storagedomains/
fabe0451-701f-4235-8f7e-e20e458819ed/deactivate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

8.2. Force Remove Action

An API user forces the removal of a data center when encountering unresolvable problems with storage domains, such as the loss of connection to a master storage domain or a lack of available hosts when deleting storage domains. The API includes a **`force`** action to help with these situations.

This action removes database entries associated with a chosen data center before the API removes the data center from the Red Hat Enterprise Virtualization environment. This means the API removes the data center regardless of associated storage domains.

This action requires a **`DELETE`** method. The request body contains an **`action`** representation with the **`force`** parameter set to **`true`**. The request also requires an additional **`Content-type: application/xml`** header to process the XML representation in the body.

Example 8.8. Force remove action on a data center

```
DELETE /api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
  <force>true</force>
</action>
```

This action:

- Deletes all database information for **`data`** storage domains associated the data center;
- Deletes all database information for resources, such as virtual machines and templates, on **`data`** storage domains associated the data center;
- Detaches **`iso`** and **`export`** storage domains from the data center; and
- Deletes the database information for the data center.

This action overrides the requirement for a data center to be empty before deletion.

Important

This action only removes the database entries for resources associated with the data center. The **`data`** storage domains associated with the data center require manual format before reuse. Metadata for **`iso`** and **`export`** domains require manual cleaning prior to use on another data center.

Chapter 9.

Host Clusters

The **clusters** collection provides information about host clusters in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="clusters"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a host cluster resource representation.

> **Note**
>
> The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 9.1. Host cluster elements

Element	Type	Description	Properties
link rel="networks"	relationship	A link to the sub-collection for networks associated with this cluster.	
link rel="permissions"	relationship	A link to the sub-collection for cluster permissions. See *Section 7.3.7, “ Permissions ”*.	
cpu id=	complex	A server CPU reference that defines the CPU type all hosts must support in the cluster. See *Section 6.1.3, “CPUs”*.	⚠
data_center id=	GUID	A reference to the data center membership of this cluster. See *Chapter 8, Data Centers*.	⚠ 🔒
memory_policy	complex	Defines the cluster's policy on host memory utilization. See *Table 9.2, “Memory policy elements”*.	🔒
scheduling_policy	complex	Defines the load-balancing or power sharing modes for hosts in the cluster. See *Table 9.3, “Scheduling policy elements”*.	🔒
version major= minor=	complex	The compatibility level of the cluster. See *Chapter 5, Compatibility Level Versions*.	🔒
supported_versions	complex	A list of possible **version** levels for the cluster. See *Chapter 5, Compatibility Level Versions*.	⊘
error_handling	complex/enumerated	Defines virtual machine handling when a host within a cluster becomes non-operational. Requires a single **on_error** element containing an enumerated type property listed in **capabilities**. See *Section 6.1.19, “Error Handling”*.	

The **memory_policy** element contains the following elements:

Table 9.2. Memory policy elements

Element	Type	Description	Properties
overcommit percent=	complex	The percentage of host memory allowed in use before no more virtual machines can start on a host. Virtual machines can use more than the available host memory due to memory sharing under KSM. Recommended values include **100** (None), **150** (Server Load) and **200** (Desktop Load).	🔒
transparent_hugepages	complex	Define the **enabled** status of Transparent Hugepages. The status is either true or false. Check **capabilities** feature set (see *Section 6.1.2, “Features”*) to ensure your version supports **transparent hugepages**.	🔒

The **scheduling_policy** element contains the following elements:

Table 9.3. Scheduling policy elements

Element	Type	Description	Properties
policy	enumerated	The VM scheduling mode for hosts in the cluster. A list of enumerated types are listed in **capabilities**. See *Section 6.3, “Scheduling Policies”*.	🔒
thresholds low= high= duration=	complex	Defines CPU limits for the host. The **high** attribute controls the highest CPU usage percentage the host can have before being considered overloaded. The **low** attribute controls the lowest CPU usage percentage the host can have before being considered underutilized. The **duration** attribute refers to the number of seconds the host needs to be overloaded before the scheduler starts and moves the load to another host.	🔒

Example 9.1. An XML representation of a host cluster

```
<cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
  href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95">
    <name>Default</name>
    <description>The default server cluster</description>
    <link rel="networks"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks"/>
    <link rel="permissions"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/permissions"/>
    <cpu id="Intel Penryn Family"/>
    <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"
      href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988"/>
    <memory_policy>
        <overcommit percent="100"/>
        <transparent_hugepages>
            <enabled>false</enabled>
        </transparent_hugepages>
    </memory_policy>
    <scheduling_policies>
      <policy>evenly_distributed</policy>
      <thresholds low="10" high="75" duration="120"/>
    </scheduling_policies>
    <version minor="0" major="3"/>
    <supported_versions>
        <version minor="0" major="3"/>
    </supported_versions>
    <error_handling>
        <on_error>migrate</on_error>
    </error_handling>
</cluster>
```

Creation of a new cluster requires the **name**, **cpu id=** and **datacenter** elements. Identify the **datacenter** with either the **id** attribute or **name** element. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 9.2. Creating a host cluster

```
POST /api/clusters HTTP/1.1
Accept: application/xml
Content-type: application/xml

<cluster>
    <name>cluster1</name>
    <cpu id="Intel Penryn Family"/>
    <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"/>
</cluster>
```

The **name**, **description**, **cpu id=** and **error_handling** elements are updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 9.3. Updating a host cluster

```
PUT /api/clusters/99408929-82cf-4dc7-a532-9d998063fa95 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<cluster>
    <description>Cluster 1</description>
</cluster>
```

Removal of a cluster requires a **DELETE** request.

Example 9.4. Removing a cluster

```
DELETE /api/clusters/99408929-82cf-4dc7-a532-9d998063fa95 HTTP/1.1

HTTP/1.1 204 No Content
```

9.1. Networks Sub-Collection

Networks associated with a cluster are represented with the **networks** sub-collection. Every host within a cluster connects to these associated networks.

The representation of a cluster's **network** sub-collection is the same as a standard **network** resource with an additional **cluster id=** to signify a relationship to the cluster and a **display** element to represent the display network status in the cluster.

An API user manipulates the **networks** sub-collection as described in *Chapter 7, Common Features*. **POST**ing a network **id** or **name** reference to the **networks** sub-collection associates the network with the cluster.

Example 9.5. Associating a network resource with a cluster

```
POST /api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<network>
    <name>rhevm</name>
</network>

HTTP/1.1 201 Created
Location: http://{host}/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks/
da05ac09-00be-45a1-b0b5-4a6a2438665f
Content-Type: application/xml

<network id="da05ac09-00be-45a1-b0b5-4a6a2438665f"
  href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks/
  da05ac09-00be-45a1-b0b5-4a6a2438665f">
    <name>rhevm</name>
    <status>
        <state>operational</state>
    </status>
    <description>Display Network</description>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <data_center id="d70d5e2d-b8ad-494a-a4d2-c7a5631073c4"
      href="/api/datacenters/d70d5e2d-b8ad-494a-a4d2-c7a5631073c4"/>
    <display>true</display>
</network>
```

The display network status is set using a **PUT** request to specify the Boolean value (true or false) of the **display** element.

Example 9.6. Setting the display network status

```
PUT /api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks/da05ac09-00be-45a1-
b0b5-4a6a2438665f HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<network>
```

```
    <display>false</display>
</network>
```

An association is removed with a **DELETE** request to the appropriate element in the collection.

Example 9.7. Removing a network association from a cluster

```

DELETE /api/clusters/99408929-82cf-4dc7-a532-9d998063fa95/networks/da05ac09-00be-45a1-
b0b5-4a6a2438665f HTTP/1.1

HTTP/1.1 204 No Content
```

Chapter 10.

Networks

The **networks** collection provides information about the logical networks in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="networks"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a network resource representation.

> **Note**
>
> The icons used in the properties column of this table are described in *Table 7.1, "Element property icons"*

Table 10.1. Network elements

Element	Type	Description	Properties
data_center id=	GUID	A reference to the data center of which this cluster is a member. See *Chapter 8, Data Centers*.	⚠ 🔒
vlan id=	integer	A VLAN tag.	
stp	Boolean: true or false	**true** if Spanning Tree Protocol is enabled on this network.	
status	One of **operational** or **non_operational**	The status of the network. These states are listed in **network_states** under **capabilities** (See *Section 6.1.21, "Resource Status States"*).	⊘

Example 10.1. An XML representation of a network resource

```
<network id="00000000-0000-0000-0000-000000000009"
  href="/api/networks/00000000-0000-0000-0000-000000000009">
    <name>rhevm</name>
    <description>Management Network</description>
    <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"
      href="/api/datacenters/01a45ff0-915a-11e0-8b87-5254004ac988"/>
    <stp>false</stp>
    <status>
        <state>operational</status>
    </status>
    <display>false</display>
</network>
```

Creation of a new data center requires the **name** and **datacenter** elements. See *Section 7.2.4, " Creating a Resource in a Collection "* for more information.

Example 10.2. Creating a network resource

```
POST /api/networks HTTP/1.1
```

```
Accept: application/xml
Content-type: application/xml

<network>
    <name>network 1</name>
    <data_center id="01a45ff0-915a-11e0-8b87-5254004ac988"/>
</network>
```

The **name**, **description**, **ip**, **vlan**, **stp** and **display** elements are updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 10.3. Updating a network resource

```
PUT /api/networks/e6575a87-377c-4f67-9c1b-7b94eff76b17 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<network>
    <description>Network 1</description>
</network>
```

Removal of a network requires a **DELETE** request.

Example 10.4. Removing a network

```
DELETE /api/networks/e6575a87-377c-4f67-9c1b-7b94eff76b17 HTTP/1.1

HTTP/1.1 204 No Content
```

Important

The API as documented in this section is experimental and subject to change. It is not covered by the backwards compatibility statement in *Section 6, “Backwards Compatibility Statement”*.

Chapter 11.

Storage Domains

The **storagedomains** collection provides information about the storage domains in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="storagedomains"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a storage domain resource representation.

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 11.1. Storage domain elements

Element	Type	Description	Properties
link rel="permissions"	relationship	A link to the sub-collection for storage domain permissions. See *Section 7.3.7, “ Permissions ”*.	
link rel="files"	relationship	A link to the **files** sub-collection for this storage domains.	
link rel="vms"	relationship	A link to the **vms** sub-collection for a storage domain with **type** set to **export**.	
link rel="templates"	relationship	A link to the **templates** sub-collection for a storage domain with **type** set to **export**.	
type	enumerated	The storage domain type. A list of enumerated values are available in **capabilities**. See *Section 6.1.7, “Storage Domain Types”*.	⚠ 🔒
master	Boolean: true or false	**true** if this is the master storage domain of a data center.	🔒
host	complex	A reference to the host on which this storage domain should be initialized. The only restriction on this host is that it should have access to the physical storage specified.	⚠ 🔒
storage	complex	Describes the underlying storage of the storage domain. For more information see *Section 11.1, “ Storage types ”*.	⚠ 🔒
available	integer	Space available in bytes.	⊘
used	integer	Space used in bytes.	⊘

Element	Type	Description	Properties
`committed`	integer	Space committed in bytes.	⊘
`storage_format`	enumerated	Describes the storage format version for the storage domain. A list of enumerated values are available in **`capabilities`**. See *Section 6.1.20, “Storage Formats”*.	⚠ 🔒

Example 11.1. An XML representation of a storage domain

```
<storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"
  href="/api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed">
    <name>data0</name>
    <link rel="permissions"
      href="/api/storagedomains/be24cd98-8e23-49c7-b425-1a12bd12abb0/permissions"/>
    <link rel="files"
      href="/api/storagedomains/be24cd98-8e23-49c7-b425-1a12bd12abb0/files"/>
    <type>data</type>
    <master>true</master>
    <storage>
        <type>nfs</type>
        <address>172.31.0.6</address>
        <path>/exports/RHEVX/images/0</path>
    </storage>
    <available>156766306304</available>
    <used>433791696896</used>
    <committed>617401548800</committed>
    <storage_format>v1</storage_format>
</storage_domain>
```

Creation of a new storage domain requires the **`name`**, **`type`**, **`host`** and **`storage`** elements. Identify the **`host`** element with the **`id`** attribute or **`name`** element. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 11.2. Creating a storage domain

```
POST /api/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain>
    <name>data1</name>
    <type>data</type>
    <host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"/>
    <storage>
        <type>nfs</type>
        <address>172.31.0.6</address>
        <path>/exports/RHEVX/images/0</path>
    </storage>
</storage_domain>
```

Only the **`name`** element is updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 11.3. Updating a storage domain

```
PUT /api/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml

<storage_domain>
    <name>data2</name>
    ...
</storage_domain>
```

Removal of a storage domain requires a **`DELETE`** request.

Example 11.4. Removing a storage domain

```
DELETE /api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed HTTP/1.1

HTTP/1.1 204 No Content
```

The API user attaches the storage domain to a data center after creation. See *Section 8.1.1, “Attaching a Storage Domain”* for instructions.

Important

The API as documented in this chapter is experimental and subject to change. It is not covered by the backwards compatibility statement in *Section 6, “Backwards Compatibility Statement”*.

11.1. Storage types

The **`storage`** element contains a **`type`** element, which is an enumerated value found under the **`capabilities`** collection. See *Section 6.1.6, “Storage Types”*.

The storage element also contains additional elements specific to each storage **`type`**. The next few section examine these additional storage **`type`** elements.

11.1.1. NFS

The following table contains **`nfs`** specific elements in a **`storage`** description.

Table 11.2. NFS specific elements

Element	Type	Description	Properties
`address`	string	The host name or IP address of the NFS server.	🔒
`path`	string	The path of NFS mountable directory on the server.	🔒

11.1.2. iSCSI and FCP

The following table contains **`iscsi`** and **`fcp`** specific elements in a **`storage`** description.

Table 11.3. iSCSI and FCP specific elements

Element	Type	Description	Properties
`logical_unit id=`	complex	The **`id`** of the logical unit. A storage domain also accepts multiple iSCSI or FCP logical units.	🔒

The **`logical_unit`** contains a set of sub-elements.

Table 11.4. Logical unit elements

Element	Type	Description	Properties
`address`	string	The address of the server containing the storage device.	🔒
`port`	integer	The port number of the server.	🔒
`target`	string	The target IQN for the storage device.	🔒
`username`	string	A CHAP user name for logging into a target.	🔒
`password`	string	A CHAP password for logging into a target.	🔒
`serial`	string	The serial ID for the target.	🔒
`vendor_id`	string	The vendor name for the target.	🔒
`product_id`	string	The product code for the target.	🔒
`lun_mapping`	integer	The Logical Unit Number device mapping for the target.	🔒

In the case of iSCSI, if a **`logical_unit`** description also contains details of the iSCSI target with the LUN in question, the target performs an automatic login when the storage domain is created.

11.1.3. LocalFS

The **`localfs`** specific elements in a **`storage`** description are:

Table 11.5. Localfs specific elements

Element	Type	Description	Properties
`path`	string	The path of local storage domain on the host.	🔒

A **`localfs`** storage domain requires a data center with **`storage_type`** set to **`localfs`** (see *Chapter 8, Data Centers*). This data center only contains a single host cluster, and the host cluster only contains a single host.

11.2. Export Storage Domains

Storage domains with **`type`** set to **`export`** contain **`vms`** and **`templates`** sub-collections, which list the import candidate VMs and templates stored on that particular storage domain.

Example 11.5. Listing the virtual machines sub-collection of an export storage domain

```
GET /api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/vms
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<vms>
    <vm id="082c794b-771f-452f-83c9-b2b5a19c0399"
      href="/api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/
      vms/082c794b-771f-452f-83c9-b2b5a19c0399">
        <name>vm1</name>
        ...
        <storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"
          href="/api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed"/>
        <actions>
            <link rel="import" href="/api/storagedomains/
              fabe0451-701f-4235-8f7e-e20e458819ed/vms/
              082c794b-771f-452f-83c9-b2b5a19c0399/import"/>
        </actions>
    </vm>
</vms>
```

VMs and templates in these collections have a similar representation to their counterparts in the top-level VMs and templates collection, except they also contain a **`storage_domain`** reference and an **`import`** action.

The **`import`** action imports a virtual machine or a template from an **`export`** storage domain. The destination cluster and storage domain is specified with **`cluster`** and **`storage_domain`** references.

Example 11.6. Action to import a virtual machine from an export storage domain

```
POST /api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/vms/
082c794b-771f-452f-83c9-b2b5a19c0399/import HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <storage_domain>
        <name>images0</name>
    </storage_domain>
    <cluster>
        <name>Default</name>
    </cluster>
</action>
```

Example 11.7. Action to import a template from an export storage domain

```
POST /api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/templates/
082c794b-771f-452f-83c9-b2b5a19c0399/import HTTP/1.1
Accept: application/xml
```

```
Content-type: application/xml

<action>
    <storage_domain>
        <name>images0</name>
    </storage_domain>
    <cluster>
        <name>Default</name>
    </cluster>
</action>
```

Delete a virtual machine or template from an **export** storage domain with a **DELETE** request.

Example 11.8. Delete virtual machine from an export storage domain

```
DELETE /api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed/vms/
082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1
Accept: application/xml

HTTP/1.1 204 No Content
```

11.3. Files Sub-Collection

The **files** sub-collection under each storage domain provides a way for clients to list available files. This sub-collection is specifically targeted to ISO storage domains, which contain ISO images and virtual floppy disks (VFDs) that an administrator uploads through Red Hat Enterprise Virtualization Manager.

The addition of a CD-ROM device to a VM requires an ISO image from the **files** sub-collection of an ISO storage domain.

Example 11.9. Listing the files sub-collection of an ISO storage domain

```
GET /api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/files HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<files>
    <file id="en_winxp_pro_with_sp2.iso"
      href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/files/
      en_winxp_pro_with_sp2.iso">
        <name>en_winxp_pro_with_sp2.iso</name>
        <type>iso</type>
        <storage_domain id="00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"
          href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"/>
    </file>
    <file id="boot.vfd"
      href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da/files/
      boot.vfd">
        <name>boot.vfd</name>
        <type>vfd</type>
        <storage_doman id="00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"
          href="/api/storagedomains/00f0d9ce-da15-4b9e-9e3e-3c898fa8b6da"/>
    </file>
</files>
```

Like other resources, files have opaque **id** and **href** attributes. The **name** element contains the filename.

11.4. Import Existing Storage Domain

The API provides a user with the ability to remove an ISO or Export storage domain from one Red Hat Enterprise Virtualization Manager instance without re-formatting the underlying storage and import it into another instance. Importing is achieved similarly to adding a new storage domain, except the **name** is not specified.

Example 11.10. Importing an existing export storage domain

```
POST /api/storagedomains HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<storage_domain>
    <type>export</type>
    <storage>
        <type>nfs</type>
        <address>172.31.0.6</address>
        <path>/exports/RHEVX/export-domain</path>
    </storage>
    <host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"/>
</storage_domain>

HTTP/1.1 201 Created
Content-Type: application/xml

<storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"
  href="/api/storagedomains/fabe0451-701f-4235-8f7e-e20e458819ed">
    <name>export1</name>
    ...
</storage_domain>
```

11.5. Delete Storage Domain

A **storage_domain** reference is passed in the body of a **DELETE** request for a storage domain. The **storage_domain** reference is in the following form:

```
<storage_domain>
    <host id="..."/>
</storage_domain>
```

OR

```
<storage_domain>
    <host>
        <name>...</name>
    </host>
</storage_domain>
```

The API user provides a optional **format** element to specify whether or not to format the storage domain after deletion.

Example 11.11. Formatting a storage domain after deletion

```
<storage_domain>
    <host id="..."/>
    <format>true</format>
</storage_domain>
```

If no **`format`** element is passed, the storage domain remains unformatted.

Chapter 12.

Hosts

The **hosts** collection provides information about the hosts in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="hosts"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a host resource representation.

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 12.1. Host elements

Element	Type	Description	Properties
link rel="storage"	relationship	A link to the **storage** sub-collection for host storage.	⊘
link rel="nics"	relationship	A link to the **nics** sub-collection for host network interfaces.	
link rel="tags"	relationship	A link to the **tags** sub-collection for host tags.	
link rel="permissions"	relationship	A link to the **permissions** sub-collection for host permissions. See *Section 7.3.7, “ Permissions ”*.	
link rel="statistics"	relationship	A link to the **statistics** sub-collection for host statistics.	⊘
type	One of **rhel** or **rhev-h**	The host type.	⊘
address	string	The IP address or hostname of the host.	⚠ 🔒
status	See below	The host status.	⊘
cluster id=	GUID	A reference to the cluster that includes this host.	
port	integer	The listen port of the VDSM daemon running on this host.	🔒
storage_manager	Boolean: true or false	**true** if the host is the storage pool manager (SPM).	🔒
power_management	complex	Configuration options for host power management. See *Section 12.1, “Power Management”*.	
ksm	Boolean: true or false	**true** if Kernel SamePage Merging (KSM) is enabled.	

Element	Type	Description	Properties
`transparent_hugepages`	Boolean: true or false	**`true`** if Transparent Hugepages is enabled.	
`iscsi`	complex	The SCSI **`initiator`** for the host.	🔒
`cpu`	complex	Statistics for the host CPU. Includes sub-elements for the CPU's **`name`**, **`topology cores=`** and **`speed`**.	🚫
`summary`	complex	Summary statistics of the virtual machines on the host. Includes sub-elements for numbers of **`active`**, **`migrating`** and **`total`** VMs.	🚫
`version major= minor=`	complex	The compatibility level of the host See *Chapter 5, Compatibility Level Versions*.	🔒
`root_password`	string	The root password of this host, by convention only included in the client-provided host representation on creation.	⚠ 🔒

The **`status`** contains one of the following enumerative values: **`down`**, **`error`**, **`initializing`**, **`installing`**, **`install_failed`**, **`maintenance`**, **`non_operational`**, **`non_responsive`**, **`pending_approval`**, **`preparing_for_maintenance`**, **`connecting`**, **`reboot`**, **`unassigned`** and **`up`**. These states are listed in **`host_states`** under **`capabilities`** (See *Section 6.1.21, “Resource Status States”*).

Example 12.1. An XML representation of a host

```
<host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"
  href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3">
    <name>host1</name>
    <actions>
        <link rel="install"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/install"/>
        <link rel="activate"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/activate"/>
        <link rel="fence"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/fence"/>
        <link rel="deactivate"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/deactivate"/>
        <link rel="approve"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/approve"/>
        <link rel="iscsilogin"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/iscsilogin"/>
        <link rel="iscsidiscover"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/iscsidiscover"/>
        <link rel="commitnetconfig"
          href="/api/hosts/762f3276-9d1e-11e0-a27c-525400d75548/commitnetconfig"/>
    </actions>
    <link rel="storage"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/storage"/>
    <link rel="nics"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics"/>
    <link rel="tags"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/tags"/>
    <link rel="permissions"
```

```
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/permissions"/>
    <link rel="statistics"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/statistics"/>
    <type>rhev-h</type>
    <address>host1.example.com</address>
    <status>
        <state>up</state>
    </status>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <port>54321</port>
    <storage_manager>true</storage_manager>
    <power_management>
        <enabled>false</enabled>
        <options/>
    </power_management>
    <ksm>
        <enabled>true</enabled>
    </ksm>
    <transparent_hugepages>
        <enabled>true</enabled>
    </transparent_hugepages>
    <iscsi>
        <initiator>iqn.2001-04.com.example:diskarrays-sn-a8675309</initiator>
    </iscsi>
    <cpu>
        <topology cores="2"/>
        <name>Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz</name>
        <speed>2993</speed>
    </cpu>
    <summary>
        <active>2</active>
        <migrating>0</migrating>
        <total>3</total>
    </summary>
    <version major="3" minor="0"/>
</host>
```

Creation of a new host requires the **`name`**, **`address`** and **`root_password`** elements. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 12.2. Creating a host

```
POST /api/hosts HTTP/1.1
Accept: application/xml
Content-type: application/xml

<host>
  <name>host2</name>
  <address>host2.example.com</address>
  <root_password>p@55w0Rd!</root_password>
</host>
```

New host creation applies only to the addition of Red Hat Enterprise Linux hosts. Red Had Enterprise Virtualization Manager detects hypervisor hosts automatically and requires approval for their use.

The **`root_password`** element is only included in the client-provided initial representation and is not exposed in the representations returned from subsequent requests.

The **name**, **name**, **cluster**, **power_management**, **transparent_hugepages** and **ksm** elements are updatable post-creation. See *Section 7.3.2, " Updating a Resource "* for more information.

Example 12.3. Updating a host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<host>
  <name>host3</name>
</host>
```

Removal of a host requires a **DELETE** request.

Example 12.4. Removing a host

```
DELETE /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3 HTTP/1.1

HTTP/1.1 204 No Content
```

12.1. Power Management

The **power_management** element provides users with the ability to set a power management configuration, which is required for host fencing. Certain sub-elements are required when configuring **power_management**.

Table 12.2. Power management options

Element	Type	Description	Properties
type=	fencing device code	A list of valid fencing device codes are available in the **capabilities** collection. See *Section 6.1.4, "Power Managers"*.	⚠ 🔒
enabled	Boolean: true or false	Indicates whether power management configuration is enabled or disabled.	⚠
address	string	The host name or IP address of the host.	⚠
username	string	A valid user name for power management.	
password	string	A valid, robust password for power management.	
options	complex	Fencing options for the selected **type=**.	

The **options** element requires a list of **option** sub-elements. Each **option** requires a **name** and **type** attributes. Certain options are only available for specific fencing types as defined in the **capabilities** collection (see *Section 6.1.4, "Power Managers"*).

A new host includes an optional **power_management** configuration when **POST**ing to the host resource. The **power_management** configuration is updatable using a **PUT** request.

Example 12.5. An XML representation of a host's power management configuration

```
<host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"
  href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3">
    <name>host1</name>
    ...
    <power_management type="ilo">
        <enabled>true</enabled>
        <address>192.168.1.107</address>
        <username>admin</username>
        <password>p@55w0Rd!</password>
        <options>
            <option name="secure" value="true"/>
            <option name="port" value="54345"/>
            <option name="slot" value="3"/>
        </options>
    </power_management>
    ...
</host>
```

12.2. Memory Management

The API provides two configuration settings for a host's memory management.

Kernel SamePage Merging (KSM) reduces references to memory pages from multiple identical pages to a single page reference. This helps with optimization for memory density. KSM uses the **ksm** element.

Example 12.6. Setting KSM memory management

```
PUT /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3 HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"
  href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3">
    <ksm>true</ksm>
</host>
```

Transparent Hugepage support expands the size of memory pages beyond the standard 4kB limit. This reduces memory consumption and increases host performance. Transparent Hugepage support uses the **transparent_hugepages** element.

Example 12.7. Setting Transparent Hugepage memory management

```
PUT /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3 HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"
  href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3">
    <transparent_hugepages>true</transparent_hugepages>
```

```
</host>
```

Availability of Transparent Hugepage support is found in the **capabilities** collection. See *Section 6.1.2, “Features”*.

12.3. Network Interface Sub-Collection

The **nics** sub-collection represents a host's physical network interfaces. Each **host_nic** element in the representation acts as a network interface and contains the following elements:

> **Note**
>
> The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 12.3. Elements for a host's network interfaces

Element	Type	Description	Properties
name	string	The name of the host network interface, e.g. **eth0**	⚠ [1] 🔒
link rel="statistics"	relationship	A link to the **statistics** sub-collection for a host's network interface statistics.	🚫
link rel="master"	relationship	A reference to the master bonded interface, if this is a slave interface.	🔒
host id=	GUID	A reference to the host.	🔒
network id=	GUID	A reference to the network, if any, that the interface is attached.	⚠ [2]
mac address=	string	The MAC address of the interface.	🔒
ip address= netmask= gateway=	complex	The IP level configuration of the interface.	
boot_protocol	enumerated	The protocol for IP address assignment when the host is booting. A list of enumerated values is available in **capabilities**. See *Section 6.1.18, “Boot Protocols”*.	
speed	integer	The network interface speed in bits per second.	🔒
status	enumerated	The link status for the network interface. These states are listed in **host_nic_states** under	🔒

Element	Type	Description	Properties
		`capabilities` (See *Section 6.1.21, “Resource Status States”*).	
`vlan id`	integer	The VLAN which this interface represents.	🔒
`bonding`	complex	A list of **`options`** and **`slave`** NICs for bonded interfaces.	⚠ [3] 🔒

[1] Only required when adding bonded interfaces. Other interfaces are read-only and cannot be added.

[2] Only required when adding bonded interfaces. Other interfaces are read-only and cannot be added.

[3] Only required when adding bonded interfaces. Other interfaces are read-only and cannot be added.

Example 12.8. An XML representation of a network interface on a host

```
<host_nic id="e8f02fdf-3d7b-4135-86e1-1bf185570cd8"
  href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
  e8f02fdf-3d7b-4135-86e1-1bf185570cd8">
    <name>bond0</name>
    <link rel="statistics"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
      e8f02fdf-3d7b-4135-86e1-1bf185570cd8/statistics"/>
    <host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"/>
    <network id="e657d631-657d-42bb-a536-73501a085d85"
      href="/api/networks/e657d631-657d-42bb-a536-73501a085d85"/>
    <mac address="D6:76:F1:3A:AF:74"/>
    <ip address="192.168.0.128" netmask="255.255.255.0" gateway="192.168.0.1"/>
    <boot_protocol>dhcp</boot_protocol>
    <speed>1000000000</speed>
    <status>
        <state>up</state>
    </status>
    <bonding>
        <options>
            ...
        </options>
        <slaves>
            <host_nic id="eb14e154-5e73-4f7f-bf6b-7f52609d94ec"/>
            <host_nic id="6aede5ca-4c54-4b37-a81b-c0d6b53558ea"/>
        </slaves>
    </bonding>
    <actions>
        <link rel="attach"
          href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
          e8f02fdf-3d7b-4135-86e1-1bf185570cd8/attach"/>
        <link rel="detach"
          href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
          e8f02fdf-3d7b-4135-86e1-1bf185570cd8/detach"/>
    </actions>
</host_nic>
```

An API user creates only bonded interfaces (see *Section 12.3.1, “Bonded Interfaces”*). All other network interfaces contain updatable **`network`**, **`ip`** and **`boot_protocol`** elements using a **`PUT`** request.

When adding a new network interface, the **name** and **network** elements are required. Identify the **network** element with the **id** attribute or **name** element.

An API user modifies a network interface with a **PUT** request.

```
PUT /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
e8f02fdf-3d7b-4135-86e1-1bf185570cd8 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<nic>
    <ip address="192.168.0.129" netmask="255.255.255.0" gateway="192.168.0.1"/>
</nic>
```

An API user removes a network interface with a **DELETE** request.

```
DELETE /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
e8f02fdf-3d7b-4135-86e1-1bf185570cd8 HTTP/1.1

HTTP/1.1 204 No Content
```

Important

The API as documented in this section is experimental and subject to change. It is not covered by the backwards compatibility statement in *Section 6, “Backwards Compatibility Statement”*.

12.3.1. Bonded Interfaces

A bonded interface is represented as a **host_nic** resource containing a **bonding** element.

Table 12.4. Bonded interface properties

Element	Type	Description	Properties
options	complex	A list of **option** elements for a bonded interface. Each **option** contains property **name** and **value** attributes.	⚠ [1] 🔒
slaves	complex	A list of slave **host_nic id=** elements for a bonded interface.	⚠ [2] 🔒

[1] Only required when adding bonded interfaces. Other interfaces are read-only and cannot be added.

[2] Only required when adding bonded interfaces. Other interfaces are read-only and cannot be added.

An API user creates a new bond when **POST**ing to a **host_nic** with bonding options and slave interfaces. The **name**, **network** and **bonded** elements are required when creating a new bonded interface. Either the **id** or **name** elements identify the **network** and slave **host_nic**s.

Example 12.9. Creating a bonded interface

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<host_nic>
    <name>bond4</name>
    <network id="e657d631-657d-42bb-a536-73501a085d85"/>
    <bonding>
        <options>
            ...
        </options>
        <slaves>
            <host_nic id="eb14e154-5e73-4f7f-bf6b-7f52609d94ec"/>
            <host_nic id="6aede5ca-4c54-4b37-a81b-c0d6b53558ea"/>
        </slaves>
    </bonding>
</host_nic>
```

Important

bond0, **bond1**, **bond2**, **bond3** and **bond4** are the only valid names for a bonded interface.

Like other resources, a **DELETE** request to a bonded interface URI deletes it.

Important

Changes to bonded interface configuration must be explicitly committed. See *Section 12.6.8, “ Commit Network Configuration Action ”*.

12.3.2. Network Interface Statistics

Each host's network interface exposes a **statistics** sub-collection for a host's network interface statistics. Each **statistic** contains the following elements:

Table 12.5. Elements for a host's network interface statistics

Element	Type	Description
name	string	The unique identifier for the statistic entry.
description	string	A plain text description of the statistic.
unit	string	The unit or rate to measure the statistical values.
type	One of **GAUGE** or **COUNTER**	The type of statistic measures.
values type=	One of **INTEGER** or **DECIMAL**	The data type for the statistical values that follow.
value	complex	A data set that contains **datum**.
datum	see **values type**	An individual piece of data from a **value**.
host_nic id=	relationship	A relationship to the containing **host_nic** resource.

The following table lists the statistic types for network interfaces on hosts.

Table 12.6. Host NIC statistic types

Name	Description
data.current.rx	The rate in bytes per second of data received.
data.current.tx	The rate in bytes per second of data transmitted.
errors.total.rx	Total errors from receiving data.
errors.total.tx	Total errors from transmitting data.

Example 12.10. An XML representation of a host's network interface statistics sub-collection

```
<statistics>
    <statistic id="ecd0559f-e88f-3330-94b4-1f091b0ffdf7"
      href="/api/hosts/25fcdd2e-d452-11e0-bb4d-525400d75548/nics/
      c34728e8-4338-4540-ac9b-86b8582e602e/statistics/
      ecd0559f-e88f-3330-94b4-1f091b0ffdf7">
        <name>data.current.rx</name>
        <description>Receive data rate</description>
        <values type="DECIMAL">
            <value>
                <datum>0</datum>
            </value>
        </values>
        <type>GAUGE</type>
        <unit>BYTES_PER_SECOND</unit>
        <host_nic id="c34728e8-4338-4540-ac9b-86b8582e602e"
          href="/api/hosts/25fcdd2e-d452-11e0-bb4d-525400d75548/nics/
          c34728e8-4338-4540-ac9b-86b8582e602e"/>
    </statistic>
    ...
</statistics>
```

Note

This **statistics** sub-collection is read-only.

12.3.3. Attach Action

A host network interface is attached to a network, indicating the given network is accessible over the interface. Either the **id** or **name** elements identify the **network**.

Example 12.11. Action to attach a host network interface to a network

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
e8f02fdf-3d7b-4135-86e1-1bf185570cd8/attach HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <network id="e657d631-657d-42bb-a536-73501a085d85"/>
</action>
```

Important

This networking configuration change must be explicitly committed. See *Section 12.6.8, “ Commit Network Configuration Action ”*.

12.3.4. Detach Action

Detach an interface from a network. Either the **`id`** or **`name`** elements identify the **`network`**.

Example 12.12. Action to detach a host network interface to a network

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/nics/
e8f02fdf-3d7b-4135-86e1-1bf185570cd8/detach HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <network id="e657d631-657d-42bb-a536-73501a085d85"/>
</action>
```

Important

This networking configuration change must be explicitly committed. See *Section 12.6.8, “ Commit Network Configuration Action ”*.

12.4. Storage Sub-Collection

The **`storage`** sub-collection provides a list of the iSCSI and FCP storage representations available on the host. This storage is used to create storage domains, as described in *Chapter 11, Storage Domains*.

Each **`storage`** representation in the sub-collection represents a SCSI LUN.

Example 12.13. An XML representation of the storage sub-collection on a host

```
<host_storage>
    <storage id="82fb123b-321e-40a1-9889-95dcd2654463"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/storage/
      82fb123b-321e-40a1-9889-95dcd2654463">
        <name>LUN0</name>
        <type>iscsi</type>
        <logical_unit id="LUN0">
            <address>mysan.example.com</address>
            <target>iqn.2009-08.com.example:mysan.foobar</target>
        </logical_unit>
    </storage>
</host_storage>
```

Note

The **host_storage** collection is read-only.

Important

The API as documented in this section is experimental and subject to change. It is not covered by the backwards compatibility statement in *Section 6, “Backwards Compatibility Statement”*.

12.5. Statistics Sub-Collection

Each host resource exposes a **statistics** sub-collection for host-specific statistics. Each **statistic** contains the following elements:

Table 12.7. Elements for host statistics

Element	Type	Description
name	string	The unique identifier for the statistic entry.
description	string	A plain text description of the statistic.
unit	string	The unit or rate to measure the statistical values.
type	One of **GAUGE** or **COUNTER**	The type of statistic measures.
values type=	One of **INTEGER** or **DECIMAL**	The data type for the statistical values that follow.
value	complex	A data set that contains **datum**.
datum	see **values type**	An individual piece of data from a **value**.
host id=	relationship	A relationship to the containing **host** resource.

The following table lists the statistic types for hosts.

Table 12.8. Host statistic types

Name	Description
memory.total	Total memory in bytes on the host.
memory.used	Memory in bytes used on the host.
memory.free	Memory in bytes free on the host.
memory.buffers	I/O buffers in bytes.
memory.cached	OS caches in bytes.
swap.total	Total swap memory in bytes on the host.
swap.free	Swap memory in bytes free on the host.
swap.used	Swap memory in bytes used on the host.
swap.cached	Swap memory in bytes also cached in host's memory.
ksm.cpu.current	Percentage of CPU usage for Kernel SamePage Merging.

Name	Description
`cpu.current.user`	Percentage of CPU usage for users.
`cpu.current.system`	Percentage of CPU usage for system.
`cpu.current.idle`	Percentage of idle CPU usage.
`cpu.load.avg.5m`	CPU load average per five minutes.

Example 12.14. An XML representation of the host's statistics sub-collection

```
<statistics>
    <statistic id="4ae97794-f56d-3f05-a9e7-8798887cd1ac"
      href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/
      statistics/4ae97794-f56d-3f05-a9e7-8798887cd1ac">
        <name>memory.total</name>
        <description>Total memory</description>
        <unit>BYTES</unit>
        <type>GUAGE</type>
        <values type="INTEGER">
            <value>
                <datum>3983540224<datum>
            </value>
        </values>
        <host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"
          href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"/>
    </statistic>
    ...
</statistics>
```

Note

A host's **`statistics`** sub-collection is read-only.

12.6. Actions

The following sections describe the actions associated with **`host`** resources.

The API contains a number of possible actions for hosts: **`install`**, **`activate`**, **`fence`**, **`deactivate`**, **`approve`**, **`iscsilogin`**, **`iscsidiscover`** and **`commitnetconfig`**.

12.6.1. Install Action

Install VDSM and related software on the host. The host type defines additional parameters for the action.

- **Red Hat Enterprise Linux host** - This host type requires an **`root_password`** element that refers to the password for the host's **`root`** user.

- **Red Hat Enterprise Virtualization Hypervisor host** - This host type requires an **`image`** element that refers to an ISO file stored on the Red Hat Enterprise Virtualization Manager server.

Example 12.15. Action to install VDSM to a Red Hat Enterprise Linux host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/install HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <root_password>p@55w0Rd!</root_password>
</action>
```

Example 12.16. Action to install VDSM to a Red Hat Enterprise Virtualization Hypervisor host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/install HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <image>/usr/share/rhev-hypervisor/rhev-hypervisor.iso</image>
</action>
```

12.6.2. Activate Action

Activate the host for use, such as running virtual machines.

Example 12.17. Action to activate a host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/activate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

12.6.3. Fence Action

An API user controls a host's power management device with the **fence** action. See *Section 12.1, “Power Management”* for details on configuring a fencing device for a host.

The **capabilities** lists available **fence_type** options. See *Section 6.1.5, “Fence Types”* for details on accessing this list.

Example 12.18. Action to fence a host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/fence
Accept: application/xml
Content-Type: application/xml

<action>
    <fence_type>start</fence_type>
</action>
```

12.6.4. Deactivate Action

Deactivate the host to perform maintenance tasks.

Example 12.19. Action to deactivate a host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/deactivate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

12.6.5. Approve Action

Approve a pre-installed Red Hat Enterprise Virtualization Hypervisor host for usage in the virtualization environment. This action also accepts an optional **cluster** element to define the target cluster for this host.

Example 12.20. Action to approve a host

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/approve HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"/>
<action>
```

12.6.6. iSCSI Login Action

The **iscsilogin** action enables a host to login to an iSCSI target. Logging into a target makes the contained LUNs available in the **host_storage** collection. See *Section 12.4, “Storage Sub-Collection”*.

Example 12.21. Action to enable a host to login to iSCSI target

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/iscsilogin HTTP/1.1
Accept: application/xml
Content-Type: application/xml


<action>
    <iscsi>
        <address>mysan.example.com</address>
        <target>iqn.2009-08.com.example:mysan.foobar</target>
        <username>jimmy</username>
        <password>s3kr37</password>
    </iscsi>
</action>
```

12.6.7. iSCSI Discover Action

The **iscsidiscover** action enables an iSCSI portal to be queried for its list of LUNs.

Example 12.22. Action to query a list of LUNs for iSCSI portal

```
POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/iscsidiscover HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<action>
    <iscsi>
        <address>mysan.example.com</address>
    </iscsi>
</action>

HTTP/1.1 202 Accept
Content-Type: application/xml

<action id="e9126d04-0f74-4e1a-9139-13f11fcbb4ab"
  href="/api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/iscsidiscover/
  e9126d04-0f74-4e1a-9139-13f11fcbb4ab">
    <iscsi_target>iqn.2009-08.com.example:mysan.foobar</iscsi_target>
    ...
<action>
```

12.6.8. Commit Network Configuration Action

An API user commits the network configuration to persist a host network interface attachment or detachment, or persist the creation and deletion of a bonded interface.

Example 12.23. Action to commit network configuration

```

POST /api/hosts/2ab5e1da-b726-4274-bbf7-0a42b16a0fc3/commitnetconfig HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

Important

Networking configuration is only committed after the Manager has established that host connectivity is not lost as a result of the configuration changes. If host connectivity is lost, the host requires a reboot and automatically reverts to the previous networking configuration.

Chapter 13.

Virtual Machines

The **vms** collection provides information about virtual machines in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="vms"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a virtual machine resource representation.

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 13.1. Virtual machine elements

Element	Type	Description	Properties
link rel="disks"	relationship	A link to the **disks** sub-collection for virtual machine resources.	
link rel="nics"	relationship	A link to the **nics** sub-collection for virtual machine resources.	
link rel="cdroms"	relationship	A link to the **cdroms** sub-collection for virtual machine resources.	
link rel="snapshots"	relationship	A link to the **snapshots** sub-collection for virtual machine resources.	
link rel="tags"	relationship	A link to the **tags** sub-collection for virtual machine resources.	
link rel="permissions"	relationship	A link to the **permissions** sub-collection for virtual machine permissions. See *Section 7.3.7, “ Permissions ”*.	
link rel="statistics"	relationship	A link to the **statistics** sub-collection for virtual machine resources.	⊘
type	enumerated	The virtual machine type. A list of enumerated values are available in **capabilities**. See *Section 6.1.8, “Virtual Machine Types”*.	🔒
status	See below	The virtual machine status.	⊘
memory	integer	The amount of memory allocated to the guest in bytes.	
cpu	complex	The CPU **topology** i.e. number of **sockets** and **cores** available to the guest.	

Element	Type	Description	Properties
`os type=`	string, e.g. **`RHEL5`** or **`WindowsXP`**	The guest operating system type.	
`os boot dev=`	enumerated	A list of boot devices described by a **`dev`** attribute on a **`boot`** element. A list of enumerated values are available in **`capabilities`**. See *Section 6.1.9, “Boot Devices”*.	
`os kernel`	string	A path to a kernel image the virtual machine is configured to boot. This option supports booting a Linux kernel directly rather than through the BIOS bootloader.	
`os initrd`	string	A path to an initrd image to be used with the previously specified kernel. This option supports booting a Linux kernel directly rather than through the BIOS bootloader.	
`os cmdline`	string	A kernel command line parameter string to be used with the defined kernel. This option supports booting a Linux kernel directly rather than through the BIOS bootloader.	
`high_availability`	complex	Set **`enabled`** to **`true`** if the virtual machine should be automatically restarted if the virtual machine or its host crashes. A **`priority`** element controls the order in which Virtual Machines are re-started.	
`display`	complex	The display **`type`** (either **`vnc`** or **`spice`**), port, and the number of **`monitors`**.	
`cluster id=`	GUID	A reference to the virtual machine's host cluster. See *Chapter 9, Host Clusters*.	⚠ 🔒
`template id=`	GUID	A reference to the template on which this virtual machine is based.	⚠ 🔒
`domain id=`	GUID	A reference to the virtual machine's domain.	🔒
`start_time`	**`xsd:dateTime`** format: **`YYYY-MM-DDThh:mm:ss`**	The date and time at which this virtual machine was started.	🚫
`creation_time`	**`xsd:dateTime`** format: **`YYYY-MM-DDThh:mm:ss`**	The date and time at which this virtual machine was created.	🚫

Element	Type	Description	Properties
origin	One of **rhev**, **vmware** or **xen**	The system from which this virtual machine originated.	🔒
stateless	Boolean: true or false	**true** if the virtual machine is stateless. A stateless virtual machine contains a snapshot of its disk image taken at boot and deleted at shutdown. This means state changes do not persist after a reboot.	
placement_policy	complex	Sets the placement policy for virtual machine migration. Requires a default **host=** and an **affinity** (one of **migratable**, **user_migratable** or **pinned**). Leave the **host** element empty to set no preferred host.	
memory_policy	complex	Sets the memory policy for virtual machines. Defines the minimum amount of **guaranteed** memory on a host in order for the virtual machine to run.	
custom_properties	complex	A set of user-defined environment variable passed as parameters to custom scripts. Each **custom_property** contains **name** and **value** attributes. A list of enumerated values are available in **capabilities**. See *Section 6.1.17, "Custom Properties"*.	
usb	complex	Defines the USB policy for a virtual machine. Requires an **enabled** element set to a Boolean value.	
guest_info	complex	A reference to the guest client information. Includes an **ip** element with an **address=** attribute.	🔒
vmpool	complex	A reference to the virtual machine pool. This element only appears for virtual machines part of a pool.	🔒
timezone	tz database format: **Area/ Location**	The the Sysprep timezone setting for a Windows virtual machine. Only certain timezones are allowed as specified in *Appendix D, Timezones*.	
domain	complex	The the Sysprep domain setting for a Windows virtual machine. Requires a **name** from the **domains** collection. See *Chapter 16, Domains* for more information about domains.	

The **status** contains one of the following enumerative values: **unassigned**, **down**, **up**, **powering_up**, **powered_down**, **paused**, **migrating_from**, **migrating_to**, **unknown**,

not_responding, **wait_for_launch**, **reboot_in_progress**, **saving_state**, **restoring_state**, **suspended**, **image_illegal**, **image_locked** or **powering_down**. These states are listed in **vm_states** under **capabilities** (See *Section 6.1.21, “Resource Status States”*).

Example 13.1. An XML representation of a virtual machine

```
<vm id="082c794b-771f-452f-83c9-b2b5a19c0399"
  href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399">
    <name>vm1</name>
    <description>Virtual Machine 1</description>
    <actions>
        <link rel="start"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/start"/>
        <link rel="stop"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/stop"/>
        <link rel="shutdown"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/shutdown"/>
        <link rel="suspend"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/suspend"/>
        <link rel="detach"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/detach"/>
        <link rel="migrate"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/migrate"/>
        <link rel="export"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/export"/>
        <link rel="import"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/import"/>
        <link rel="move"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/move"/>
        <link rel="ticket"
          href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/ticket"/>
    </actions>
    <link rel="disks"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/disks"/>
    <link rel="nics"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/nics"/>
    <link rel="cdroms"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/cdroms"/>
    <link rel="snapshots"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/snapshots"/>
    <link rel="users"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/users"/>
    <link rel="tags"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/tags"/>
    <link rel="permissions"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/permissions"/>
    <link rel="statistics"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/statistics"/>
    <type>desktop</type>
    <status>
        <state>up</state>
    </status>
    <memory>536870912</memory>
    <cpu>
        <topology cores="1" sockets="1"/>
    </cpu>
    <os type="RHEL5">
        <boot dev="hd"/>
        <kernel/>
        <initrd/>
        <cmdline/>
    </os>
    <highly_available>
```

```
        <enabled>true</enabled>
        <priority>20</priority>
    </highly_available>
    <display>
        <type>vnc</type>
        <port>5910</port>
        <monitors>1</monitors>
    </display>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <template id="00000000-0000-0000-0000-000000000000"
      href="/api/templates/00000000-0000-0000-0000-000000000000"/>
    <start_time>2010-18-16T13:14:15</start_time>
    <creation_time>2010-08-16T14:24:29</creation_time>
    <origin>rhev</origin>
    <stateless>false</stateless>
    <placement_policy>
        <host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"/>
        <affinity>migratable</affinity>
    </placement_policy>
    <memory_policy>
        <guaranteed>536870912</guaranteed>
    </memory_policy>
    <usb>
        <enabled>true</enabled>
    </usb>
    <custom_properties>
        <custom_property value="124" name="sndbuf"/>
    </custom_properties>
    <guest_info>
        <ip address="192.168.0.25"/>
    </guest_info>
</vm>
```

Creation of a new virtual machine requires the **`name`**, **`template`** and **`cluster`** elements. Identify the **`template`** and **`cluster`** elements with the **`id`** attribute or **`name`** element. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 13.2. Creating a virtual machine with 512 MB and boots from the virtual hard disk

```
POST /api/vms HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vm>
    <name>vm2</name>
    <description>Virtual Machine 2</description>
    <type>desktop</type>
    <memory>536870912</memory>
    <cluster>
        <name>default</name>
    </cluster>
    <template>
        <name>Blank</name>
    </template>
    <os>
      <boot dev="hd"/>
    </os>
</vm>
```

The **name**, **description**, **type**, **memory**, **cpu topology**, **os**, **high_availability**, **display**, **timezone**, **domain**, **stateless**, **placement_policy**, **memory_policy**, **usb** and **custom_properties** elements are updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 13.3. Updating a virtual machine to contain 1 GB of memory

```
PUT /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vm>
    <memory>1073741824</memory>
</vm>
```

Removal of a virtual machine requires a **DELETE** request.

Example 13.4. Removing a virtual machine

```
DELETE /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1

HTTP/1.1 204 No Content
```

13.1. Disks Sub-Collection

The **disks** sub-collection represents all virtual hard disk devices on a virtual machine. A **disk** representation contains the following elements:

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 13.2. Elements for virtual machine disks

Element	Type	Description	Properties
link rel="statistics"	relationship	A link to the **statistics** sub-collection for a virtual machine's disk statistics.	⊘
storage_domains	Complex	The storage domains associated with this disk. Each **storage_domain** element contains an **id** attribute with the associated storage domain's GUID.	⚠[1] 🔒
size	integer	Size of the disk in bytes.	⚠ 🔒

Element	Type	Description	Properties
`type`	enumerated	The type (function) of the disk device. A list of enumerated values is available in **`capabilities`**. See *Section 6.1.12, “Disk Types”*.	🔒
`status`	One of **`illegal`**, **`invalid`**, **`locked`** or **`ok`**	The status of the disk device. These states are listed in **`disk_states`** under **`capabilities`** (See *Section 6.1.21, “Resource Status States”*).	🚫
`interface`	enumerated	The type of interface driver used to connect to the disk device. A list of enumerated values is available in **`capabilities`**. See *Section 6.1.15, “Disk Interfaces”*.	
`format`	enumerated	The underlying storage format. A list of enumerated values is available in **`capabilities`**. See *Section 6.1.14, “Disk Formats”*. Copy On Write (COW) allows snapshots, with a small performance overhead. Raw does not allow snapshots, but offers improved performance.	🔒
`sparse`	Boolean: true or false	**`true`** if the physical storage for the disk should not be preallocated.	🔒
`bootable`	Boolean: true or false	**`true`** if this disk is to be marked as bootable.	
`wipe_after_delete`	Boolean: true or false	**`true`** if the underlying physical storage for the disk should be zeroed when the disk is deleted.	
`propagate_errors`	Boolean: true or false	**`true`** if disk errors should not cause virtual machine to be paused and, instead, disk errors should be propagated to the guest OS.	
`vm id=`	GUID	The ID of the containing virtual machine.	🔒

[1] Only required when the first disk is being added to a virtual machine that was not itself created from a template.

Example 13.5. An XML representation of a disk device

```
<disk id="ed7feafe-9aaf-458c-809a-ed789cdbd5b4"
  href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/disks/
  ed7feafe-9aaf-458c-809a-ed789cdbd5b4">
    <link rel="statistics"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/disks/
      ed7feafe-9aaf-458c-809a-ed789cdbd5b4/statistics"/>
    <storage_domains>
        <storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"/>
    </storage_domains>
    <size>10737418240</size>
    <type>system</type>
    <status>
        <state>ok</state>
    </status>
    <interface>virtio</interface>
    <format>raw</format>
    <bootable>true</bootable>
    <vm id="cdc0b102-fbfe-444a-b9cb-57d2af94f401"
      href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401"/>
</disk>
```

When adding a new disk, the **`size`** element is required. Also the API requires the **`storage_domains`** element when the first disk is added to a virtual machine and not itself created from a template.

Example 13.6. Creating a new a disk device on a Virtual Machine

```
POST /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/disks HTTP/1.1
Accept: application/xml
Content-type: application/xml

<disk>
    <storage_domains>
        <storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"/>
    </storage_domains>
    <size>8589934592</size>
    <type>system</type>
    <interface>virtio</interface>
    <format>cow</format>
    <bootable>true</bootable>
</disk>
```

13.1.1. Disk Cloning

Clone a disk from a template with the **`clone`** element. Set the **`clone`** element to **`true`** within the **`disks`** sub-collection when creating a virtual machine. This clones a disk from the base template and attaches it to the virtual machine.

Example 13.7. Cloning a disk from a template

The following example clones a disk from a template during the creation of a virtual machine.

```
POST /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vm>
    <name>cloned_vm</name>
    <template id="64d4aa08-58c6-4de2-abc4-89f19003b886"/>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"/>
    <disks>
        <clone>true</clone>
        <disk id="4825ffda-a997-4e96-ae27-5503f1851d1b">
            <format>COW</format>
        </disk>
        <disk id="42aef10d-3dd5-4704-aa73-56a023c1464c">
            <format>COW</format>
        </disk>
    </disks>
</vm>
```

13.1.2. Disk Statistics

Each virtual machine's disk exposes a **statistics** sub-collection for disk-specific statistics. Each **statistic** contains the following elements:

Table 13.3. Elements for virtual machine disk statistics

Element	Type	Description
name	string	The unique identifier for the statistic entry.
description	string	A plain text description of the statistic.
unit	string	The unit or rate to measure the statistical values.
type	One of **GAUGE** or **COUNTER**	The type of statistic measures.
values type=	One of **INTEGER** or **DECIMAL**	The data type for the statistical values that follow.
value	complex	A data set that contains **datum**.
datum	see **values type**	An individual piece of data from a **value**.
disk id=	relationship	A relationship to the containing **disk** resource.

The following table lists the statistic types for virtual machine disks.

Table 13.4. Virtual machine disk statistic types

Name	Description
data.current.read	The data transfer rate in bytes per second when reading from the disk.
data.current.write	The data transfer rate in bytes per second when writing to the disk.

Example 13.8. An XML representation of a virtual machine's statistics sub-collection

```
<statistics>
    <statistic id="33b9212b-f9cb-3fd0-b364-248fb61e1272"
      href="/api/vms/3a42530e-3bc5-4094-829d-489257894c2a/disks/
      f28ec14c-fc85-43e1-818d-96b49d50e27b/statistics/
      33b9212b-f9cb-3fd0-b364-248fb61e1272">
        <name>data.current.read</name>
        <description>Read data rate</description>
        <values type="DECIMAL">
            <value>
                <datum>0</datum>
            </value>
        </values>
        <type>GAUGE</type>
        <unit>BYTES_PER_SECOND</unit>
        <disk id="f28ec14c-fc85-43e1-818d-96b49d50e27b"
          href="/api/vms/3a42530e-3bc5-4094-829d-489257894c2a/
          disks/f28ec14c-fc85-43e1-818d-96b49d50e27b"/>
    </statistic>
    ...
</statistics>
```

Note

This **statistics** sub-collection is read-only.

13.2. Network Interfaces Sub-Collection

The **nics** sub-collection represents all network interface devices on a virtual machine. A **nic** representation contains the following elements:

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 13.5. Elements for virtual machine network interfaces

Element	Type	Description	Properties
link rel="statistics"	relationship	A link to the **statistics** sub-collection for a virtual machine's network interface statistics.	⊘
network id=	GUID	A reference to the network which the interface should be connected.	⚠
interface	enumerated	The type of driver used for the nic. A list of enumerated values is available in **capabilities**. See *Section 6.1.11, “NIC Interface Types”*.	
mac address=	string	The MAC address of the interface.	🔒

Example 13.9. An XML representation of a network interface

```
<nic id="7a3cff5e-3cc4-47c2-8388-9adf16341f5e"
  ref="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/nics/
  7a3cff5e-3cc4-47c2-8388-9adf16341f5e">
    <link rel="statistics"
      href="/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/nics/
      7a3cff5e-3cc4-47c2-8388-9adf16341f5e/statistics"/>
    <name>nic1</name>
    <interface>virtio</interface>
    <mac address="00:1a:4a:16:84:07"/>
    <network id="00000000-0000-0000-0000-000000000009"
      href="/api/networks/00000000-0000-0000-0000-000000000009"/>
    <vm id="cdc0b102-fbfe-444a-b9cb-57d2af94f401"
      href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401"/>
</nic>
```

When adding a new network interface, the **name** and **network** elements are required. Identify the **network** element with the **id** attribute or **name** element.

An API user modifies a network interface with a **PUT** request.

```
PUT /api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/nics/
7a3cff5e-3cc4-47c2-8388-9adf16341f5e HTTP/1.1
Accept: application/xml
Content-type: application/xml

<nic>
    <name>nic2</name>
    <network id="00000000-0000-0000-0000-000000000010"/>
    <type>e1000</type>
</nic>
```

An API user removes a network interface with a **DELETE** request.

```
DELETE /api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/nics/
7a3cff5e-3cc4-47c2-8388-9adf16341f5e HTTP/1.1

HTTP/1.1 204 No Content
```

13.2.1. Network Interface Statistics

Each virtual machine's network interface exposes a **statistics** sub-collection for network interface statistics. Each **statistic** contains the following elements:

Table 13.6. Elements for a virtual machine's network interface statistics

Element	Type	Description
name	string	The unique identifier for the statistic entry.
description	string	A plain text description of the statistic.
unit	string	The unit or rate to measure the statistical values.
type	One of **GAUGE** or **COUNTER**	The type of statistic measures.
values type=	One of **INTEGER** or **DECIMAL**	The data type for the statistical values that follow.
value	complex	A data set that contains **datum**.
datum	see **values type**	An individual piece of data from a **value**.
nic id=	relationship	A relationship to the containing **nic** resource.

The following table lists the statistic types for network interfaces on virtual machines.

Table 13.7. Virtual machine NIC statistic types

Name	Description
data.current.rx	The rate in bytes per second of data received.
data.current.tx	The rate in bytes per second of data transmitted.
errors.total.rx	Total errors from receiving data.
errors.total.tx	Total errors from transmitting data.

Example 13.10. An XML representation of a virtual machine's NIC statistics sub-collection

```
<statistics>
    <statistic id="ecd0559f-e88f-3330-94b4-1f091b0ffdf7"
      href="/api/vms/3a42530e-3bc5-4094-829d-489257894c2a/nics/
      6cd08e76-57c0-41ba-a728-7eba46ae1e36/statistics/
      ecd0559f-e88f-3330-94b4-1f091b0ffdf7">
        <name>data.current.rx</name>
        <description>Receive data rate</description>
        <values type="DECIMAL">
            <value>
                <datum>0</datum>
            </value>
        </values>
        <type>GAUGE</type>
        <unit>BYTES_PER_SECOND</unit>
        <nic id="6cd08e76-57c0-41ba-a728-7eba46ae1e36"
          href="/api/vms/3a42530e-3bc5-4094-829d-489257894c2a/
          nics/6cd08e76-57c0-41ba-a728-7eba46ae1e36"/>
    </statistic>
    ...
</statistics>
```

Note

This **`statistics`** sub-collection is read-only.

13.3. CD-ROMs Sub-Collection

The **`cdroms`** sub-collection represents the CD-ROM device on a virtual machine. A **`cdrom`** representation contains the following elements:

Table 13.8. Elements for virtual machine CD-ROMs

Element	Type	Description	Properties
`file id=`	string/filename	A reference to an ISO image. See *Section 11.3, “ Files Sub-Collection ”*.	

Example 13.11. An XML representation of a CD-ROM device

```
<cdrom id="00000000-0000-0000-0000-000000000000"
  href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/cdroms/
  00000000-0000-0000-0000-000000000000">
    <file id="rhel-server-6.0-x86_64-dvd.iso"/>
    <vm id="cdc0b102-fbfe-444a-b9cb-57d2af94f401"
      href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401"/>
</cdrom>
```

When adding a new CD-ROM, the **`file id`** element is required.

The API changes the CD-ROM while the virtual machine is powered-off using a **`PUT`** request:

Example 13.12. Changing a CD-ROM file

```
PUT /api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/
cdroms/00000000-0000-0000-0000-000000000000 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<cdrom>
    <file id="fedora-15-x86_64-dvd.iso"/>
</cdrom>
```

The API changes the CD-ROM during a current session using a **PUT** request with an additional **current** URI argument:

Example 13.13. Changing a CD-ROM file during a current session

```
PUT /api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/
cdroms/00000000-0000-0000-0000-000000000000?current HTTP/1.1
Accept: application/xml
Content-type: application/xml

<cdrom>
    <file id="fedora-15-x86_64-dvd.iso"/>
</cdrom>
```

Note

A virtual machine only contains a single CD-ROM device.

13.4. Snapshots Sub-Collection

A virtual machine saves and restores disk state as a number of snapshots. These are represented and managed through a **rel="snapshot"** sub-collection that behaves similar to other collections, as described in *Chapter 7, Common Features*.

Virtual machine snapshots are represented with **snapshot** elements that contain the following elements:

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 13.9. Elements for virtual machine snapshots

Element	Type	Description	Properties
`vm id=`	GUID	The ID and URI of the virtual machine to which this snapshot pertains.	🔒
`date`	**`xsd:dateTime`** format: **`YYYY-MM-DDThh:mm:ss`**	The date and time at which this snapshot was created.	⊘
`link rel="prev"`	relationship	A link to the previous snapshot of this virtual machine.	⊘

When adding a new snapshot, only the **`description`** element is specified.

Note

Note that it is not possible to modify snapshot elements using **`PUT`**.

Example 13.14. An XML representation of a virtual machine snapshot

```
<snapshot id="f5288fd5-5178-4b7d-b87c-c01a40e40168"
  href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/snapshots/
  f5288fd5-5178-4b7d-b87c-c01a40e40168">
    <description>Virtual Machine 1 - Snapshot A</description>
    <actions>
        <link rel="restore"
        href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/snapshots/
        f5288fd5-5178-4b7d-b87c-c01a40e40168/restore"/>
    </actions>
    <link rel="prev"
      href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/snapshots/
      ce411b3e-e4e0-4482-8b2f-d1ed998b9130"/>
    <vm id="5114bb3e-a4e6-44b2-b783-b3eea7d84720"
      href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720"/>
    <date>2010-08-16T14:24:29</date>
</snapshot>
```

An API user restores to a virtual machine snapshot using the **`rel="restore"`** action link in the snapshot representation.

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/snapshots/f5288fd5-5178-4b7d-b87c-
c01a40e40168/restore HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

13.5. Statistics Sub-Collection

Each virtual machine resource exposes a **`statistics`** sub-collection for virtual machine-specific statistics. Each **`statistic`** contains the following elements:

Table 13.10. Elements for virtual machine statistics

Element	Type	Description
`name`	string	The unique identifier for the statistic entry.
`description`	string	A plain text description of the statistic.
`unit`	string	The unit or rate to measure the statistical values.
`type`	One of **`GAUGE`** or **`COUNTER`**	The type of statistic measures.
`values type=`	One of **`INTEGER`** or **`DECIMAL`**	The data type for the statistical values that follow.
`value`	complex	A data set that contains **`datum`**.
`datum`	see **`values type`**	An individual piece of data from a **`value`**.
`vm id=`	relationship	A relationship to the containing **`vm`** resource.

The following table lists the statistic types for virtual machines.

Table 13.11. Virtual machine statistic types

Name	Description
`memory.installed`	Total memory in bytes allocated for the virtual machine's use.
`memory.used`	Current memory in bytes used by the virtual machine.
`cpu.current.guest`	Percentage of CPU used by the guest.
`cpu.current.hypervisor`	Percentage of CPU overhead on the hypervisor.
`cpu.current.total`	Total percentage of the current CPU in use.

Example 13.15. An XML representation of a virtual machine's statistics sub-collection

```
<statistics>
    <statistic id="ef802239-b74a-329f-9955-be8fea6b50a4"
      href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401/
      statistics/ef802239-b74a-329f-9955-be8fea6b50a4">
        <name>memory.installed</name>
        <description>Total memory configured</description>
        <unit>BYTES</unit>
        <type>GUAGE</type>
        <values type="DECIMAL">
            <value>
                <datum>1073741824<datum>
            </value>
        </values>
        <vm id="cdc0b102-fbfe-444a-b9cb-57d2af94f401"
          href="/api/vms/cdc0b102-fbfe-444a-b9cb-57d2af94f401"/>
    </statistic>
    ...
</statistics>
```

Note

A virtual machine's **`statistics`** sub-collection is read-only.

13.6. Actions

The following sections describe the actions associated with **`vm`** resources.

The API contains a number of possible actions for virtual machines: **`start`**, **`stop`**, **`shutdown`**, **`suspend`**, **`detach`**, **`migrate`**, **`export`**, **`move`** and **`ticket`**.

Information on the action for importing virtual machines is found in *Section 11.2, “ Export Storage Domains ”*.

13.6.1. Start Action

The start action launches a virtual machine.

Example 13.16. Action to start a virtual machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/start HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

The start action allows a **`vm`** element to be provided as a parameter. If a **`vm`** element is provided, the virtual machine uses the values from the provided element and overrides system settings at start time. These settings persist until a user stops the virtual machine.

Example 13.17. Action to start a virtual machine with overridden parameters

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/start HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <pause>true</pause>
    <vm>
        <stateless>true</stateless>
        <display>
            <type>spice</type>
        </display>
        <os>
            <boot dev="cdrom"/>
        </os>
        <cdroms>
            <cdrom>
                <file id="windows-xp.iso"/>
            </cdrom>
        </cdroms>
        <domain>
            <name>domain.example.com</name>
            <user>
                <user_name>domain_user</user_name>
                <password>domain_password</password>
            </user>
        </domain>
        <placement_policy>
            <host id="02447ac6-bcba-448d-ba2b-f0f453544ed2"/>
        </placement_policy>
    </vm>
```

```
</action>
```

Note

Only virtual machines running Windows operating systems use the **`domain`** element when overriding parameters on boot with the **`start`** action. The **`domain`** element determines the domain that the Windows virtual machine joins. If the domain does not exist in the **`domains`** collection (see *Chapter 16, Domains*), this element requires additional **`user`** authentication details, including a **`user_name`** and **`password`**. If the domain exists in the **`domains`** collection, the action requires no additional **`user`** authentication details.

13.6.2. Stop Action

The stop action forces a virtual machine to power-off.

Example 13.18. Action to stop a virtual machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/stop HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

13.6.3. Shutdown Action

The shutdown action sends a shutdown request to a virtual machine.

Example 13.19. Action to send a shutdown request to a virtual machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/shutdown HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

13.6.4. Suspend Action

The suspend action saves the virtual machine state to disk and stops it. The virtual machine state is restored with the start action.

Example 13.20. Action to save virtual machine state and suspend the machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/suspend HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

13.6.5. Detach Action

The detach action detaches a virtual machine from a pool.

Example 13.21. Action to detach a virtual machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/detach HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action/>
```

13.6.6. Migrate Action

The migrate action migrates a virtual machine to another physical host. The destination **host** element is an optional element as Red Hat Enterprise Virtualization Manager automatically selects a default host for migration. If an API user requires a specific **host**, the user can specify the host with either an **id** or **name** parameter.

Example 13.22. Action to migrate a virtual machine to another host

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/migrate HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <host id="2ab5e1da-b726-4274-bbf7-0a42b16a0fc3"/>
</action>
```

13.6.7. Export Action

The export action exports a virtual machine to an **export** storage domain. A destination storage domain must be specified with a **storage_domain** reference. By default, the export action overwrites any existing virtual machine of the same name in the destination domain. An API user changes this behaviour by setting the **overwrite** parameter to **true**. Finally, if snapshots of the virtual machine are not included with the exported virtual machine, the **discard_snapshots** parameter is set to **true**.

Example 13.23. Action to export a virtual machine to an export storage domain

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/export HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <storage_domain>
        <name>export1</name>
    </storage_domain>
    <overwrite>true<overwrite/>
    <discard_snapshots>true<discard_snapshots/>
</action>
```

13.6.8. Move Action

The move action moves virtual machine disks to a different storage domain. The destination storage domain is specified via a **`storage_domain`** reference to either a **`name`** or an **`id`**.

Example 13.24. Action to move virtual machine disks to a different storage domain

```
POST /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399/move HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <storage_domain>
        <name>images2</name>
    </storage_domain>
</action>
```

13.6.9. Ticket Action

The ticket action generates a time-sensitive authentication token for accessing a virtual machine's display. The client-provided **`action`** optionally includes a **`ticket`** representation containing a **`value`** (if the token string needs to take on a particular form) and/or an **`expiry`** time in minutes. In any case, the response specifies the actual ticket value and expiry used.

Example 13.25. Action to generate authentication token for a virtual machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/ticket HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <ticket>
        <expiry>120</expiry>
    </ticket>
</action>

200 OK
Content-Type: application/xml

<action id="94e07552-14ba-4c27-8ce6-2cc75190d3ef"
  href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/ticket/
  94e07552-14ba-4c27-8ce6-2cc75190d3ef">
    <status>
        <state>complete</state>
    </status>
    <ticket>
        <value>5c7CSzK8Sw41</value>
        <expiry>120</expiry>
    </ticket>
    <link rel="parent"
      href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720"/>
    <link rel="replay"
      href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/ticket"/>
</action>
```

13.6.10. Force Remove Action

An API user forces the removal of a faulty virtual machine with the **`force`** action. This action requires a **`DELETE`** method. The request body contains an **`action`** representation with the **`force`** parameter set to **`true`**. The request also requires an additional **`Content-type: application/xml`** header to process the XML representation in the body.

Example 13.26. Force remove action on a virtual machine

```
DELETE /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <force>true</force>
</action>
```

Chapter 14.

Templates

The **templates** collection provides information about the virtual machine templates in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="templates"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a virtual machine template resource representation.

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 14.1. Virtual machine template elements

Element	Type	Description	Properties
link rel="disks"	relationship	A link to the **disks** sub-collection for virtual machine template resources.	⊘
link rel="nics"	relationship	A link to the **nics** sub-collection for virtual machine template resources.	
link rel="cdroms"	relationship	A link to the **cdroms** sub-collection for virtual machine template resources.	⊘
link rel="permissions"	relationship	A link to the **permissions** sub-collection for virtual machine template permissions. See *Section 7.3.7, “ Permissions ”*.	
type	enumerated	The type of virtual machine the template provides. A list of enumerated values are available in **capabilities**. See *Section 6.1.8, “Virtual Machine Types”*.	
status	One of **illegal**, **locked** or **ok**	The template status. These states are listed in **template_states** under **capabilities** (See *Section 6.1.21, “Resource Status States”*).	⊘
memory	integer	The amount of memory allocated to the guest, in bytes.	
cpu	complex	The CPU **topology** (i.e. number of **sockets** and **cores**) available to the guest.	
os type=	string, e.g. **RHEL5** or **WindowsXP**	The guest operating system type.	
os boot dev=	enumerated	A list of boot devices, described by a **dev** attribute on a **boot** element.	

Element	Type	Description	Properties
		A list of enumerated values are available in **capabilities**. See *Section 6.1.9, “Boot Devices”*.	
os kernel	string	A path to a kernel image which the template is configured to boot from.	
os initrd	string	A path to an initrd image to be used with the kernel above.	
os cmdline	string	A kernel command line parameter string to be used with the kernel above.	
cluster id=	GUID	A reference to the template's host cluster. See *Chapter 9, Host Clusters*.	🔒
vm id=	GUID	A reference to the VM on which this template is based. See *Chapter 13, Virtual Machines*.	⚠ 🔒
domain id=	GUID	A reference to the template's domain.	🔒
creation_time	**xsd:dateTime** format: **YYYY-MM-DDThh:mm:ss**	The date and time at which this template was created.	🚫
origin	One of **rhev**, **vmware** or **xen**	The system from which this template originated.	🔒
high_availability	complex	Set **enabled** to **true** if the VM should be automatically restarted if the host crashes. A **priority** element controls the order in which VMs are re-started.	
display	complex	The display **type** (either **vnc** or **spice**), port, and the number of **monitors**.	
stateless	Boolean: true or false	A stateless template contains a snapshot of its disk image taken at boot and deleted at shutdown. This means state changes do not persist after a reboot.	
usb	complex	Defines the USB policy for a virtual machine template. Requires an **enabled** element set to a Boolean value.	
timezone	tz database format: **Area/Location**	The the Sysprep timezone setting for a Windows virtual machine template. Only certain timezones are allowed as specified in *Appendix D, Timezones*.	

Element	Type	Description	Properties
domain	complex	The the Sysprep domain setting for a Windows virtual machine template. Requires a **name** from the **domains** collection. See *Chapter 16, Domains* for more information about domains.	

Example 14.1. An XML representation of a virtual machine template

```
<template id="00000000-0000-0000-0000-000000000000"
  href="/api/templates/00000000-0000-0000-0000-000000000000">
    <name>Blank</name>
    <description>Blank template</description>
    <actions>
        <link rel="export"
          href="/api/templates/00000000-0000-0000-0000-000000000000/export"/>
    </actions>
    <link rel="disks"
      href="/api/templates/00000000-0000-0000-0000-000000000000/disks"/>
    <link rel="nics"
      href="/api/templates/00000000-0000-0000-0000-000000000000/nics"/>
    <link rel="cdroms"
      href="/api/templates/00000000-0000-0000-0000-000000000000/cdroms"/>
    <link rel="permissions"
      href="/api/templates/00000000-0000-0000-0000-000000000000/permissions"/>
    <type>server</type>
    <status>
        <state>ok</state>
    </status>
    <memory>536870912</memory>
    <cpu>
        <topology cores="1" sockets="1"/>
    </cpu>
    <os>
        <boot dev="hd"/>
        <kernel/>
        <initrd/>
        <cmdline/>
    </os>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <creation_time>2010-08-16T14:24:29</creation_time>
    <origin>rhev</origin>
    <highly_available>
        <enabled>true</enabled>
        <priority>100</priority>
    </highly_available>
    <display>
        <type>vnc</type>
        <port>5910</port>
        <monitors>1</monitors>
    </display>
    <stateless>false</stateless>
    <usb>
        <enabled>true</enabled>
    </usb>
</template>
```

Creation of a new template requires the **name** and **vm** elements. Identify the **vm** with the **id** attribute or **name** element. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 14.2. Creating a template from a virtual machine

```
POST /api/templates HTTP/1.1
Accept: application/xml
Content-type: application/xml

<template>
    <name>template1</name>
    <vm id="082c794b-771f-452f-83c9-b2b5a19c0399"/>
</template>
```

The **`name`**, **`description`**, **`type`**, **`memory`**, **`cpu topology`**, **`os`**, **`high_availability`**, **`display`**, **`stateless`**, **`usb`** and **`timezone`** elements are updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 14.3. Updating a virtual machine template to contain 1 GB of memory

```
PUT /api/templates/284t367e-332y-935u-26u9-u9n3l56e8429 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<template>
    <memory>1073741824</memory>
</template>
```

Removal of a virtual machine template requires a **`DELETE`** request.

Example 14.4. Removing a virtual machine template

```
DELETE /api/templates/284t367e-332y-935u-26u9-u9n3l56e8429 HTTP/1.1

HTTP/1.1 204 No Content
```

Templates contain a number of device sub-collections that follow the same XML representation structure as devices in the **`vms`** collection:

- **`disks`** - This is a read-only collection. See *Section 13.1, “Disks Sub-Collection”* for more information.
- **`nics`** - The **`nics`** sub-collection accepts requests for creation, modification and removal of network interfaces in a virtual machine template. See *Section 13.2, “Network Interfaces Sub-Collection”* for more information.
- **`cdroms`** - This is a read-only collection. See *Section 13.3, “CD-ROMs Sub-Collection”* for more information.

14.1. Export Action

The **`templates`** collection contains an **`export`** action. Information on the action for importing templates is found in *Section 11.2, “ Export Storage Domains ”*.

The export action exports a template to an **`Export`** storage domain. A destination storage domain is specified with a **`storage_domain`** reference. By default, the export action overwrites any existing template with the same name in the destination domain. The **`exclusive`** parameter set to **`true`** avoids this.

Example 14.5. Action to export a template to an export storage domain

```
POST /api/templates/00000000-0000-0000-0000-000000000000/export HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
    <storage_domain id="fabe0451-701f-4235-8f7e-e20e458819ed"/>
    <exclusive>true<exclusive/>
</action>
```

Chapter 15.

Virtual Machine Pools

The **`vmpools`** collection provides information about the virtual machine pools in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **`rel="vmpools"`** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a virtual machine pool resource representation.

> **Note**
>
> The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 15.1. Virtual machine pool elements

Element	Type	Description	Properties
`size`	integer	The number of virtual machines in the pool.	
`cluster id=`	GUID	A reference to the cluster resource which virtual machines in this pool run.	⚠ 🔒
`template id=`	GUID	A reference to the template resource which virtual machines in this pool are based.	⚠ 🔒

Example 15.1. An XML representation of a virtual machine pool

```
<vmpool id="3t6y18o-44u3-e7h7-56j7-3k5d8g9w0t"
  href="/api/vmpools/3t6y18o-44u3-e7h7-56j7-3k5d8g9w0t">
    <name>VMPool1</name>
    <description>Virtual Machine Pool 1</description>
    <size>2</size>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <template id="00000000-0000-0000-0000-000000000000"
      href="/api/templates/00000000-0000-0000-0000-000000000000"/>
</vmpool>
```

A new pool requires the **`name`**, **`cluster`** and **`template`** elements. Identify the **`cluster`** and **`template`** with the **`id`** attribute or **`name`** element. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 15.2. Creating a virtual machine pool

```
POST /api/vmpools HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vmpool>
    <name>VM Pool A</name>
    <cluster id="99408929-82cf-4dc7-a532-9d998063fa95"
      href="/api/clusters/99408929-82cf-4dc7-a532-9d998063fa95"/>
    <template id="00000000-0000-0000-0000-000000000000"
      href="/api/templates/00000000-0000-0000-0000-000000000000"/>
</vmpool>
```

The **`name`**, **`description`** and **`size`** are updatable post-creation. See *Section 7.3.2, " Updating a Resource "* for more information.

Example 15.3. Updating a virtual machine pool

```
PUT /api/vmpools/3t6y18o-44u3-e7h7-56j7-3k5d8g9w0t HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vmpool>
    <name>VM Pool B</name>
    <description>Virtual Machine Pool B</description>
    <size>3</size>
</vmpool>
```

Removal of a virtual machine pool requires a **`DELETE`** request.

Example 15.4. Removing a virtual machine

```
DELETE /api/vmpools/3t6y18o-44u3-e7h7-56j7-3k5d8g9w0t HTTP/1.1

HTTP/1.1 204 No Content
```

Important

The API as documented in this chapter is experimental and subject to change. It is not covered by the backwards compatibility statement in *Section 6, "Backwards Compatibility Statement"*.

Chapter 16.

Domains

The API provides the ability to access user and group information from the organization's directory service using the **domains** collection. Domain information is referenced with the **rel="domains"** link.

Table 16.1. Domain elements

Element	Type	Description
name	string	The domain name.
link rel="users"	relationship	A link to the sub-collection for users associated with this domain.
link rel="groups"	relationship	A link to the sub-collection for groups associated with this domain.

The links to **users** and **groups** sub-collections also accept search queries. See *Section 7.2.3, " Searching Collections with Queries "* for more information.

Example 16.1. An XML representation of a domain resource

```
<domain id="77696e32-6b38-7268-6576-2e656e676c61"
  href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61">
    <name>domain.example.com</name>
    <link rel="users"
      href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61/users"/>
    <link rel="groups"
      href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61/groups"/>
    <link rel="users/search"
      href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61/
      users?search={query}"/>
    <link rel="groups/search"
      href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61/
      groups?search={query}"/>
</domain>
```

Note

The **domains** collection and its sub-collections are read-only.

16.1. Domain Users Sub-Collection

The **users** sub-collection contains all users in the directory service. This information is used to add new users to the Red Hat Enterprise Virtualization environment as per *Chapter 19, Users*.

Table 16.2. Domain user elements

Element	Type	Description
name	string	The name of the user.
user_name	string	The username from directory service.
domain id	GUID	The containing directory service domain.
groups	complex	A list of directory service groups for this user.

Example 16.2. An XML representation of a user in the users sub-collection

```
<user id="225f15cd-e891-434d-8262-a66808fcb9b1"
  href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61/users/
  d3b4e7be-5f57-4dac-b937-21e1771a501f">
    <name>RHEV-M Admin</name>
    <user_name>rhevmadmin@domain.example.com</user_name>
    <domain id="77696e32-6b38-7268-6576-2e656e676c61"
      href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61"/>
    <groups>
        <group>
            <name>domain.example.com/Users/Enterprise Admins</name>
        </group>
        <group>
            <name>domain.example.com/Users/Domain Admins</name>
        </group>
        ...
    </groups>
</user>
```

16.2. Domain Groups Sub-Collection

The **groups** sub-collection contains all groups in the directory service. A domain **group** resource contains a set of elements.

Table 16.3. Domain group elements

Element	Type	Description
name	string	The name of the group.
domain id	GUID	The containing directory service domain.

Example 16.3. An XML representation of a group in the groups sub-collection

```
<group id="85bf8d97-273c-4a5c-b801-b17d58330dab"
  href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61/groups/
  85bf8d97-273c-4a5c-b801-b17d58330dab">
    <name>example.com/Users/Enterprise Admins</name>
    <domain id="77696e32-6b38-7268-6576-2e656e676c61"
      href="/api/domains/77696e32-6b38-7268-6576-2e656e676c61"/>
</group>
```

Chapter 17.

Groups

The **`groups`** collection contains imported groups from directory services. A **`group`** resource contains a set of elements.

Table 17.1. Imported group elements

Element	Type	Description
`link rel="tags"`	relationship	A link to the **`tags`** sub-collection for tags attached to this group.
`link rel="permissions"`	relationship	A link to the **`permissions`** sub-collection for permissions attached to this group.
`link rel="roles"`	relationship	A link to the **`roles`** sub-collection for roles attached to this group.

Example 17.1. An XML representation of a group resource

```
<group id="85bf8d97-273c-4a5c-b801-b17d58330dab"
  href="/api/groups/85bf8d97-273c-4a5c-b801-b17d58330dab">
    <name>Everyone</name>
    <link rel="tags"
      href="/api/groups/85bf8d97-273c-4a5c-b801-b17d58330dab/tags"/>
    <link rel="permissions"
      href="/api/groups/85bf8d97-273c-4a5c-b801-b17d58330dab/permissions"/>
    <link rel="roles"
      href="/api/groups/85bf8d97-273c-4a5c-b801-b17d58330dab/roles"/>
</group>
```

Roles

The **`rel="roles"`** link obtained from the entry point URI (see *Chapter 4, Entry Point*) provides access to a static set of system roles. Each individual **`role`** element contains the following:

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 18.1. Role elements

Element	Type	Description	Properties
`link="permits"`	relationship	A link to the **`permits`** sub-collection for role permits.	⚠
`mutable`	Boolean: true or false	Defines the ability to update or delete the role. Roles with **`mutable`** set to **`false`** are roles built into the Red Hat Enterprise Virtualization environment.	⊘
`administrative`	Boolean: true or false	Defines the role as administrative-only.	

Example 18.1. An XML representation of the roles collection

```
<roles>
    <role id="00000000-0000-0000-0000-000000000001"
      href="/api/roles/00000000-0000-0000-0000-000000000001">
        <name>SuperUser</name>
        <description>Roles management administrator</description>
        <link rel="permits"
          href="/api/roles/00000000-0000-0000-0000-000000000001/permits"/>
        <mutable>false</mutable>
        <administrative>true</administrative>
    </role>
    <role id="00000000-0000-0000-0001-000000000001"
      href="/api/roles/00000000-0000-0000-0001-000000000001">
        <name>RHEVMUser</name>
        <description>RHEVM user</description>
        <link rel="permits"
          href="/api/roles/00000000-0000-0000-0001-000000000001/permits"/>
        <mutable>false</mutable>
        <administrative>false</administrative>
    </role>
    <role id="00000000-0000-0000-0001-000000000002"
       href="/api/roles/00000000-0000-0000-0001-000000000002">
        <name>RHEVMPowerUser</name>
        <description>RHEVM power user</description>
        <link rel="permits"
          href="/api/roles/00000000-0000-0000-0001-000000000002/permits"/>
        <mutable>false</mutable>
        <administrative>false</administrative>
    </role>
</roles>
```

Creation of a role requires values for **name**, **administrative** and a list of initial **permits**. See *Section 7.2.4, “ Creating a Resource in a Collection ”* for more information.

Example 18.2. Creating a role

```
POST /api/roles HTTP/1.1
Accept: application/xml
Content-type: application/xml

<role>
    <name>Finance Role</name>
    <administrative>true</administrative>
    <permits>
        <permit id="1"/>
    </permits>
</role>
```

The **name**, **description** and **administrative** elements are updatable post-creation. See *Section 7.3.2, “ Updating a Resource ”* for more information.

Example 18.3. Updating a role

```
PUT /api/roles/8de42ad7-f307-408b-80e8-9d28b85adfd7 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<role>
    <name>Engineering Role</name>
    <description>Standard users in the Engineering Role</description>
    <administrative>false</administrative>
</role>
```

Removal of a role requires a **DELETE** request.

Example 18.4. Removing a role

```
DELETE /api/roles/8de42ad7-f307-408b-80e8-9d28b85adfd7 HTTP/1.1

HTTP/1.1 204 No Content
```

18.1. Permits Sub-Collection

Each role contains a set of allowable actions, or **permits**, which the API lists in **capabilities**. For more information on access to **permits**, see *Section 6.2, “Permits”*.

A role's **permits** are listed as a sub-collection:

Example 18.5. Listing a role's permits

```
GET /api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9/permits HTTP/1.1
Accept: application/xml
```

```
HTTP/1.1 200 OK
Content-Type: application/xml

<permits>
    <permit id="1"
      href="/api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9/permits/1">
        <name>create_vm</name>
        <administrative>false</administrative>
        <role id="b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9"
          href="/api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9"/>
    </permit>
    ...
</permits>
```

Assign a **permit** to a role with a **POST** request to the **permits** sub-collection. Use either an **id** attribute or a **name** element to specify the **permit** to assign.

Example 18.6. Assign a permit to a role

```
POST /api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9/permits HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<permit id="1"/>

HTTP/1.1 201 Created
Content-Type: application/xml

<permits>
    <permit id="1"
      href="/api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9/permits/1">
        <name>create_vm</name>
        <administrative>false</administrative>
        <role id="b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9"
          href="/api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9"/>
    </permit>
</permits>
```

Remove a **permit** from a role with a **DELETE** request to the **permit** resource.

Example 18.7. Remove a permit from a role

```
DELETE /api/roles/b67dfbe2-0dbc-41e4-86d3-a2fbef02cfa9/permits/1 HTTP/1.1

HTTP/1.1 204 No Content
```

Chapter 19.

Users

Users are exposed in a top-level collection and are referenced with the **`rel="users"`** link. Individual **`user`** elements contain the following:

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 19.1. User elements

Element	Type	Description	Properties
`user_name`	string	The user principal name (UPN). The UPN is used as a more convenient identifier when adding a new user.	🔒
`link rel="tags"`	relationship	A link to the **`tags`** sub-collection for user resources.	
`link rel="roles"`	relationship	A link to the **`roles`** sub-collection for user resources.	
`name`	string	A free-text name for the user.	🔒
`description`	string	A free-text description of the user.	🔒
`domain`	string	The containing directory service domain.	🔒
`groups`	complex	A list of directory service groups for this user.	🔒

Example 19.1. An XML representation of a user resource

```
GET /api/users HTTP/1.1
Accept: application/xml

<user id="225f15cd-e891-434d-8262-a66808fcb9b1"
  href="/api/users/225f15cd-e891-434d-8262-a66808fcb9b1">
    <name>RHEV-M Admin</name>
    <actions/>
    <link rel="roles"
      href="/api/users/225f15cd-e891-434d-8262-a66808fcb9b1/roles"/>
    <link rel="tags"
      href="/api/users/225f15cd-e891-434d-8262-a66808fcb9b1/tags"/>
    <domain>domain.example.com</domain>
    <logged_in>false</logged_in>
    <user_name>rhevmadmin@domain.example.com</user_name>
    <groups>
        <group>Group Policy Creator Owners@domain.example.com/Users</group>
        <group>Domain Admins@domain.example.com/Users</group>
        <group>Enterprise Admins@domain.example.com/Users</group>
        <group>Schema Admins@domain.example.com/Users</group>
        <group>Administrators@domain.example.com/Builtin</group>
    </groups>
</user>
```

The API adds an existing directory service user to the Red Hat Enterprise Virtualization Manager database with a **`POST`** request to the **`users`** collection. The client-provided new user representation includes an embedded **`roles`** list with at least one initial **`role`** to assign to the user. For example, the following request assigns two initial roles to the user **`joe@domain.example.com`**:

Example 19.2. Adding a user from directory service and assigning two roles

```
POST /api/users HTTP/1.1
Content-Type: application/xml
Accept: application/xml

<user>
    <user_name>joe@domain.example.com</user_name>
    <roles>
        <role>
            <name>RHEVMPowerUser</name>
        </role>
        <role id="00000000-0000-0000-0001-000000000003"/>
    </roles>
</user>
```

The new user is identified either by Red Hat Enterprise Virtualization Manager user ID or via the directory service user principal name (UPN). The user ID format reported from the directory service domain might be different to the expected Red Hat Enterprise Virtualization Manager format, such as in LDIF [1] , the ID has the opposite byte order and is base-64 encoded. Hence it is usually more convenient to refer to the new user by UPN.

Note

The user exists in the directory service domain before it is added to the Red Hat Enterprise Virtualization Manager database. An API user has the option to query this domain through the **`domains`** collection prior to creation of the user.

Roles are identified either by name or ID. The example above shows both approaches.

Further roles are attached or detached with **`POST`** or **`DELETE`** requests to the roles sub-collection of an individual user. The example below illustrates how the API adds the **`RHEVMVDIUser`** role to the role assignments for a particular user.

[1] The LDAP Data Interchange Format is described in *RFC 2849* [http://tools.ietf.org/html/rfc2849].

Note

The embedded user roles list of the **`user`** element is only used for the initial creation. All interactions post-creation with the user's role assignments go through the **`roles`** sub-collection.

Example 19.3. Adding roles to a user

```
POST /api/users/225f15cd-e891-434d-8262-a66808fcb9b1/roles HTTP/1.1
Content-Type: application/xml
Accept: application/xml

<role>
    <name>RHEVMVDIUser</name>
</role>
```

Note

Users are not updated with the **`PUT`** verb. The only changes allowed post-creation are in the user's role assignments.

The API removes users from the Red Hat Enterprise Virtualization Manager database with a **`DELETE`** request on the **`users`** collection. The directory service domain remains unchanged after such a deletion.

Tags

The **tags** collection provides information about tags in a Red Hat Enterprise Virtualization environment. An API user accesses this information through the **rel="tags"** link obtained from the entry point URI (see *Chapter 4, Entry Point*).

The following table shows specific elements contained in a tag resource representation.

Note

The icons used in the properties column of this table are described in *Table 7.1, “Element property icons”*

Table 20.1. Tag elements

Element	Type	Description	Properties
host	GUID	A reference to the host which the tag is attached. See *Chapter 12, Hosts*.	🔒
user	GUID	A reference to the user which the tag is attached. See *Chapter 19, Users*.	🔒
vm	GUID	A reference to the VM which the tag is attached. See *Chapter 13, Virtual Machines*.	🔒
parent	complex	A reference to the VM which the tag is attached.	

Example 20.1. An XML representation of a tag resource

```
<tag id="f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e"
  href="/api/tags/f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e">
    <name>Finance</name>
    <description>Resources for the Finance department</description>
    <parent>
        <tag id="-1" href="/api/tags/-1"/>
    </parent>
</tag>
```

Creation of a new tag requires the **name** element. The **name**, **description** and **parent** elements are updatable post-creation.

20.1. Associating Tags With a Host, User or VM

The collection referenced by **link rel="tags"** from a **host**, **user** or **vms** represents the set of tags associated with the entity.

The **tag** representations are as described in *Chapter 20, Tags*, except they also contain a **host id**, **user id** or **vm id** reference to the entity in question.

Each tags collection is manipulated as described in *Chapter 7, Common Features*. Associating a tag with an entity is achieved by **POST**ing a tag reference (identifying the tag either by its **id** or **name**) to the collection.

Example 20.2. Associating a tag with a virtual machine

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/tags HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<tag>
    <name>Finance</name>
</tag>

HTTP/1.1 201 Created
Content-Type: application/xml

<tag id="f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e"
  href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/tags/
  f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e">
    <name>Finance</name>
    <description>Resources for the Finance department</description>
    <vm id="5114bb3e-a4e6-44b2-b783-b3eea7d84720"
      href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720"/>
</tag>
```

Removing an association is achieved with a **DELETE** request to the appropriate element in the collection.

Example 20.3. Removing a tag from a virtual machine

```
DELETE /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/tags/
f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e HTTP/1.1

HTTP/1.1 204 No Content
```

To query the set of entities associated with a given tag, the **collection/search** URI template for the appropriate collection should be used to search for entities matching **tag=MyTag**.

Example 20.4. Querying a collection for tagged resources

```
GET /api/vms?search=tag%3DFinance HTTP/1.1
Accept: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<vms>
    <vm id="5114bb3e-a4e6-44b2-b783-b3eea7d84720"
      href="/api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720">
        ...
    </vm>
    ...
</vms>
```

20.2. Parent Tags

An API user assigns a **`parent`** element to a tag to create a hierarchical link to a parent tag. The tags are presented as a flat collection, which descends from the **`root`** tag, with tag representations containing a link element to a parent tag

Note

The **`root`** tag is a special pseudo-tag assumed as the default parent tag if no parent tag is specified. The **`root`** tag cannot be deleted nor assigned a parent tag.

This tag hierarchy is expressed in the following way:

Example 20.5. Tag Hierarchy

```
<tags>
    <tag id="-1" href="/api/tags/-1">
        <name>root</name>
        <description>root</description>
        <parent>
            <tag id="-1" href="/api/tags/-1"/>
        </parent>
    </tag>
    <tag id="f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e"
      href="/api/tags/f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e">
        <name>Finance</name>
        <description>Resources for the Finance department</description>
        <parent>
            <tag id="-1" href="/api/tags/-1"/>
        </parent>
    </tag>
    <tag id="ty38wobf-23n5-18we-v18j-5u8t348cs7rt"
      href="/api/tags/ty38wobf-23n5-18we-v18j-5u8t348cs7rt">
        <name>Billing</name>
        <description>Billing Resources</description>
        <parent>
            <tag id="f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e"
              href="/api/tags/f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e"/>
        </parent>
    </tag>
</tags>
```

In this XML representation, the tags follow this hierarchy:

```
root                  (id: -1)
  - Finance           (id: f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e)
      - Billing       (id: ty38wobf-23n5-18we-v18j-5u8t348cs7rt)
```

`POST`ing a new tag with a **`parent`** element creates an association with a parent tag, using either the **`id`** attribute or the **`name`** element to reference the parent tag

Example 20.6. Setting an association with a parent tag with the id attribute

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/tags HTTP/1.1
Accept: application/xml
Content-Type: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<tag>
    <name>Billing</name>
    <description>Billing Resources</description>
    <parent>
        <tag id="f436ebfc-67f2-41bd-8ec6-902b6f7dcb5et"/>
    </parent>
</tag>
```

Example 20.7. Setting an association with a parent tag with the name element

```
POST /api/vms/5114bb3e-a4e6-44b2-b783-b3eea7d84720/tags HTTP/1.1
Accept: application/xml
Content-Type: application/xml

HTTP/1.1 200 OK
Content-Type: application/xml

<tag>
    <name>Billing</name>
    <description>Billing Resources</description>
    <parent>
        <tag>
            <name>Finance</name>
        </tag>
    </parent>
</tag>
```

A tag changes a parent using a **PUT** request:

Example 20.8. Changing the parent tag

```
PUT /api/tags/ty38wobf-23n5-18we-v18j-5u8t348cs7rt HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<tag>
    <parent>
        <tag id="f436ebfc-67f2-41bd-8ec6-902b6f7dcb5e"/>
    </parent>
</tag>
```

Events

The **rel="events"** link obtained from the entry point URI accesses the **events** collection and lists system events from Red Hat Enterprise Virtualization Manager.

Table 21.1. Event elements

Element	Type	Description
description	string	A description of the system event
code	integer	The integer event code. See *Appendix C, Event Codes* for a full list of event codes with descriptions.
severity	One of **normal**, **warning**, **error** or **alert**	The level of severity for the event.
time	**xsd:dateTime** format: **YYYY-MM-DDThh:mm:ss**	The timestamp indicating when the event happened.
user id	GUID	The identification code for the user who triggered the event.

Example 21.1. An XML representation of the events collection

```
<events>
    <event id="537" href="/api/events/537">
        <description>User vdcadmin logged in.</description>
        <code>30</code>
        <severity>normal</severity>
        <time>2011-01-12T10:48:27.827+02:00</time>
        <user id="9b9002d1-ec33-4083-8a7b-31f6b8931648"
          href="/api/users/9b9002d1-ec33-4083-8a7b-31f6b8931648"/>
    </event>
    ...
</events>
```

In addition to **user**, an **event** representation also contains a set of XML element relationships to resources relevant to the event.

Example 21.2. An XML representation of a virtual machine creation event

```
<event id="635" href="/api/events/635">
    <description>VM bar was created by rhevadmin.</description>
    <code>34</code>
    <severity>normal</severity>
    <time>2011-07-11T16:32:03.172+02:00</time>
    <user id="4621b611-43eb-4d2b-ae5f-1180850268c4"
      href="/api/users/4621b611-43eb-4d2b-ae5f-1180850268c4"/>
    <vm id="9b22d423-e16b-4dd8-9c06-c8e9358fbc66"
      href="/api/vms/9b22d423-e16b-4dd8-9c06-c8e9358fbc66"/>
    <storage_domain id="a8a0e93d-c570-45ab-9cd6-3c68ab31221f"
      href="/api/storagedomains/a8a0e93d-c570-45ab-9cd6-3c68ab31221f"/>
</event>
```

> This example representation provides XML element relationships to a virtual machine resource and a storage domain resource.

> **Note**
>
> The **`events`** collection is read-only.

21.1. Searching Events

The **`events`** collection provides search queries similar to other resource collections (See *Section 7.2.3, " Searching Collections with Queries "*). An additional feature when searching the **`events`** collection is the ability to search from a certain event. This queries all of events since a specified event.

Querying from an event requires an additional **`from`** argument added to the URI after the query. This **`from`** argument references an event **`id`** code.

Example 21.3. Searching from an event

```
GET /api/events?search=type%3D30&from=1012 HTTP/1.1
Accept: application/xml
```

This displays all events with **`type`** set to 30 since **`id="1012"`**

```
HTTP/1.1 200 OK
Content-Type: application/xml
<events>
    <event id="1018" href="/api/events/1018">
        <description>User admin logged in.</description>
        <code>30</code>
        <severity>normal</severity>
        <time>2011-07-11T14:03:22.485+10:00</time>
        <user id="80b71bae-98a1-11e0-8f20-525400866c73"
          href="/api/users/80b71bae-98a1-11e0-8f20-525400866c73"/>
    </event>
    <event id="1016" href="/api/events/1016">
        <description>User admin logged in.</description>
        <code>30</code>
        <severity>normal</severity>
        <time>2011-07-11T14:03:07.236+10:00</time>
        <user id="80b71bae-98a1-11e0-8f20-525400866c73"
          href="/api/users/80b71bae-98a1-11e0-8f20-525400866c73"/>
    </event>
    <event id="1014" href="/api/events/1014">
        <description>User admin logged in.</description>
        <code>30</code>
        <severity>normal</severity>
        <time>2011-07-11T14:02:16.009+10:00</time>
        <user id="80b71bae-98a1-11e0-8f20-525400866c73"
          href="/api/users/80b71bae-98a1-11e0-8f20-525400866c73"/>
    </event>
</events>
```

21.2. Paginating Events

A virtualization environment generates a large amount of events after a period of time. However, the API only displays a default number of events for one search query. To display more than the default, the API separates results into pages with the **page** command in a search query.

The following search query tells the API to paginate results using a **page** value in combination with the **sortby** clause:

sortby time asc page 1

The **sortby** clause defines the base element to order of the results and whether the results are ascending or descending. For search queries of **events**, set the base element to **time** and the order to ascending (**asc**) so the API displays all events from the creation of your virtualization environment.

The **page** condition defines the page number. One page equals the default number of events to list. Pagination begins at **page 1**. To view more pages, increase the **page** value:

sortby time asc page 2

sortby time asc page 3

sortby time asc page 4

Example 21.4. Paginating events

This example paginates **event** resources. The URL-encoded request is:

```
GET /api/events?search=sortby%20time%20asc%20page%201 HTTP/1.1
Accept: application/xml
```

Increase the **page** value to view the next page of results.

```
GET /api/events?search=sortby%20time%20asc%20page%202 HTTP/1.1
Accept: application/xml
```

Use an additional **from** argument to set the starting **id**.

```
GET /api/events?search=sortby%20time%20asc%20page%202&from=30 HTTP/1.1
Accept: application/xml
```

Appendix A. API Usage with cURL

This appendix provides instructions on adapting REST requests for use with **cURL**. **cURL** is a command line tool for transfering data across various protocols, including HTTP, and supports multiple platforms such as Linux, Windows, Mac OS and Solaris. Most Linux distributions include **cURL** as a package.

Installing cURL

A Red Hat Enterprise Linux user installs **cURL** with the following terminal command:

```
yum install curl
```

For other platforms, seek installation instructions on the **cURL** website (*http://curl.haxx.se/*).

Using cURL

cURL uses a command line interface to send requests to a HTTP server. Integrating a request requires the following command syntax:

```
Usage: curl [options] uri
```

The **`uri`** refers to target HTTP address to send the request. This is a location on your Red Hat Enterprise Virtualization Manager host within the API entry point path (**`/api`**).

cURL options

-X *`COMMAND`*, --request *`COMMAND`*

The request command to use. In the context of the REST API, use **`GET`**, **`POST`**, **`PUT`** or **`DELETE`**.

Example: **`-X GET`**

-H *`LINE`*, --header *`LINE`*

HTTP header to include with the request. Use multiple header options if more than one header is required.

Example: **`-H "Accept: application/xml" -H "Content-Type: application/xml"`**

-u *`USERNAME:PASSWORD`*, --user *`USERNAME:PASSWORD`*

The username and password of the Red Hat Enterprise Virtualization user. This attribute acts as a convenient replacement for the **`Authorization:`** header.

Example: **`-u admin@internal:p@55w0rd!`**

--cacert *`CERTIFICATE`*

The location of the certificate file for SSL communication to the REST API. The certificate file is saved locally on the client machine. Use the **`-k`** attribute to bypass SSL. See *Chapter 2, Authentication and Security* for more information on obtaining a certificate.

Example: **`--cacert ~/Certificates/rhevm.cer`**

-d *`BODY`*, --data *`BODY`*

The body to send for requests. Use with **`POST`**, **`PUT`** and **`DELETE`** requests. Ensure to specify the **`Content-Type: application/xml`** header if a body exists in the request.

Example: **`-d "<cdrom><file id='rhel-server-6.0-x86_64-dvd.iso'/></cdrom>"`**

Examples

The following examples show how to adapt REST requests to **cURL** command syntax:

Example A.1. **GET** request

The following **GET** request lists the virtual machines in the **vms** collection. Note that a **GET** request does not contain a body.

```
GET /api/vms HTTP/1.1
Accept: application/xml
```

Adapt the method (**GET**), header (**Accept: application/xml**) and URI (**https://*[RHEVM-Host]*:8443/api/vms**) into the following **cURL** command:

```
$ curl -X GET -H "Accept: application/xml" -u [USER:PASS] --cacert [CERT] https://[RHEVM-
Host]:8443/api/vms
```

An XML representation of the **vms** collection displays.

Example A.2. **POST** request

The following **POST** request creates a virtual machine in the **vms** collection. Note that a **POST** request requires a body.

```
POST /api/vms HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vm>
  <name>vm1</name>
  <cluster>
    <name>default</name>
  </cluster>
  <template>
    <name>Blank</name>
  </template>
  <memory>536870912</memory>
  <os>
    <boot dev="hd"/>
  </os>
</vm>
```

Adapt the method (**POST**), headers (**Accept: application/xml** and **Content-type: application/xml**), URI (**https://*[RHEVM-Host]*:8443/api/vms**) and request body into the following **cURL** command:

```
$ curl -X POST -H "Accept: application/xml" -H "Content-type: application/xml" -
u [USER:PASS] --cacert [CERT] -d "<vm><name>vm1</name><cluster><name>default</name></
cluster><template><name>Blank</name></template><memory>536870912</memory><os><boot
 dev='hd'/></os></vm>" https://[RHEVM-Host]:8443/api/vms
```

The REST API creates a new virtual machine and displays an XML representation of the resource.

Example A.3. **PUT** request

The following **PUT** request updates the memory of a virtual machine resource. Note that a **PUT** request requires a body.

```
PUT /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<vm>
    <memory>1073741824</memory>
</vm>
```

Adapt the method (**PUT**), headers (**Accept: application/xml** and **Content-type: application/xml**), URI (**https://*[RHEVM-Host]*:8443/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399**) and request body into the following **cURL** command:

```
$ curl -X PUT -H "Accept: application/xml" -H "Content-type: application/xml" -
u [USER:PASS] --cacert [CERT] -d "<vm><memory>1073741824</memory></vm>" https://[RHEVM-
Host]:8443//api/vms/082c794b-771f-452f-83c9-b2b5a19c039
```

The REST API updates the virtual machine with a new memory configuration.

Example A.4. **DELETE** request

The following **DELETE** request removes a virtual machine resource.

```
DELETE /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1
```

Adapt the method (**DELETE**) and URI (**https://*[RHEVM-Host]*:8443/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399**) into the following **cURL** command:

```
$ curl -X DELETE -u [USER:PASS] --cacert [CERT] https://[RHEVM-Host]:8443//api/
vms/082c794b-771f-452f-83c9-b2b5a19c039
```

The REST API removes the virtual machine. Note the **Accept: application/xml** request header is optional due to the empty result of **DELETE** requests.

Example A.5. **DELETE** request with body

The following **DELETE** request force removes a virtual machine resource as indicated with the optional body.

```
DELETE /api/vms/082c794b-771f-452f-83c9-b2b5a19c0399 HTTP/1.1
Accept: application/xml
Content-type: application/xml

<action>
  <force>true</force>
</action>
```

Adapt the method (**DELETE**), headers (**Accept: application/xml** and **Content-type: application/xml**), URI (**https://*[RHEVM-Host]*:8443/api/vms/082c794b-771f-452f-83c9-b2b5a19c0399**) and request body into the following **cURL** command:

```
$ curl -X DELETE -H "Accept: application/xml" -H "Content-type: application/xml" -
u [USER:PASS] --cacert [CERT] -d "<action><force>true</force></action>" https://[RHEVM-
Host]:8443//api/vms/082c794b-771f-452f-83c9-b2b5a19c039
```

The REST API force removes the virtual machine.

cURL Library (libcurl)

In addition the the standard command line tools, **cURL** also features **libcurl**, a library for programming language integration. For more information on supported programming languages and integration methods, see the **libcurl** website (*http://curl.haxx.se/libcurl/*).

Appendix B. Java Keystores

This appendix demonstrates how to import the X.509 certificate exported from the Red Hat Enterprise Virtualization server (See *Section 2.1, "TLS/SSL Certification"* for information on certificate exports) into a new Java keystore file.

Procedure B.1. Import a certificate into a new Java keystore

This process helps a user import the **rhevm.cer** certificate from *Section 2.1, "TLS/SSL Certification"* into a Java keystore. This procedure requires the **keytool** management utility from the Java Development Kit (JDK) available for Linux and Windows systems.

1. Access your client machine and locate the **rhevm.cer** certificate.

2. Import the **rhevm.cer** certificate using the Java **keytool** management utility.

   ```
   keytool -importcert -v -trustcacerts -keystore restapi.jks -noprompt -alias rhevm -file
    rhevm.cer
   ```

   The **keytool** utility creates a new keystore file named **restapi.jks**.

3. **keytool** asks for the keystore password. Enter a password and **keytool** asks to verify it.

4. **keytool** adds the **rhevm.cer** certificate to the **restapi.jks** keystore. Use **keytool -list** command to view the certificate's entry in the keystore:

   ```
   keytool -list -keystore restapi.jks -storepass [password]
   ```

Important

Some versions of **keytool** parse the certificate incorrectly. If **keytool** does not recognise the certificate, convert it to a different X.509 format with the **openssl** tool:

```
openssl x509 -in rhevm.cer -out rhevm.new -outform [pem|der]
```

This creates a file called **rhevm.new** to use in place of **rhevm.cer**.

Appendix C. Event Codes

This table lists all event codes for *Chapter 21, Events*.

Table C.1. Event codes

Code	Description
0	UNASSIGNED
1	VDC_START
2	VDC_STOP
12	HOST_FAILURE
13	HOST_DETECTED
14	HOST_RECOVER
15	HOST_MAINTENANCE
16	HOST_ACTIVATE
17	HOST_MAINTENANCE_FAILED
18	HOST_ACTIVATE_FAILED
19	HOST_RECOVER_FAILED
20	USER_HOST_START
21	USER_HOST_STOP
22	IRS_FAILURE
26	IRS_DISK_SPACE_LOW
30	USER_VDC_LOGIN
31	USER_VDC_LOGOUT
32	USER_RUN_VM
33	USER_STOP_VM
34	USER_ADD_VM
35	USER_UPDATE_VM
36	USER_REMOVE_VM
37	USER_ADD_VM_STARTED
38	USER_CHANGE_DISK_VM
39	USER_PAUSE_VM
40	USER_RESUME_VM
41	USER_HOST_RESTART
42	USER_ADD_HOST
43	USER_UPDATE_HOST
44	USER_REMOVE_HOST
45	USER_CREATE_SNAPSHOT
46	USER_TRY_BACK_TO_SNAPSHOT
47	USER_RESTORE_FROM_SNAPSHOT
48	USER_ADD_VM_TEMPLATE
49	USER_UPDATE_VM_TEMPLATE

Code	Description
50	USER_REMOVE_VM_TEMPLATE
51	USER_ADD_VM_TEMPLATE_FINISHED_SUCCESS
52	USER_ADD_VM_TEMPLATE_FINISHED_FAILURE
53	USER_ADD_VM_FINISHED_SUCCESS
54	USER_FAILED_RUN_VM
55	USER_FAILED_PAUSE_VM
56	USER_FAILED_STOP_VM
57	USER_FAILED_ADD_VM
58	USER_FAILED_UPDATE_VM
59	USER_FAILED_REMOVE_VM
60	USER_ADD_VM_FINISHED_FAILURE
61	VM_DOWN
62	VM_MIGRATION_START
63	VM_MIGRATION_DONE
64	VM_MIGRATION_ABORT
65	VM_MIGRATION_FAILED
66	VM_FAILURE
68	USER_CREATE_SNAPSHOT_FINISHED_SUCCESS
69	USER_CREATE_SNAPSHOT_FINISHED_FAILURE
70	USER_RUN_VM_AS_STATELESS_FINISHED_FAILURE
71	USER_TRY_BACK_TO_SNAPSHOT_FINISH_SUCCESS
72	USER_CHANGE_FLOPPY_VM
73	USER_INITIATED_SHUTDOWN_VM
74	USER_FAILED_SHUTDOWN_VM
75	USER_FAILED_CHANGE_FLOPPY_VM
76	USER_STOPPED_VM_INSTEAD_OF_SHUTDOWN
77	USER_FAILED_STOPPING_VM_INSTEAD_OF_SHUTDOWN
78	USER_ADD_DISK_TO_VM
79	USER_FAILED_ADD_DISK_TO_VM
80	USER_REMOVE_DISK_FROM_VM
81	USER_FAILED_REMOVE_DISK_FROM_VM
82	USER_MOVED_VM
83	USER_FAILED_MOVE_VM
84	USER_MOVED_TEMPLATE
85	USER_FAILED_MOVE_TEMPLATE
86	USER_COPIED_TEMPLATE
87	USER_FAILED_COPY_TEMPLATE
88	USER_UPDATE_VM_DISK

Code	Description
89	USER_FAILED_UPDATE_VM_DISK
90	USER_HOST_SHUTDOWN
91	USER_MOVED_VM_FINISHED_SUCCESS
92	USER_MOVED_VM_FINISHED_FAILURE
93	USER_MOVED_TEMPLATE_FINISHED_SUCCESS
94	USER_MOVED_TEMPLATE_FINISHED_FAILURE
95	USER_COPIED_TEMPLATE_FINISHED_SUCCESS
96	USER_COPIED_TEMPLATE_FINISHED_FAILURE
97	USER_ADD_DISK_TO_VM_FINISHED_SUCCESS
98	USER_ADD_DISK_TO_VM_FINISHED_FAILURE
99	USER_TRY_BACK_TO_SNAPSHOT_FINISH_FAILURE
100	USER_RESTORE_FROM_SNAPSHOT_FINISH_SUCCESS
101	USER_RESTORE_FROM_SNAPSHOT_FINISH_FAILURE
102	USER_FAILED_CHANGE_DISK_VM
103	USER_FAILED_RESUME_VM
104	USER_FAILED_ADD_HOST
105	USER_FAILED_UPDATE_HOST
106	USER_FAILED_REMOVE_HOST
107	USER_FAILED_HOST_RESTART
108	USER_FAILED_ADD_VM_TEMPLATE
109	USER_FAILED_UPDATE_VM_TEMPLATE
110	USER_FAILED_REMOVE_VM_TEMPLATE
111	USER_STOP_SUSPENDED_VM
112	USER_STOP_SUSPENDED_VM_FAILED
113	USER_REMOVE_VM_FINISHED
114	USER_VDC_LOGIN_FAILED
115	USER_FAILED_TRY_BACK_TO_SNAPSHOT
116	USER_FAILED_RESTORE_FROM_SNAPSHOT
117	USER_FAILED_CREATE_SNAPSHOT
118	USER_FAILED_HOST_START
119	VM_DOWN_ERROR
120	VM_MIGRATION_FAILED_FROM_TO
121	SYSTEM_HOST_RESTART
122	SYSTEM_FAILED_HOST_RESTART
123	HOST_SLOW_STORAGE_RESPONSE_TIME
124	VM_IMPORT
125	VM_IMPORT_FAILED
126	VM_NOT_RESPONDING

Code	Description
127	HOST_RUN_IN_NO_KVM_MODE
128	VM_MIGRATION_TRYING_RERUN
129	VM_CLEARED
130	USER_FAILED_HOST_SHUTDOWN
131	USER_EXPORT_VM
132	USER_EXPORT_VM_FAILED
133	USER_EXPORT_TEMPLATE
134	USER_EXPORT_TEMPLATE_FAILED
135	TEMPLATE_IMPORT
136	TEMPLATE_IMPORT_FAILED
137	USER_FAILED_HOST_STOP
138	VM_PAUSED_ENOSPC
139	VM_PAUSED_ERROR
140	VM_MIGRATION_FAILED_DURING_MOVE_TO_MAINTANANCE
141	HOST_VERSION_NOT_SUPPORTED_FOR_CLUSTER
142	VM_SET_TO_UNKNOWN_STATUS
143	VM_WAS_SET_DOWN_DUE_TO_HOST_REBOOT_OR_MANUAL_FENCE
144	VM_IMPORT_INFO
145	VM_BLK_VIRTIO_NO_CACHE
149	USER_ADD
150	USER_INITIATED_RUN_VM
151	USER_INITIATED_RUN_VM_FAILED
152	USER_RUN_VM_ON_NON_DEFAULT_HOST
153	USER_STARTED_VM
182	USER_FAILED_ATTACH_USER_TO_VM
201	IRS_DISK_SPACE_LOW_ERROR
204	IRS_HOSTED_ON_HOST
250	USER_UPDATE_VM_CLUSTER_DEFAULT_HOST_CLEARED
251	USER_REMOVE_VM_TEMPLATE_FINISHED
300	USER_ADD_VM_POOL
301	USER_ADD_VM_POOL_FAILED
302	USER_ADD_VM_POOL_WITH_VMS
303	USER_ADD_VM_POOL_WITH_VMS_FAILED
304	USER_REMOVE_VM_POOL
305	USER_REMOVE_VM_POOL_FAILED
306	USER_ADD_VM_TO_POOL
307	USER_ADD_VM_TO_POOL_FAILED
308	USER_REMOVE_VM_FROM_POOL

Code	Description
309	USER_REMOVE_VM_FROM_POOL_FAILED
310	USER_ATTACH_USER_TO_POOL
311	USER_ATTACH_USER_TO_POOL_FAILED
312	USER_DETACH_USER_FROM_POOL
313	USER_DETACH_USER_FROM_POOL_FAILED
314	USER_UPDATE_VM_POOL
315	USER_UPDATE_VM_POOL_FAILED
316	USER_ATTACH_USER_TO_VM_FROM_POOL
317	USER_ATTACH_USER_TO_VM_FROM_POOL_FAILED
318	USER_ATTACH_USER_TO_VM_FROM_POOL_FINISHED_SUCCESS
319	USER_ATTACH_USER_TO_VM_FROM_POOL_FINISHED_FAILURE
320	USER_ADD_VM_POOL_WITH_VMS_ADD_HOST_FAILED
325	USER_REMOVE_ADUSER
326	USER_FAILED_REMOVE_ADUSER
327	USER_FAILED_ADD_ADUSER
328	USER_ATTACH_USER_TO_TIME_LEASED_POOL
329	USER_ATTACH_USER_TO_TIME_LEASED_POOL_FAILED
330	USER_DETACH_USER_FROM_TIME_LEASED_POOL
331	USER_DETACH_USER_FROM_TIME_LEASED_POOL_FAILED
332	USER_ATTACH_AD_GROUP_TO_TIME_LEASED_POOL
333	USER_ATTACH_AD_GROUP_TO_TIME_LEASED_POOL_FAILED
334	USER_DETACH_AD_GROUP_FROM_TIME_LEASED_POOL
335	USER_DETACH_AD_GROUP_FROM_TIME_LEASED_POOL_FAILED
336	USER_UPDATE_USER_TO_TIME_LEASED_POOL
337	USER_UPDATE_USER_TO_TIME_LEASED_POOL_FAILED
338	USER_UPDATE_AD_GROUP_TO_TIME_LEASED_POOL
339	USER_UPDATE_AD_GROUP_TO_TIME_LEASED_POOL_FAILED
342	USER_MERGE_SNAPSHOT
343	USER_FAILED_MERGE_SNAPSHOT
344	USER_UPDATE_VM_POOL_WITH_VMS
345	USER_UPDATE_VM_POOL_WITH_VMS_FAILED
346	USER_PASSWORD_CHANGED
347	USER_PASSWORD_CHANGE_FAILED
348	USER_CLEAR_UNKNOWN_VMS
349	USER_FAILED_CLEAR_UNKNOWN_VMS
350	USER_ADD_BOOKMARK
351	USER_ADD_BOOKMARK_FAILED
352	USER_UPDATE_BOOKMARK

Code	Description
353	USER_UPDATE_BOOKMARK_FAILED
354	USER_REMOVE_BOOKMARK
355	USER_REMOVE_BOOKMARK_FAILED
356	USER_MERGE_SNAPSHOT_FINISHED_SUCCESS
357	USER_MERGE_SNAPSHOT_FINISHED_FAILURE
360	USER_DETACH_USER_FROM_VM
361	USER_FAILED_DETACH_USER_FROM_VM
400	USER_ATTACH_VM_TO_AD_GROUP
401	USER_ATTACH_VM_TO_AD_GROUP_FAILED
402	USER_DETACH_VM_TO_AD_GROUP
403	USER_DETACH_VM_TO_AD_GROUP_FAILED
404	USER_ATTACH_VM_POOL_TO_AD_GROUP
405	USER_ATTACH_VM_POOL_TO_AD_GROUP_FAILED
406	USER_DETACH_VM_POOL_TO_AD_GROUP
407	USER_DETACH_VM_POOL_TO_AD_GROUP_FAILED
408	USER_REMOVE_AD_GROUP
409	USER_REMOVE_AD_GROUP_FAILED
430	USER_UPDATE_TAG
431	USER_UPDATE_TAG_FAILED
432	USER_ADD_TAG
433	USER_ADD_TAG_FAILED
434	USER_REMOVE_TAG
435	USER_REMOVE_TAG_FAILED
436	USER_ATTACH_TAG_TO_USER
437	USER_ATTACH_TAG_TO_USER_FAILED
438	USER_ATTACH_TAG_TO_USER_GROUP
439	USER_ATTACH_TAG_TO_USER_GROUP_FAILED
440	USER_ATTACH_TAG_TO_VM
441	USER_ATTACH_TAG_TO_VM_FAILED
442	USER_ATTACH_TAG_TO_HOST
443	USER_ATTACH_TAG_TO_HOST_FAILED
444	USER_DETACH_HOST_FROM_TAG
445	USER_DETACH_HOST_FROM_TAG_FAILED
446	USER_DETACH_VM_FROM_TAG
447	USER_DETACH_VM_FROM_TAG_FAILED
448	USER_DETACH_USER_FROM_TAG
449	USER_DETACH_USER_FROM_TAG_FAILED
450	USER_DETACH_USER_GROUP_FROM_TAG

Code	Description
451	USER_DETACH_USER_GROUP_FROM_TAG_FAILED
452	USER_ATTACH_TAG_TO_USER_EXISTS
453	USER_ATTACH_TAG_TO_USER_GROUP_EXISTS
454	USER_ATTACH_TAG_TO_VM_EXISTS
455	USER_ATTACH_TAG_TO_HOST_EXISTS
456	USER_LOGGED_IN_VM
457	USER_LOGGED_OUT_VM
458	USER_LOCKED_VM
459	USER_UNLOCKED_VM
460	USER_DETACH_USER_FROM_TIME_LEASED_POOL_INTERNAL
461	USER_DETACH_USER_FROM_TIME_LEASED_POOL_FAILED_INTERNAL
462	USER_DETACH_AD_GROUP_FROM_TIME_LEASED_POOL_INTERNAL
463	USER_DETACH_AD_GROUP_FROM_TIME_LEASED_POOL_FAILED_INTERNAL
467	UPDATE_TAGS_VM_DEFAULT_DISPLAY_TYPE
468	UPDATE_TAGS_VM_DEFAULT_DISPLAY_TYPE_FAILED
470	USER_ATTACH_VM_POOL_TO_AD_GROUP_INTERNAL
471	USER_ATTACH_VM_POOL_TO_AD_GROUP_FAILED_INTERNAL
472	USER_ATTACH_USER_TO_POOL_INTERNAL
473	USER_ATTACH_USER_TO_POOL_FAILED_INTERNAL
494	HOST_MANUAL_FENCE_STATUS
495	HOST_MANUAL_FENCE_STATUS_FAILED
496	HOST_FENCE_STATUS
497	HOST_FENCE_STATUS_FAILED
498	HOST_APPROVE
499	HOST_APPROVE_FAILED
500	HOST_FAILED_TO_RUN_VMS
501	USER_SUSPEND_VM
502	USER_FAILED_SUSPEND_VM
503	USER_SUSPEND_VM_OK
504	HOST_INSTALL
505	HOST_INSTALL_FAILED
506	HOST_INITIATED_RUN_VM
507	HOST_INITIATED_RUN_VM_FAILED
509	HOST_INSTALL_IN_PROGRESS
510	HOST_INSTALL_IN_PROGRESS_WARNING
511	HOST_INSTALL_IN_PROGRESS_ERROR
512	USER_SUSPEND_VM_FINISH_SUCCESS
513	HOST_RECOVER_FAILED_VMS_UNKNOWN

Code	Description
514	HOST_INITIALIZING
515	HOST_CPU_LOWER_THAN_CLUSTER
516	HOST_CPU_RETRIEVE_FAILED
517	HOST_SET_NONOPERATIONAL
518	HOST_SET_NONOPERATIONAL_FAILED
519	HOST_SET_NONOPERATIONAL_NETWORK
520	USER_ATTACH_USER_TO_VM
521	USER_SUSPEND_VM_FINISH_FAILURE
522	HOST_SET_NONOPERATIONAL_DOMAIN
523	HOST_SET_NONOPERATIONAL_DOMAIN_FAILED
524	AUTO_SUSPEND_VM
524	HOST_DOMAIN_DELAY_INTERVAL
525	AUTO_SUSPEND_VM_FINISH_SUCCESS
526	AUTO_SUSPEND_VM_FINISH_FAILURE
527	AUTO_FAILED_SUSPEND_VM
528	USER_EJECT_VM_DISK
529	USER_EJECT_VM_FLOPPY
530	HOST_MANUAL_FENCE_FAILED_CALL_FENCE_SPM
531	HOST_LOW_MEM
555	USER_MOVE_TAG
556	USER_MOVE_TAG_FAILED
600	USER_HOST_MAINTENANCE
601	CPU_FLAGS_NX_IS_MISSING
602	USER_HOST_MAINTENANCE_MIGRATION_FAILED
603	HOST_SET_NONOPERATIONAL_IFACE_DOWN
800	IMAGES_SYNCRONIZER_DESKTOP_NOT_EXIST_IN_VDC
801	IMAGES_SYNCRONIZER_TEMPLATE_NOT_EXIST_IMAGE_EXIST
802	IMAGES_SYNCRONIZER_SNAPSHOT_NOT_EXIST_IN_VDC
803	IMAGES_SYNCRONIZER_SNAPSHOTS_NOT_ATTACHED_TO_VM_IN_VDC
804	IMAGES_SYNCRONIZER_TEMPLATE_NOT_EXIST_IN_VDC
805	IMAGES_SYNCRONIZER_DESKTOP_NOT_EXIST_IN_IRS
806	IMAGES_SYNCRONIZER_SNAPSHOT_NOT_EXIST_IN_IRS
807	IMAGES_SYNCRONIZER_DESKTOP_WITHOUT_TEMPLATE_VDC
808	IMAGES_SYNCRONIZER_IMAGE_TEMPLATE_NOT_EXIST
809	USER_ADD_HOST_GROUP
810	USER_ADD_HOST_GROUP_FAILED
811	USER_UPDATE_HOST_GROUP
812	USER_UPDATE_HOST_GROUP_FAILED

Code	Description
813	USER_REMOVE_HOST_GROUP
814	USER_REMOVE_HOST_GROUP_FAILED
815	USER_VDC_LOGOUT_FAILED
816	MAC_POOL_EMPTY
817	CERTIFICATE_FILE_NOT_FOUND
818	RUN_VM_FAILED
819	HOST_REGISTER_ERROR_UPDATING_HOST
820	HOST_REGISTER_ERROR_UPDATING_HOST_ALL_TAKEN
821	HOST_REGISTER_HOST_IS_ACTIVE
822	HOST_REGISTER_ERROR_UPDATING_NAME
823	HOST_REGISTER_ERROR_UPDATING_NAMES_ALL_TAKEN
824	HOST_REGISTER_NAME_IS_ACTIVE
825	HOST_REGISTER_AUTO_APPROVE_PATTERN
826	HOST_REGISTER_FAILED
827	HOST_REGISTER_EXISTING_HOST_UPDATE_FAILED
828	HOST_REGISTER_SUCCEEDED
829	VM_MIGRATION_ON_CONNECT_CHECK_FAILED
830	VM_MIGRATION_ON_CONNECT_CHECK_SUCCEEDED
831	USER_DEDICATE_VM_TO_POWERCLIENT
832	USER_DEDICATE_VM_TO_POWERCLIENT_FAILED
833	MAC_ADDRESS_IS_IN_USE
835	SYSTEM_UPDATE_HOST_GROUP
836	SYSTEM_UPDATE_HOST_GROUP_FAILED
850	USER_ADD_PERMISSION
851	USER_ADD_PERMISSION_FAILED
852	USER_REMOVE_PERMISSION
853	USER_REMOVE_PERMISSION_FAILED
854	USER_ADD_ROLE
855	USER_ADD_ROLE_FAILED
856	USER_UPDATE_ROLE
857	USER_UPDATE_ROLE_FAILED
858	USER_REMOVE_ROLE
859	USER_REMOVE_ROLE_FAILED
860	USER_ATTACHED_ACTION_GROUP_TO_ROLE
861	USER_ATTACHED_ACTION_GROUP_TO_ROLE_FAILED
862	USER_DETACHED_ACTION_GROUP_FROM_ROLE
863	USER_DETACHED_ACTION_GROUP_FROM_ROLE_FAILED
864	USER_ADD_ROLE_WITH_ACTION_GROUP

Code	Description
865	USER_ADD_ROLE_WITH_ACTION_GROUP_FAILED
900	AD_COMPUTER_ACCOUNT_SUCCEEDED
901	AD_COMPUTER_ACCOUNT_FAILED
920	NETWORK_ATTACH_NETWORK_TO_HOST
921	NETWORK_ATTACH_NETWORK_TO_HOST_FAILED
922	NETWORK_DETACH_NETWORK_FROM_HOST
923	NETWORK_DETACH_NETWORK_FROM_HOST_FAILED
924	NETWORK_ADD_BOND
925	NETWORK_ADD_BOND_FAILED
926	NETWORK_REMOVE_BOND
927	NETWORK_REMOVE_BOND_FAILED
928	NETWORK_HOST_NETWORK_MATCH_CLUSTER
929	NETWORK_HOST_NETWORK_NOT_MATCH_CLUSTER
930	NETWORK_REMOVE_VM_INTERFACE
931	NETWORK_REMOVE_VM_INTERFACE_FAILED
932	NETWORK_ADD_VM_INTERFACE
933	NETWORK_ADD_VM_INTERFACE_FAILED
934	NETWORK_UPDATE_VM_INTERFACE
935	NETWORK_UPDATE_VM_INTERFACE_FAILED
936	NETWORK_ADD_TEMPLATE_INTERFACE
937	NETWORK_ADD_TEMPLATE_INTERFACE_FAILED
938	NETWORK_REMOVE_TEMPLATE_INTERFACE
939	NETWORK_REMOVE_TEMPLATE_INTERFACE_FAILED
940	NETWORK_UPDATE_TEMPLATE_INTERFACE
941	NETWORK_UPDATE_TEMPLATE_INTERFACE_FAILED
942	NETWORK_ADD_NETWORK
943	NETWORK_ADD_NETWORK_FAILED
944	NETWORK_REMOVE_NETWORK
945	NETWORK_REMOVE_NETWORK_FAILED
946	NETWORK_ATTACH_NETWORK_TO_HOST_GROUP
947	NETWORK_ATTACH_NETWORK_TO_HOST_GROUP_FAILED
948	NETWORK_DETACH_NETWORK_TO_HOST_GROUP
949	NETWORK_DETACH_NETWORK_TO_HOST_GROUP_FAILED
950	USER_ADD_STORAGE_POOL
951	USER_ADD_STORAGE_POOL_FAILED
952	USER_UPDATE_STORAGE_POOL
953	USER_UPDATE_STORAGE_POOL_FAILED
954	USER_REMOVE_STORAGE_POOL

Code	Description
955	USER_REMOVE_STORAGE_POOL_FAILED
956	USER_ADD_STORAGE_DOMAIN
957	USER_ADD_STORAGE_DOMAIN_FAILED
958	USER_UPDATE_STORAGE_DOMAIN
959	USER_UPDATE_STORAGE_DOMAIN_FAILED
960	USER_REMOVE_STORAGE_DOMAIN
961	USER_REMOVE_STORAGE_DOMAIN_FAILED
962	USER_ATTACH_STORAGE_DOMAIN_TO_POOL
963	USER_ATTACH_STORAGE_DOMAIN_TO_POOL_FAILED
964	USER_DETACH_STORAGE_DOMAIN_FROM_POOL
965	USER_DETACH_STORAGE_DOMAIN_FROM_POOL_FAILED
966	USER_ACTIVATED_STORAGE_DOMAIN
967	USER_ACTIVATE_STORAGE_DOMAIN_FAILED
968	USER_DEACTIVATED_STORAGE_DOMAIN
969	USER_DEACTIVATE_STORAGE_DOMAIN_FAILED
970	SYSTEM_DEACTIVATED_STORAGE_DOMAIN
971	SYSTEM_DEACTIVATE_STORAGE_DOMAIN_FAILED
972	USER_EXTENDED_STORAGE_DOMAIN
973	USER_EXTENDED_STORAGE_DOMAIN_FAILED
974	USER_REMOVE_VG
975	USER_REMOVE_VG_FAILED
976	USER_ACTIVATE_STORAGE_POOL
977	USER_ACTIVATE_STORAGE_POOL_FAILED
978	SYSTEM_FAILED_CHANGE_STORAGE_POOL_STATUS
979	SYSTEM_CHANGE_STORAGE_POOL_STATUS_NO_HOST_FOR_SPM
980	SYSTEM_CHANGE_STORAGE_POOL_STATUS_PROBLEMATIC
981	USER_FORCE_REMOVE_STORAGE_DOMAIN
982	USER_FORCE_REMOVE_STORAGE_DOMAIN_FAILED
983	RECONSTRUCT_MASTER_FAILED_NO_MASTER
984	RECONSTRUCT_MASTER_DONE
985	RECONSTRUCT_MASTER_FAILED
986	SYSTEM_CHANGE_STORAGE_POOL_STATUS_PROBLEMATIC_SEARCHING_NEW_SPM
987	SYSTEM_CHANGE_STORAGE_POOL_STATUS_PROBLEMATIC_WITH_ERROR
988	USER_CONNECT_HOSTS_TO_LUN_FAILED
989	SYSTEM_CHANGE_STORAGE_POOL_STATUS_PROBLEMATIC_FROM_NON_OPERATIONAL
990	SYSTEM_MASTER_DOMAIN_NOT_IN_SYNC
991	RECOVERY_STORAGE_POOL
992	RECOVERY_STORAGE_POOL_FAILED

Code	Description
993	SYSTEM_CHANGE_STORAGE_POOL_STATUS_RESET_IRS
994	CONNECT_STORAGE_SERVERS_FAILED
995	CONNECT_STORAGE_POOL_FAILED
996	STORAGE_DOMAIN_ERROR
1100	NETWORK_UPDATE_DISPLAY_TO_HOST_GROUP
1101	NETWORK_UPDATE_DISPLAY_TO_HOST_GROUP_FAILED
1102	NETWORK_UPDATE_NETWORK_TO_HOST_INTERFACE
1103	NETWORK_UPDATE_NETWORK_TO_HOST_INTERFACE_FAILED
1104	NETWORK_COMMINT_NETWORK_CHANGES
1105	NETWORK_COMMINT_NETWORK_CHANGES_FAILED
1106	NETWORK_HOST_USING_WRONG_CLUSER_VLAN
1107	NETWORK_HOST_MISSING_CLUSER_VLAN
1150	IMPORTEXPORT_EXPORT_VM
1151	IMPORTEXPORT_EXPORT_VM_FAILED
1152	IMPORTEXPORT_IMPORT_VM
1153	IMPORTEXPORT_IMPORT_VM_FAILED
1154	IMPORTEXPORT_REMOVE_TEMPLATE
1155	IMPORTEXPORT_REMOVE_TEMPLATE_FAILED
1156	IMPORTEXPORT_EXPORT_TEMPLATE
1157	IMPORTEXPORT_EXPORT_TEMPLATE_FAILED
1158	IMPORTEXPORT_IMPORT_TEMPLATE
1159	IMPORTEXPORT_IMPORT_TEMPLATE_FAILED
1160	IMPORTEXPORT_REMOVE_VM
1161	IMPORTEXPORT_REMOVE_VM_FAILED
1162	IMPORTEXPORT_STARTING_EXPORT_VM
1163	IMPORTEXPORT_STARTING_IMPORT_TEMPLATE
1164	IMPORTEXPORT_STARTING_EXPORT_TEMPLATE
1165	IMPORTEXPORT_STARTING_IMPORT_VM
1166	IMPORTEXPORT_STARTING_REMOVE_TEMPLATE
1167	IMPORTEXPORT_STARTING_REMOVE_VM
1168	IMPORTEXPORT_FAILED_TO_IMPORT_VM
1169	IMPORTEXPORT_FAILED_TO_IMPORT_TEMPLATE
9000	HOST_ALERT_FENCING_IS_NOT_CONFIGURED
9001	HOST_ALERT_FENCING_TEST_FAILED
9002	HOST_ALERT_FENCING_OPERATION_FAILED
9003	HOST_ALERT_FENCING_OPERATION_SKIPPED
9004	HOST_ALERT_FENCING_NO_PROXY_HOST
9500	TASK_STOPPING_ASYNC_TASK

Code	Description
9501	TASK_CLEARING_ASYNC_TASK

Appendix D. Timezones

The API maps Windows Standard Format timezone names to tz database format when specifying a timezone for a virtual machine or VM template. This means the API only accepts certain tz database codes, which the following table lists:

Table D.1. Accepted tz database codes

tz database Format	Windows Standard Format
Africa/Cairo	Egypt Standard Time
Africa/Casablanca	Morocco Standard Time
Africa/Johannesburg	South Africa Standard Time
Africa/Lagos	W. Central Africa Standard Time
Africa/Nairobi	E. Africa Standard Time
Africa/Reykjavik	Greenwich Standard Time
Africa/Windhoek	Namibia Standard Time
America/Anchorage	Alaskan Standard Time
America/Bogota	SA Pacific Standard Time
America/Buenos_Aires	Argentina Standard Time
America/Caracas	Venezuela Standard Time
America/Chicago	Central Standard Time
America/Chihuahua	Mexico Standard Time
America/Chihuahua	Mountain Standard Time
America/Denver	Mountain Standard Time
America/Godthab	Greenland Standard Time
America/Guatemala	Central America Standard Time
America/Halifax	Atlantic Standard Time
America/La_Paz	SA Western Standard Time
America/Los_Angeles	Pacific Standard Time
America/Manaus	Central Brazilian Standard Time
America/Mexico_City	Central Standard Time
America/Mexico_City	Mexico Standard Time
America/Montevideo	Montevideo Standard Time
America/New_York	Eastern Standard Time
America/Phoenix	US Mountain Standard Time
America/Regina	Canada Central Standard Time
America/Santiago	Pacific SA Standard Time
America/Sao_Paulo	E. South America Standard Time
America/St_Johns	Newfoundland Standard Time
America/Tijuana	Pacific Standard Time
Asia/Amman	Jordan Standard Time
Asia/Baghdad	Arabic Standard Time

tz database Format	Windows Standard Format
Asia/Baku	Azerbaijan Standard Time
Asia/Bangkok	SE Asia Standard Time
Asia/Beirut	Middle East Standard Time
Asia/Calcutta	India Standard Time
Asia/Colombo	Sri Lanka Standard Time
Asia/Dhaka	Central Asia Standard Time
Asia/Dubai	Arabian Standard Time
Asia/Irkutsk	North Asia East Standard Time
Asia/Jerusalem	Israel Standard Time
Asia/Kabul	Afghanistan Standard Time
Asia/Karachi	Pakistan Standard Time
Asia/Katmandu	Nepal Standard Time
Asia/Krasnoyarsk	North Asia Standard Time
Asia/Novosibirsk	N. Central Asia Standard Time
Asia/Rangoon	Myanmar Standard Time
Asia/Riyadh	Arab Standard Time
Asia/Seoul	Korea Standard Time
Asia/Shanghai	China Standard Time
Asia/Singapore	Singapore Standard Time
Asia/Taipei	Taipei Standard Time
Asia/Tashkent	West Asia Standard Time
Asia/Tehran	Iran Standard Time
Asia/Tokyo	Tokyo Standard Time
Asia/Vladivostok	Vladivostok Standard Time
Asia/Yakutsk	Yakutsk Standard Time
Asia/Yekaterinburg	Ekaterinburg Standard Time
Asia/Yerevan	Armenian Standard Time
Asia/Yerevan	Caucasus Standard Time
Atlantic/Azores	Azores Standard Time
Atlantic/Cape_Verde	Cape Verde Standard Time
Atlantic/South_Georgia	Mid-Atlantic Standard Time
Australia/Adelaide	Cen. Australia Standard Time
Australia/Brisbane	E. Australia Standard Time
Australia/Darwin	AUS Central Standard Time
Australia/Hobart	Tasmania Standard Time
Australia/Perth	W. Australia Standard Time
Australia/Sydney	AUS Eastern Standard Time
Etc/GMT-3	Georgian Standard Time

tz database Format	Windows Standard Format
Etc/GMT+12	Dateline Standard Time
Etc/GMT+3	SA Eastern Standard Time
Etc/GMT+5	US Eastern Standard Time
Europe/Berlin	W. Europe Standard Time
Europe/Budapest	Central Europe Standard Time
Europe/Istanbul	GTB Standard Time
Europe/Kiev	FLE Standard Time
Europe/London	GMT Standard Time
Europe/Minsk	E. Europe Standard Time
Europe/Moscow	Russian Standard Time
Europe/Paris	Romance Standard Time
Europe/Warsaw	Central European Standard Time
Indian/Mauritius	Mauritius Standard Time
Pacific/Apia	Samoa Standard Time
Pacific/Auckland	New Zealand Standard Time
Pacific/Fiji	Fiji Standard Time
Pacific/Guadalcanal	Central Pacific Standard Time
Pacific/Honolulu	Hawaiian Standard Time
Pacific/Port_Moresby	West Pacific Standard Time
Pacific/Tongatapu	Tonga Standard Time

Appendix E. Revision History

Revision 1-39 Thu Dec 1 2011 Daniel Macpherson *dmacpher@redhat.com*

BZ#759326 - Removed inaccurate instruction in force delete vm
BZ#757123 - Change to vm start action example for host's placement_policy.
BZ#754574 - Correction to XML examples.
BZ#754351 - Corrections to cURL and XML examples.

Revision 1-38 Thu Nov 24 2011 Stephen Gordon *sgordon@redhat.com*

BZ#756789 - Corrections to arguments in cURL examples.

Revision 1-37 Fri Nov 11 2011 Daniel Macpherson *dmacpher@redhat.com*

BZ#750323, BZ#750325 - Corrections to typos in examples.

Revision 1-36 Mon Oct 31 2011 Daniel Macpherson *dmacpher@redhat.com*

Minor corrections.

Revision 1-35 Thu Oct 27 2011 Daniel Macpherson *dmacpher@redhat.com*

Added admonitions for sections excluded from Backwards Compatibility Statement.
Revised cURL Usage appendix to ensure markup is similar to other RHEV documentation.

Revision 1-34 Thu Oct 27 2011 Daniel Macpherson *dmacpher@redhat.com*

Added Backwards Compatibility Statement to Preface.

Revision 1-33 Wed Oct 26 2011 Daniel Macpherson *dmacpher@redhat.com*

Added three examples to cURL integration appendix.

Revision 1-32 Tue Oct 25 2011 Daniel Macpherson *dmacpher@redhat.com*

Proofread examples.
Added cURL integration Appendix.
Modified Storage Domains Sub-Collection section in Data Center chapter to make Attaching Storage Domain instructions prominent in TOC.

Revision 1-31 Thu Oct 19 2011 Daniel Macpherson *dmacpher@redhat.com*

Minor changes for localization purposes.

Revision 1-30 Wed Oct 12 2011 Daniel Macpherson *dmacpher@redhat.com*

BZ#739993 - Added Pagination section under Search Queries in the Common Features chapter.
BZ#743592 - Added sentence on setting no preferred host in placement_policy for Virtual Machines.

Minor changes for localization purposes.

Revision 1-29 **Tue Oct 4 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

Minor fixes to typos in examples.

Revision 1-28 **Tue Oct 4 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#742863 - Capabilities chapter: Changed CPU types in example.
BZ#742775 - Coverted enumerated values to lowercase.

Revision 1-27 **Fri Sept 30 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#740523 - Added section on cloning disks from a virtual machine template.
Minor changes for localization purposes.

Revision 1-26 **Wed Sept 28 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#739993 - Added section for paginating events.

Revision 1-25 **Fri Sept 22 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#738478 - Removal of element in the Example chapter
BZ#739897 - Added host element to storage domain creation
BZ#740471 - Amended section on querying collections to include sortby instructions
BZ#740401, BZ#740402 - Completed docs Quality Engineering Review

Revision 1-23 **Fri Sept 19 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#737860 - Minor change to VM origin in VM creation example.

Revision 1-22 **Fri Sept 9 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#734463 - Updated element tables in VM disks, VM NICs and Host NICs sub-collections to include statistics link.
BZ#734634 - Fixed storage_domains element in example and VM disks section.

Revision 1-21 **Thu Sept 8 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#734436 - Added elements for product_info in Entry Point chapter.
BZ#695635 - Added minor addition to DELETE methods without Content-Type header.
BZ#719501 - Added sysprep (domain and timezone) documentation for virtual machines and templates.
BZ#734634 - Fixed missing storage domain in disk creation example.
BZ#734463 - Added/improved statistics documentation for Hosts, Host NICs, VMs, VM disks and VM NICs.

Revision 1-18 **Thu Aug 24 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

BZ#695635 - Added new section on force remove action for data centers.
BZ#680858 - Added new section on force remove action for virtual machines.

BZ#702895 - Provided additional information on changing virtual machine CD-ROM images.
BZ#730565 - Added new section for TLS/SSL certification and an Appendix for importing a certificate to a Java keystore.
BZ#729898 - Added references to Content-type: xml/application header for requests that require a body. Reviewed headers in all examples.
BZ#720593 - Included documentation on event search queries using the "from" URI parameter.
BZ#719570 - Updated Network Interface sub-collection in Virtual Machines chapter to specify references to network are made with either "id" or "name" when creating or modifying a new resource. Updated Network Interface sub-collection in Templates chapter to remove read-only status and specify same rules as Network Interface sub-collection in Virtual Machines chapter.

Revision 1-17 **Thu Aug 04 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Final QE Revisions

Revision 1-16 **Thu Aug 04 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Minor change to entry point image

Revision 1-15 **Thu Aug 04 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Minor fixes to content

Revision 1-14 **Thu Aug 04 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Revised Guide as per Quality Engineering feedback (BZ#723767, BZ#723773, BZ#723776, BZ#723769, BZ#723768, BZ#723770, BZ#723774, BZ#723775, BZ#723777, BZ#723771)

Revision 1-12 **Tue Jul 19 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Revised Guide as per Tech Review

Revision 1-11 **Thu Jul 7 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Updated Preface to include Documentation Suite

Revision 1-10 **Thu Jul 7 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Minor edits to content

Revision 1-9 **Thu Jul 7 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Minor edits to content
Revised Entry Point Image to include new collections

Revision 1-8 **Wed Jul 6 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Finalization of draft content

Revision 1-6 **Thu Jun 30 2011** **Daniel Macpherson** *dmacpher@redhat.com*

Revised structure for version 3.0

Revision 1-5 **Mon Jun 20 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

Added new elements and document structure for version 3.0

Revision 1-4 **Thu Jun 16 2011** **Daniel Macpherson** *`dmacpher@redhat.com`*

Draft revision of document
Inclusion of Example chapter

Revision 0-0 **Thu Jun 17 2010** **Mark McLoughlin** *`markmc@redhat.com`*

Initial creation