开发手册

vCenter

产品版本：ZStack 2.5.1

文档版本：V2.5.1

ZStack

版权声明

商标说明

ZStack商标和其他云轴商标均为上海云轴信息科技有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受上海云轴公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，上海云轴公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1 引言

产品版本

目前与本文档相对应的产品版本为：ZStack 2.5.1

读者对象

本文档详述了ZStack 2.5.1 RESTful API的使用规范，并提供所有API的详细定义。本文档主要适用于以下读者：

- 架构设计师
- 开发工程师
- 测试工程师
- 项目实施人员
- 对ZStack有兴趣研究的相关人员

版本更新

2.5.1

2018/07/10主要更新：

1. Shared Block主存储功能增强
   - 一个集群支持挂载多个Shared Block主存储
   - 跨Shared Block主存储的整机迁移
2. 资源编排支持普通账户/企业管理账号体系使用
3. 修复已知问题，提升系统稳定性

2.5.0

2018/07/05主要更新：

1. 资源编排
2. 整机克隆
3. vCenter接管功能增强
   - vCenter监控报警
   - 多vCenter区分
   - 独立CPU授权

4. 操作日志/审计信息优化展示
5. 性能Top5页面展示优化
6. 其它相关功能和优化
   - 新增多个操作场景进度条
   - 操作助手和帮助文档
   - 优化界面交互
   - 优化部分业务逻辑

2.4.0

2018/06/11主要更新：

1. 企业管理模块：项目管理、工单审批、独立区域管理
2. 支持ARM服务器
3. 应用中心
4. 资源监控增强
   - 详情页资源监控
   - 资源实时监控
5. 新增主存储类型：Shared Block共享块存储
6. GPU功能增强
7. 模块许可证
8. 云主机导出增强
9. 计算规格的物理机分配策略新增非强制/强制模式
10. VPC路由器配置DNS
11. 其它相关功能和优化
    - 新增多个操作场景进度条
    - 操作助手和帮助文档
    - 优化界面交互
    - 优化部分业务逻辑

2.3.2

2018/05/11主要更新：

1. 云资源池：

- 云主机根云盘/数据云盘容量在线扩展
- 通过FTP和SFTP方式在线添加镜像模板

2. 硬件设施：
   - 分布式存储Ceph以存储池（Pool）粒度显示容量使用情况
   - 识别物理机CPU架构，识别主流Intel和AMD处理器
   - 集群按照物理机CPU架构定义属性，为云主机提供丰富的CPU多媒体指令集，以及提升热迁移兼容性
   - 指定集群云主机热/冷迁移网络
3. 网络服务：
   - 负载均衡监听协议支持HTTPS，需绑定证书使用
   - 强化监听器功能
4. VMware vCenter接管：
   - vCenter云主机迁移、克隆
   - vCenter物理机维护模式
5. 平台运维：TOP5性能分析，支持对应项搜索排序
6. 平台管理：
   - 强化定时任务功能
   - 在管理界面上修改控制台代理地址
7. 大屏监控：解决登录会话超时失效
8. 混合云：对接大河云联SD-WAN服务，提供混合云高速链路
9. 超融合解决方案：
   - 管理节点云主机管理员密码重置
   - 管理节点云主机跨网段创建/启动，跨网络异地部署
   - 管理节点云主机部署/迁移至非超融合节点，适应更广泛场景
10. 其它相关功能和优化：
    - 新增多个操作场景进度条
    - 操作助手和帮助文档
    - 优化界面交互
    - 优化部分业务逻辑

2.3.1

2018/04/03主要更新：

1. 网络拓扑
2. 新版菜单导航、新版首页
3. ZWatch：全新监控报警系统
4. ZStack定制版ISO新增：基于CentOS 7.4深度定制版本
5. 亲和组
6. 增强vCenter接管功能：接管vCenter云盘、基于vCenter云路由网络提供网络服务
7. 一个云主机加载多个ISO
8. 多种策略创建云主机
9. 一个二层网络可用于创建多个三层网络
10. 操作日志/审计全新改版
11. HTTPS安全访问UI管理界面
12. 内部访问业务流量的负载均衡
13. 优化自定义UI
14. 多个场景新增进度条、操作助手和帮助文档，优化UI交互
15. 优化部分业务逻辑

2.3.0

2018/02/08主要更新：

1. 专有网络VPC
2. 混合云灾备（混合云版支持）
3. 大屏监控
4. 用户自定义UI
5. ImageStore类型镜像服务器支持Ceph类型主存储
6. 支持vSwitch
7. 支持vCenter资源同步
8. ESXi云主机支持扁平网络
9. 云主机更换操作系统
10. 跨NFS存储数据迁移
11. 虚拟IP支持QoS

12.支持AD认证

13.云主机自定义MAC地址

14.强化浏览器上传镜像功能

15.新增云盘镜像资源

16.数据云盘扩容

17.数据云盘规格支持QoS

18.停止NeverStop状态的云主机

19.开放云路由公网IP，并支持同一虚拟IP多网络服务复用

20.支持USB设备透传，强化外接设备透传功能

21.增加VDI SPICE流量优化选项

22.支持修改已设置的存储网络

23.支持设置VXLAN对普通账户的配额

24.支持ImageStore类型镜像服务器间的数据同步

25.管理节点数据库自动备份到远程服务器

26.多个场景新增进度条、操作助手和帮助文档，优化UI交互

27.优化部分业务逻辑

2.2

2017/10/16主要更新：

1. 公有网络创建云主机

2. 自定义DHCP模式

3. 新增系统网络

4. 云主机根云盘扩容

5. 浏览器添加镜像（目前支持ImageStore类型镜像服务器）

6. 管理节点高可用：多网络配置

7. 跨Ceph存储数据迁移

8. 增强Ceph存储功能

9. 增强VDI功能

10.LDAP自定义过滤规则

11.增强裸机管理

12.单集群支持多类型主存储（目前支持本地存储+NFS/SMP类型）

13.更换License支持本地上传

14.共享存储指定存储网络，增强云主机高可用

15.多个场景新增进度条、操作助手和帮助文档，优化UI交互

16.优化部分业务逻辑

2.1

2017/08/14主要更新：

1. VDI

2. 裸机管理

3. GPU透传

4. 智能报警

5. 集群挂载多个主存储

6. 新版定时器

7. 静态路由

8. User Data导入

9. 云路由加载多个公有网络

10.增量升级

11.优化部分业务逻辑

2 系列索引

ZStack 2.5.1 开发手册系列索引如下：

- 《ZStack 2.5.1 开发手册 云资源池》
- 《ZStack 2.5.1 开发手册 硬件设施》
- 《ZStack 2.5.1 开发手册 网络资源》
- 《ZStack 2.5.1 开发手册 网络服务》
- 《ZStack 2.5.1 开发手册 vCenter》
- 《ZStack 2.5.1 开发手册 平台运维》
- 《ZStack 2.5.1 开发手册 平台管理》
- 《ZStack 2.5.1 开发手册 设置》
- 《ZStack 2.5.1 开发手册 系统全局相关》

3 API规范概述

ZStack 2.5.1提供原生RESTful支持。您可以通过REST定义的架构设计原则和约束条件，并使用支持HTTP的编程语言进行开发。

3.1 HTTP方法 (HTTP Verbs)

当前API支持如下操作资源的方法：

方法名	描述
GET	获取资源信息。 • 所有的查询API以及读API均使用该方法。
POST	创建一个资源。
PUT	修改一个资源。 • 所有对资源的修改操作，以及类RPC调用的操作，例如启动虚拟机，均使用该方法。
DELETE	删除一个资源。

3.2 传参方式

URL、Query String、HTTP body三种方式均可用于传参。每种方式可以单独使用，也可以混合使用，具体使用哪种传参方式由具体API决定。

URL传参

当对某具体资源进行操作时，资源的UUID通过编码到URL的方式进行传参。

例如启动一个UUID为 *f97143d60f1042c9badd9a1336d3c105*的虚拟机，URL格式为：

```
zstack/v1/vm-instances/f97143d60f1042c9badd9a1336d3c105/actions
```

这里UUID编码到URL路径当中。

Query String传参

所有使用HTTP GET方法的API均使用Query String传参。

例如查询所有状态为Running的虚拟机， URL格式为：

```
zstack/v1/vm-instances?condition=state=Running
```

HTTP Body传参

当使用POST方法创建一个资源，或PUT方法修改一个资源时，除通过URL传参的部分外，剩余参数均通过HTTP Body传参。

例如在指定物理主机上启动一个虚拟机：

```
PUT zstack/v1/vm-instances/f97143d60f1042c9badd9a1336d3c105/actions

{
  "startVmInstance": {
        "hostUuid": "8aef7e3a53b34eedaa05027a919156d9"
    }
}
```

这里虚拟机的UUID通过URL传参，参数**hostUuid**则通过HTTP Body传递。

3.3 HTTP Headers

当前API使用如下自定义HTTP Headers：

Authorization

除了少数API外（例如登录API），使用ZStack API前都需要一个会话(session)，在调用API时通过Authorization HTTP Header传递会话UUID。该Header的格式为：

```
Authorization: OAuth 会话UUID
```

举例：

```
Authorization: OAuth 34cbfddd470a47d8bdb0727cd2182618
```

注: OAuth和会话UUID之间用空格分隔。

X-Job-UUID

对于异步API，可以通过X-Job-UUID HTTP Header来指定该API Job的UUID，例如：

```
X-Job-UUID: d825b1a26f4e474b8c59306081920ff2
```

如果未指定该HTTP Header，ZStack会自动为API Job生成一个UUID。

注:

X-Job-UUID必须为一个v4版本的UUID（即随机UUID）字符串去掉连接符-。ZStack会验证X-Job-UUID格式的合法性，并对非法的字符串返回一个400 Bad Request的错误。

X-Web-Hook

对于异步API，可以通过X-Web-Hook HTTP Header指定一个回调URL用于接收API 返回。通过使用回调URL的方法，调用者可以避免使用轮询去查询一个异步API的执行结果。举例：

```
X-Web-Hook: http://localhost:5000/api-callback
```

X-Job-Success

当使用了X-Web-Hook回调的方式获取异步API结果时，ZStack推送给回调URL的HTTP Post请求中会包含X-Job-Success HTTP Header指明该异步API的执行结果是成功还是失败。例如：

```
X-Job-Success: true
```

当值为*true*时执行成功，为*false*时执行失败。

3.4 HTTP返回码 (HTTP Status Code)

ZStack使用如下返回码：

返回码	意义
200	API执行成功。
202	API请求已被ZStack接受，用户需要通过轮询或Web Hook的方式获取API结果。该返回码只在调用异步API时出现。
400	API请求未包含必要的参数或包含了非法的参数。具体信息可以从HTTP Response Body获得。
404	URL不存在，通常是指定了错误的API URL。如果访问的URL是异步API返回的轮询地址，表示该轮询地址已经过期。
405	API调用使用了错误的HTTP方法，例如在创建一个资源的时候用了GET方法而不是POST方法。
500	ZStack RESTful终端遭遇了一个内部错误。
503	API所执行的操作引发了一个错误，例如资源不足无法创建虚拟机。错误的具体信息可以从HTTP Response Body。

3.5 API种类

ZStack的API分为同步API和异步API两种：

同步API

所有使用GET方法的API都是同步API，调用方收到的HTTP Response中直接包含了API的结果。例如：

```
GET zstack/v1/zones/f3fa7671894a40f6a73f5bfc7d90c126

{
   "inventory": {
      "uuid": "f3fa7671894a40f6a73f5bfc7d90c126",
      "name": "zone1",
      "description": "test",
      "state": "Enabled",
      "type": "zstack",
      "createDate": "Jan 6, 2017 3:51:16 AM",
      "lastOpDate": "Jan 6, 2017 3:51:16 AM"
   }
}
```

异步API

除了登录相关的API外，所有不使用GET方法的API都为异步API。用户调用一个异步API成功后会收到202返回码以及 Body中包含的一个轮询地址（location字段），用户需要周期性的GET该轮询地址以获得API的执行结果。例如：

```
Status Code: 202

Body:

{
   "location": "http://localhost:8989/v1/api-jobs/967a26b7431c49c0b1d50d709ef1aef3"
}
```

通常情况下GET一个轮询地址可以得到四种返回：

1. 202返回码表示该API仍在处理中，用户需要继续轮询。
2. 200返回码表示API执行成功，Body中包含API结果。
3. 503返回码表示API执行失败，Body中包含错误码。
4. 404返回码，则表示轮询地址已经过期，产生这种结果的原因可能是用户访问了一个错误的轮询地址，或者太久没有访问该轮询地址（例如超过2天没有访问），该轮询地址已经被删除。

异步API也可以用Web Hook的方式获得结果，具体方法见后面章节。

3.6 API操作

跟所有的RESTful API类似，绝大多数ZStack API执行的是CRUD（Create, Read, Update, Delete）操作，以及类RPC操作。

创建资源

所有资源的创建都使用POST方法，参数通过HTTP Body传递，例如创建一个虚拟机：

```
POST zstack/v1/vm-instances

Authorization： OAuth 0c234e29a2ad4ff4b0d97d4f3b47c6cf

{
    "params": {
        "l3NetworkUuids": ["37a701c7fe4a40758da15593aedd8aff"],
        "defaultL3NetworkUuid": "37a701c7fe4a40758da15593aedd8aff",
        "dataDiskOfferingUuids": [],
        "name": "TestVm",
        "description": "Test",
        "systemTags": [],
        "instanceOfferingUuid": "dd53f94b58924510b0122e40799a4114",
        "type": "UserVm",
        "imageUuid": "cc7b56780879409f98c1f992b75a12b0"
    }
}
```

查询资源

资源的查询使用GET方法，查询条件通过Query String传参，例如查询集群*cluster1*中名字**不等于** *web-vm*的虚拟机：

```
GET zstack/v1/vm-instances?condition=name!=web-vm&condition=cluster.name=cluster1

Authorization： OAuth 0c234e29a2ad4ff4b0d97d4f3b47c6cf
```

如果已知资源的UUID，要直接获取该资源的信息，直接使用GET方法不加任何查询条件，例如：

```
GET zstack/v1/vm-instances/56f0fd314a2647ffb4f9565f6d05858e

Authorization： OAuth 0c234e29a2ad4ff4b0d97d4f3b47c6cf
```

返回UUID为*56f0fd314a2647ffb4f9565f6d05858e*虚拟机的信息。

删除资源

删除资源使用DELETE方法，被删除资源的UUID编码在URL中，例如：

```
DELETE zstack/v1/vm-instances/56f0fd314a2647ffb4f9565f6d05858e

Authorization： OAuth 0c234e29a2ad4ff4b0d97d4f3b47c6cf
```

删除UUID为*56f0fd314a2647ffb4f9565f6d05858e*的虚拟机。

修改资源与类RPC操作

但由于IaaS本身业务的性质，一部分操作更类似于RPC（远程调用）而非CRUD操作，例如启动虚拟机。根据RESTful API的一些最佳实践，ZStack将这些操作都归为资源的actions子资源，例如启动虚拟机、 停止虚拟机都是对虚拟机的actions子资源进行操作。举个例子：

启动虚拟机：

```
PUT zstack/v1/vm-instances/d46841bd4ebd47f8bf0bed85c3bdf0db/actions

{
   "startVmInstance": {}
}
```

停止虚拟机：

```
PUT zstack/v1/vm-instances/d46841bd4ebd47f8bf0bed85c3bdf0db/actions

{
   "stopVmInstance": {}
}
```

在上面的例子中，两个操作都访问的是相同的URL*/v1/vminstances/d46841bd4ebd47f8bf0bed85c3bdf0db/actions*， 具体的操作类型由包含在Body中的字段名表示，例如stopVmInstance，如果该API包含额外参数，则包含在操作字段名对应的map中。

资源操作的具体字段名和例子参考每个API的详细文档。

3.7 基本流程示例

在下例中，我们会创建一个Zone，以展示API使用的基本流程：

登录

使用API的第一步是登录以获取一个Session UUID，以供后续API调用使用。

```
PUT zstack/v1/accounts/login

body:

{
   "logInByAccount": {
      "password": "b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c
706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86",
      "accountName": "admin"
   }
}
```

这里的密码是用sha512哈希后的结果。

API返回如下：

```
status code: 200

body:

{
   "inventory": {
      "uuid": "00d038b699b74e76a01705918d48d939",
      "accountUuid": "36c27e8ff05c4780bf6d2fa65700f22e",
      "userUuid": "36c27e8ff05c4780bf6d2fa65700f22e",
      "expiredDate": "Jan 1, 2017 11:31:06 AM",
      "createDate": "Jan 1, 2017 9:31:06 AM"
   }
}
```

返回内容中包含账户UUID等其他字段，我们需要的session UUID包含在字段uuid中：*00d038b699b74e76a01705918d48d939*。

创建Zone

```
POST zstack/v1/zones

headers:

Authorization: OAuth 00d038b699b74e76a01705918d48d939

body:

{
   "params": {
      "name": "Zone1",
      "description": "Test"
   }
}
```

由于创建Zone操作是一个异步API，API返回不是直接的结果，而是一个轮询地址：

```
status code: 202

body:

{
   "location": "http://localhost:8989/v1/api-jobs/d0345d3ddcae485f8170572b15a2b581"
}
```

用户需要周期性的轮询API结果：

```
GET http://localhost:8989/v1/api-jobs/d0345d3ddcae485f8170572b15a2b581
```

```
Authorization: OAuth 00d038b699b74e76a01705918d48d939
```

如果API还未执行完成，上述GET操作得到的仍然是202返回码和上述轮询地址。当操作完成时，得到的结果如下：

```
status code: 200

body:

{
    "inventory": {
        "uuid": "f52fe55b64094ceb99b3893a238c4931",
        "name": "Zone1",
        "description": "Test",
        "state": "Enabled",
        "type": "zstack",
        "createDate": "Jan 1, 2017 9:31:07 AM",
        "lastOpDate": "Jan 1, 2017 9:31:07 AM"
    }
}
```

查询Zone

要获取创建的Zone的信息，可以用GET查询：

```
GET zstack/v1/zones/f52fe55b64094ceb99b3893a238c4931

Authorization: OAuth 00d038b699b74e76a01705918d48d939
```

返回：

```
status code: 200

body:

{
    "inventory": {
        "uuid": "f52fe55b64094ceb99b3893a238c4931",
        "name": "Zone1",
        "description": "Test",
        "state": "Enabled",
        "type": "zstack",
        "createDate": "Jan 1, 2017 9:31:07 AM",
        "lastOpDate": "Jan 1, 2017 9:31:07 AM"
    }
```

```
}
```

登出

当所有API调用完毕，我们需要对已登录的session进行登出操作：

```
DELETE zstack/v1/accounts/sessions/00d038b699b74e76a01705918d48d939
```

返回

```
status code: 200
```

3.8 Webhook

对于异步API使用轮询的方式查询操作结果是一种低效的方式，为此ZStack提供Webhook的方式主动推送异步API结果给调用者。

要使用Webhook功能，调用者只需在HTTP Headers中指定X-Job-UUID和X-Web-Hook即可。以上面创建Zone为例，使用Webhook的API版本为：

```
POST zstack/v1/zones

headers:

Authorization: OAuth 00d038b699b74e76a01705918d48d939
X-Job-UUID: d0345d3ddcae485f8170572b15a2b581
X-Web-Hook: http://127.0.0.1:8989/rest-webhook

body:

{
    "params": {
        "name": "Zone1",
        "description": "Test"
    }
}
```

API返回仍然是202返回码和一个轮询地址，但调用者无需再轮询。API执行成功后，结果会被推送到

http://127.0.0.1:8989/rest-webhook

```
POST http://127.0.0.1:8989/rest-webhook

headers:

X-Job-Success: true
X-Job-UUID: d0345d3ddcae485f8170572b15a2b581

body:

{
    "inventory": {
        "uuid": "f52fe55b64094ceb99b3893a238c4931",
```

```
        "name": "Zone1",
        "description": "Test",
        "state": "Enabled",
        "type": "zstack",
        "createDate": "Jan 1, 2017 9:31:07 AM",
        "lastOpDate": "Jan 1, 2017 9:31:07 AM"
    }
}
```

推送的结果之中，X-Job-Success指明了API执行成功与否，X-Job-UUID包含值跟API调用时的X-Job-UUID相同，调用者可以对应结果和API。

3.9 查询API

用户可以用GET方法对一个资源进行查询，并且可以像MySQL一样指定多个查询条件、排序方式、选择字段、以及进行跨表查询等等。

支持超过400万个单项查询条件，以及400万阶乘的组合查询条件。

单表查询

例如：

```
GET zstack/v1/vm-instances?q=name=vm1
```

查询名字为*vm1*的虚拟机。

例如：

```
GET zstack/v1/vm-instances?q=name=vm1&q=state=Running
```

查询名字为*vm1*并且状态为*Running*的虚拟机。

这两个例子都是对虚拟机资源本身查询，反应到数据库层面还属于单表查询。

跨表查询

我们可以通过.进行跨表查询。

例如：

```
GET zstack/v1/vm-instances?q=vmNics.ip=192.168.10.100
```

查询IP地址为*192.168.10.100*的虚拟机，这里对虚拟机和网卡两张表进行了跨表查询。

例如：

```
GET zstack/v1/vm-instances?q=host.managementIp=10.10.20.3
```

查询IP为*10.10.20.3*上运行的所有虚拟机。这里对虚拟机和物理机两张表进行了跨表查询。

所有资源的查询API都支持下列参数

名字	类型	位置	描述	可选值	起始版本
q (可选)	List	query	见查询条件。省略该字段将返回所有记录，返回记录数的上限受限于limit字段		0.6
limit (可选)	Integer	query	最多返回的记录数，类似MySQL的limit，默认值1000		0.6
start (可选)	Integer	query	起始查询记录位置，类似MySQL的offset。跟limit配合使用可以实现分页		0.6
count (可选)	Boolean	query	计数查询，相当于MySQL中的count()函数。当设置成true时，API只返回的是满足查询条件的记录数		0.6
groupBy (可选)	String	query	以字段分组，相当于MySQL中的group by关键字。例如groupBy=type		1.9
replyWithCount (可选)	Boolean	query	见分页查询		0.6
sort (可选)	String	query	以字段排序，等同于MySQL中的sort by关键字。必须跟+或者-配合使用，+表示升序，-表示降序，后面跟排序字段名，例如： • sort=+key，根据key进行升序排序 • sort=-key，根据 key进行降序排序	`+`字段名，`-`字段名	0.6
fields (可选)	List	query	指定返回的字段，等同于MySQL中的select字段功能。例如fields=name,uuid，则只返回满足条件记录的name和uuid字段		0.6

查询条件

的查询条件类似于MySQL数据库。

例如：

```
uuid=bfa67f956afb430890aa49db14b85153
totalCapacity>2000
```

```
vmInstanceUuid not null
```

注:

- **字段名、查询操作符、匹配值三者之间不能有任何空格。**
- 例如*uuid = 25506342d1384c07b7342373a57475b9*就是一个错误的查询条件，必须写为*uuid=25506342d1384c07b7342373a57475b9*。

多个查询条件之间是与关系。总共支持10个查询操作符：

1. **=**: 等于，例如：

```
vmInstanceUuid=c4981689088b40f98d2ade2548c323da
```

2. **!=**: 不等于，例如：

```
vmInstanceUuid!=c4981689088b40f98d2ade2548c323da
```

3. **>**: 大于
4. **<**: 小于
5. **>=**: 大于等于
6. **<=**: 小于等于
7. **?=**: **in**操作符，测试字段值是否在一个集合。集合中的值以,分隔。例如测试*uuid*是否属于某个集合：

```
uuid?=25506342d1384c07b7342373a57475b9,bc58d68090ac42358c0cb0fe72e3287f
```

8. **!?=**: **not int**操作符，测试字段值是否**不属于**一个集合。集合中的值以,分隔，例如测试*name*是否不等于VM1和VM2：

```
name!?=VM1,VM2
```

9. **~=**: 字符串模糊匹配，相当于MySQL中的**like**操作。使用**%**匹配一个或多个字符，使用_匹配一个字符。 例如查询一个名字是以*IntelCore*开头的：

```
name~=IntelCore%
```

10. 或者查询一个名字是以*IntelCore*开头，以*7*结尾，中间模糊匹配一个字符：

```
name~=IntelCore_7
```

这样名字是*IntelCoreI7*，*IntelCoreM7*的记录都会匹配上。

11. **!~=**: 模糊匹配非操作。查询一个字段不能模糊匹配到某个字符串，匹配条件与**~=**相同。

12.**is null**: 字段为null：

```
name=null
```

13.**not null**: 字段不为null：

```
name!=null
```

分页查询

start、**limit**、**replyWithCount**三个字段可以配合使用实现分页查询。

- **start**指定起始查询位置。
- **limit**指定查询返回的最大记录数。
- **replyWithCount**被设置成true后，查询返回中会包含满足查询条件的记录总数，跟**start**值比较就可以得知还需几次分页。

例如：

总共有1000记录满足查询条件，使用如下组合：

```
start=0 limit=100 replyWithCount=true
```

则API返回将包含头100条记录，以及total字段等于1000，表示总共满足条件的记录为1000。

获取资源可查询字段

由于支持的查询条件数非常巨大，我们无法在文档中枚举所有的查询条件。

用户可以使用命令行工具*zstack-cli*的自动补全功能来查看一个资源可查询的字段以及可跨表查询的字段。

- 以查询虚拟机为例，在*zstack-cli*里输入QueryVmInstance并按Tab键补全，可以看到提示页面：

```
- >>>QueryVmInstance
[Query Conditions:]
allVolumes.          cluster.             host.                image.               instanceOffering.
 rootVolume.
vmNics.              zone.

__systemTag__=       __userTag__=         allocatorStrategy=   clusterUuid=
 cpuNum=             cpuSpeed=
createDate=          defaultL3NetworkUuid=  description=        groupBy=             hostUuid
=            hypervisorType=
imageUuid=           instanceOfferingUuid=  lastHostUuid=       lastOpDate=
 memorySize=         name=
platform=            rootVolumeUuid=      state=               type=
uuid=                zoneUuid=

[Parameters:]
```

```
count=              fields=             limit=              replyWithCount=     sortBy=
 sortDirection=
start=              timeout=
```

- 这里中间行：

```
__systemTag__=      __userTag__=        allocatorStrategy=  clusterUuid=
 cpuNum=            cpuSpeed=
createDate=         defaultL3NetworkUuid=  description=     groupBy=            hostUuid
=           hypervisorType=
imageUuid=          instanceOfferingUuid=  lastHostUuid=     lastOpDate=
 memorySize=        name=
platform=           rootVolumeUuid=     state=              type=
uuid=               zoneUuid=
```

除**__systemTag__**和**__userTag__**两个特殊查询条件外，其余均为虚拟机表的原生字段，用户可以在API的查询条件里面指定它们，并且可以在**fields**参数中指定这些字段来过滤其它不希望API返回的字段。

例如：

```
GET zstack/v1/vm-instances?q=cpuNum>5
```

返回CPU数量多于5的虚拟机。

```
GET zstack/v1/vm-instances?q=hypervisorType=KVM&fields=uuid&fields=name
```

返回虚拟化类型为KVM的虚拟机，由于在**fields**指定了uuid和name两个字段，API返回中只会包含虚拟机的name和uuid。

注:

只有资源的原生字段可以被**fields**选取，**__userTag__**、**__systemTag__**以及下面讲到的跨表字段均不能出现在**fields**参数中。

- 提示的第一行：

```
allVolumes.         cluster.            host.               image.              instanceOffering.
 rootVolume.
vmNics.             zone.
```

指明了虚拟机资源可以跟哪些资源做跨表查询，例如：**allVolumes**代表云盘，**cluster**代表集群，**vmNics**代表网卡等。

如需查看这些资源的具体字段，只需输入资源名加.号，并按Tab键补全。

例如：

```
- >>>QueryVmInstance vmNics.
```

```
[Query Conditions:]
vmNics.eip.                vmNics.l3Network.          vmNics.loadBalancerListener.  vmNics
.portForwarding.       vmNics.securityGroup.
vmNics.vmInstance.

vmNics.__systemTag__=        vmNics.__userTag__=          vmNics.createDate=
 vmNics.deviceId=           vmNics.gateway=
vmNics.ip=                 vmNics.l3NetworkUuid=        vmNics.lastOpDate=          vmNics
.mac=                vmNics.metaData=
vmNics.netmask=            vmNics.uuid=               vmNics.vmInstanceUuid=
```

这里我们输入了资源**vmNics**并用**.**号表示我们要做一个跨表查询，Tab键为我们补全了**vmNics**资源的原生字段以及可跨表查询的其它资源。

- 例如**vmNics.ip**表示网卡的原生字段**ip**：

  ```
  GET zstack/v1/vm-instances?q=vmNics.ip=192.168.0.100
  ```

  进行了一个跨表查询，条件是网卡表的**ip**字段，返回的结果是**ip**为*192.168.0.100*的虚拟机。

- 网卡资源同样可以跟其它资源进行跨表查询，例如**vmNics.eip**。

  将网卡表和EIP表进行跨表：

  ```
  GET zstack/v1/vm-instances?q=vmNics.eip.ip=192.168.0.100
  ```

  进行了跨3表查询，返回的是EIP为*192.168.0.100*的虚拟机。

- 通过资源间连续跨表，一个资源几乎跟系统中多个有逻辑关系的资源进行跨表，例如：

```
- >>>QueryVmInstance zone.cluster.l2Network.l3Network.
[Query Conditions:]
zone.cluster.l2Network.l3Network.ipRanges.        zone.cluster.l2Network.l3Network.l2Netwo
rk.       zone.cluster.l2Network.l3Network.networkServices.
zone.cluster.l2Network.l3Network.serviceProvider.  zone.cluster.l2Network.l3Network.vmNic.
          zone.cluster.l2Network.l3Network.zone.


zone.cluster.l2Network.l3Network.__systemTag__=    zone.cluster.l2Network.l3Network
.__userTag__=      zone.cluster.l2Network.l3Network.createDate=
zone.cluster.l2Network.l3Network.description=      zone.cluster.l2Network.l3Network
.dnsDomain=        zone.cluster.l2Network.l3Network.l2NetworkUuid=
zone.cluster.l2Network.l3Network.lastOpDate=       zone.cluster.l2Network.l3Network.name=
           zone.cluster.l2Network.l3Network.state=
zone.cluster.l2Network.l3Network.system=           zone.cluster.l2Network.l3Network.type=
       zone.cluster.l2Network.l3Network.uuid=
zone.cluster.l2Network.l3Network.zoneUuid=
```

分别跟**zone**, **cluster**, **l2Network**, **l3Network**多个资源进行跨表。

注:

- 由于一个资源的逻辑关系存在环路，因此会存在环路路径。例如：以云主机为查询主体可以跟网卡进行跨表查询（例如：`QueryVmInstance vmNics.`），同时以网卡为主

体也可以跟云主机进行跨表查询（例如：QueryVmNic vmInstance.），这样就会存在环路路径。

- 使用中应该避免环路跨表查询。例如 QueryVmInstance vmNics.vmInstance.name =vm1通过跨表查询了name=vm1的云主机，它的实际效果跟QueryVmInstance name =vm1完全等同。这里的跨表是无意义的，只会生产复杂的SQL语句导致低效的数据库查询。

- **__systemTag__**和**__userTag__**是两个特殊的查询条件，允许用户通过tag查询资源。

  例如：

  ```
  QueryVmInstance __systemTag__=staticIp:10.10.1.20
  ```

  查询具有*staticIp:10.10.1.20*这个tag的虚拟机。

4 vCenter

4.1 vCenter相关接口

4.1.1 添加vCenter资源(AddVCenter)

API请求

URLs

```
POST zstack/v1/vcenters
```

Headers

```
Authorization: OAuth the-session-uuid
```

Body

```
{
  "params": {
    "username": "admin@vsphere-local.net",
    "password": "test-pass",
    "zoneUuid": "25b9e68b1cc43836a79f1a35e3108880",
    "name": "vc1",
    "https": true,
    "domainName": "vsphere-local.net"
  },
  "systemTags": [],
  "userTags": []
}
```

注: 上述示例中**systemTags**、**userTags**字段可以省略。列出是为了表示body中可以包含这两个字段。

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth b86c9016b4f24953a9edefb53ca0678c" \
-X POST -d '{"params":{"username":"admin@vsphere-local.net","password":"test-pass
","zoneUuid":"25b9e68b1cc43836a79f1a35e3108880","name":"vc1","https":true,"domainName
":"vsphere-local.net"}}' \
http://localhost:8080/zstack/v1/vcenters
```

参数列表

名字	类型	位置	描述	可选值	起始版本
username	String	body(包含在**params**结构中)	登录 vCenter 用户名		0.6
password	String	body(包含在**params**结构中)	登录 vCenter 用户密码		0.6
zoneUuid	String	body(包含在**params**结构中)	vCenter 资源将要添加至的目标区域UUID		0.6
name	String	body(包含在**params**结构中)	vCenter 资源名称		0.6
https (可选)	boolean	body(包含在**params**结构中)	是否启用 HTTPS 登录（默认开启）		0.6
domainName	String	body(包含在**params**结构中)	vCenter 域名		0.6
description (可选)	String	body(包含在**params**结构中)	vCenter 资源的详细描述		0.6
resourceUuid (可选)	String	body(包含在**params**结构中)	vCenter 资源UUID。若指定，云主机会使用该字段值作为UUID。		0.6
systemTags (可选)	List	body	系统标签		0.6
userTags (可选)	List	body	用户标签		0.6
port (可选)	Integer	body(包含在params结构中)			0.6

API返回

返回示例

```
{
  "inventory": {
    "uuid": "1497ae9206ea3f82bb13c34c307299b9",
    "name": "vc1",
    "domainName": "vsphere-local.net",
    "userName": "admin@vsphere-local.net",
    "https": true,
    "state": "Enabled",
    "status": "Connected"
  }
}
```

名字	类型	描述	起始版本
error	ErrorCode	错误码，若不为null，则表示操作失败, 操作成功时该字段为null。 详情参考*error*	0.6
inventory	VCenterInventory	详情参考*inventory*	0.6

#error

名字	类型	描述	起始版本
code	String	错误码号，错误的全局唯一标识，例如SYS.1000, HOST.1001	0.6
description	String	错误的概要描述	0.6
details	String	错误的详细信息	0.6
elaboration	String	保留字段，默认为null	0.6
opaque	LinkedHashMap	保留字段，默认为null	0.6
cause	ErrorCode	根错误，引发当前错误的源错误，若无原错误，该字段为null	0.6

#inventory

名字	类型	描述	起始版本
uuid	String	资源的UUID，唯一标示该资源	0.6

名字	类型	描述	起始版本
name	String	资源名称	0.6
description	String	资源的详细描述	0.6
domainName	String		0.6
port	Integer		0.6
userName	String		0.6
zoneUuid	String	区域UUID	0.6
https	Boolean		0.6
state	String		0.6
status	String		0.6
createDate	Timestamp	创建时间	0.6
lastOpDate	Timestamp	最后一次修改时间	0.6

SDK示例

Java SDK

```
AddVCenterAction action = new AddVCenterAction();
action.username = "admin@vsphere-local.net";
action.password = "test-pass";
action.zoneUuid = "25b9e68b1cc43836a79f1a35e3108880";
action.name = "vc1";
action.https = true;
action.domainName = "vsphere-local.net";
action.sessionId = "b86c9016b4f24953a9edefb53ca0678c";
AddVCenterAction.Result res = action.call();
```

Python SDK

```
AddVCenterAction action = AddVCenterAction()
action.username = "admin@vsphere-local.net"
action.password = "test-pass"
action.zoneUuid = "25b9e68b1cc43836a79f1a35e3108880"
action.name = "vc1"
action.https = true
action.domainName = "vsphere-local.net"
action.sessionId = "b86c9016b4f24953a9edefb53ca0678c"
```

```
AddVCenterAction.Result res = action.call()
```

4.1.2 查询vCenter资源(QueryVCenter)

API请求

URLs

```
GET zstack/v1/vcenters
GET zstack/v1/vcenters/{uuid}
```

Headers

```
Authorization: OAuth the-session-uuid
```

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth e46c3ba50f324bcbabda6c40d11d1e33" \
-X GET http://localhost:8080/zstack/v1/vcenters?q=uuid=7e580c2f8b6c4e31aff447c45c72c3bb
```

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth d0552ed3feb243a1b87fcff421388964" \
-X GET http://localhost:8080/zstack/v1/vcenters/bb39998d05004cd9aa5bf816cc308ed1
```

可查询字段

运行*zstack-cli*命令行工具，输入QueryVCenter并按Tab键查看所有可查询字段以及可跨表查询的资源名。

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
    "error": {
        "code": "SYS.1001",
        "description": "A message or a operation timeout",
        "details": "Create VM on KVM timeout after 300s"
    }
}
```

SDK示例

Java SDK

```
QueryVCenterAction action = new QueryVCenterAction();
action.conditions = asList("uuid=70118e1db5084255a0e44c7af24d57b7");
action.sessionId = "10dbde6695b74150878bec7cbe97eda5";
```

```
QueryVCenterAction.Result res = action.call();
```

Python SDK

```
QueryVCenterAction action = QueryVCenterAction()
action.conditions = ["uuid=4e5f668c85be4bb4a1b5990611ad806b"]
action.sessionId = "708e5e17b29a4527a8480faadd49fdbc"
QueryVCenterAction.Result res = action.call()
```

4.1.3 删除vCenter资源(DeleteVCenter)

API请求

URLs

```
DELETE zstack/v1/vcenters/{uuid}
```

Headers

```
Authorization: OAuth the-session-uuid
```

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth cf9047b652144c11b62e7176eef5540b" \
-X DELETE http://localhost:8080/zstack/v1/vcenters/cd00a732974e4ca99886338bac41d511?
```

参数列表

名字	类型	位置	描述	可选值	起始版本
uuid	String	url	vCenter 资源的UUID，唯一标示该资源		0.6
deleteMode (可选)	String	body	删除模式(Permissive 或者 Enforcing，默认 Permissive)		0.6
systemTags (可选)	List	body	系统标签		0.6
userTags (可选)	List	body	用户标签		0.6

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
    "error": {
        "code": "SYS.1001",
        "description": "A message or a operation timeout",
        "details": "Create VM on KVM timeout after 300s"
    }
}
```

SDK示例

Java SDK

```
DeleteVCenterAction action = new DeleteVCenterAction();
action.uuid = "17ab65db07b04666ad8eef6158e7532f";
action.deleteMode = "Permissive";
action.sessionId = "e8e3599ce54c41a4b5ff3322d2479844";
DeleteVCenterAction.Result res = action.call();
```

Python SDK

```
DeleteVCenterAction action = DeleteVCenterAction()
action.uuid = "f516dab538a64c5a9f0f2a55141ae439"
action.deleteMode = "Permissive"
action.sessionId = "aca4acf373b04a2fb1ad1112f540b1b3"
DeleteVCenterAction.Result res = action.call()
```

4.1.4 更新vCenter资源(UpdateVCenter)

API请求

URLs

```
PUT zstack/v1/vcenters/{uuid}/actions
```

Headers

```
Authorization: OAuth the-session-uuid
```

Body

```
{
  "updateVCenter": {
    "name": "Test-VCenter",
    "description": "Test-Description",
    "username": "Test-Username",
    "password": "Test-Password",
    "domainName": "Test-DomainName",
    "port": 443.0
  },
  "systemTags": [],
```

```
  "userTags": []
}
```

注: 上述示例中**systemTags**、**userTags**字段可以省略。列出是为了表示body中可以包含这两个字段。

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth b86c9016b4f24953a9edefb53ca0678c" \
-X PUT -d '{"updateVCenter":{"name":"Test-VCenter","description":"Test-Description","usernam
e":"Test-Username","password":"Test-Password","domainName":"Test-DomainName","port
":443.0}}' \
http://localhost:8080/zstack/v1/vcenters/08a6059f054a3866abf6302b6461e340/actions
```

参数列表

名字	类型	位置	描述	可选值	起始版本
uuid	String	url	资源的UUID，唯一标示该资源		0.6
name (可选)	String	body(包含在**updateVCenter**结构中)	资源名称		0.6
description (可选)	String	body(包含在**updateVCenter**结构中)	资源的详细描述		0.6
username (可选)	String	body(包含在**updateVCenter**结构中)			0.6
password (可选)	String	body(包含在**updateVCenter**结构中)			0.6
domainName (可选)	String	body(包含在**updateVCenter**结构中)			0.6
port (可选)	Integer	body(包含在**updateVCenter**结构中)			0.6
systemTags (可选)	List	body			0.6

名字	类型	位置	描述	可选值	起始版本
userTags (可选)	List	body			0.6

API返回

返回示例

```
{
  "inventory": {
    "uuid": "6c42846814243c408a225e3738a2440f",
    "name": "Test-VCenter",
    "description": "Test-Description",
    "domainName": "Test-DomainName",
    "userName": "Test-Username",
    "password": "Test-Password"
  }
}
```

名字	类型	描述	起始版本
error	ErrorCode	错误码，若不为null，则表示操作失败, 操作成功时该字段为null。 详情参考error	0.6
inventory	VCenterInventory	详情参考inventory	0.6

#error

名字	类型	描述	起始版本
code	String	错误码号，错误的全局唯一标识，例如SYS.1000, HOST.1001	0.6
description	String	错误的概要描述	0.6
details	String	错误的详细信息	0.6
elaboration	String	保留字段，默认为null	0.6
opaque	LinkedHashMap	保留字段，默认为null	0.6
cause	ErrorCode	根错误，引发当前错误的源错误，若无原错误，该字段为null	0.6

#inventory

名字	类型	描述	起始版本
uuid	String	资源的UUID，唯一标示该资源	0.6
name	String	资源名称	0.6
description	String	资源的详细描述	0.6
domainName	String		0.6
port	Integer		0.6
userName	String		0.6
zoneUuid	String	区域UUID	0.6
https	Boolean		0.6
state	String		0.6
status	String		0.6
createDate	Timestamp	创建时间	0.6
lastOpDate	Timestamp	最后一次修改时间	0.6

SDK示例

Java SDK

```
UpdateVCenterAction action = new UpdateVCenterAction();
action.uuid = "08a6059f054a3866abf6302b6461e340";
action.name = "Test-VCenter";
action.description = "Test-Description";
action.username = "Test-Username";
action.password = "Test-Password";
action.domainName = "Test-DomainName";
action.port = 443.0;
action.sessionId = "b86c9016b4f24953a9edefb53ca0678c";
UpdateVCenterAction.Result res = action.call();
```

Python SDK

```
UpdateVCenterAction action = UpdateVCenterAction()
action.uuid = "08a6059f054a3866abf6302b6461e340"
action.name = "Test-VCenter"
action.description = "Test-Description"
action.username = "Test-Username"
action.password = "Test-Password"
action.domainName = "Test-DomainName"
action.port = 443.0
action.sessionId = "b86c9016b4f24953a9edefb53ca0678c"
```

```
UpdateVCenterAction.Result res = action.call()
```

4.1.5 同步vCenter资源(SyncVCenter)

API请求

URLs

```
PUT zstack/v1/vcenters/{uuid}/actions
```

Headers

```
Authorization: OAuth the-session-uuid
```

Body

```
{
  "syncVCenter": {},
  "systemTags": [],
  "userTags": []
}
```

注: 上述示例中**systemTags**、**userTags**字段可以省略。列出是为了表示body中可以包含这两个字段。

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth b86c9016b4f24953a9edefb53ca0678c" \
-X PUT -d '{"syncVCenter":{}}' \
http://localhost:8080/zstack/v1/vcenters/ada8b8f96b3033bc9420c13bba7d51b9/actions
```

参数列表

名字	类型	位置	描述	可选值	起始版本
uuid	String	url	vCenter资源的UUID，唯一标示该资源		2.2
systemTags (可选)	List	body	系统标签		2.2
userTags (可选)	List	body	用户标签		2.2

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
    "error": {
        "code": "SYS.1001",
        "description": "A message or a operation timeout",
        "details": "Create VM on KVM timeout after 300s"
    }
}
```

SDK示例

Java SDK

```
SyncVCenterAction action = new SyncVCenterAction();
action.uuid = "ada8b8f96b3033bc9420c13bba7d51b9";
action.sessionId = "b86c9016b4f24953a9edefb53ca0678c";
SyncVCenterAction.Result res = action.call();
```

Python SDK

```
SyncVCenterAction action = SyncVCenterAction()
action.uuid = "ada8b8f96b3033bc9420c13bba7d51b9"
action.sessionId = "b86c9016b4f24953a9edefb53ca0678c"
SyncVCenterAction.Result res = action.call()
```

4.1.6 查询vCenter下记录的数据中心(QueryVCenterDatacenter)

API请求

URLs

```
GET zstack/v1/vcenters/datacenters
GET zstack/v1/vcenters/datacenters/{uuid}
```

Headers

```
Authorization: OAuth the-session-uuid
```

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth 06e9b6df90ce4d13a1b2aabbc1b9070c" \
-X GET http://localhost:8080/zstack/v1/vcenters/datacenters?q=uuid=eec5497cdc2a4f04ba5a3e497803ff69
```

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth 160f1e9d49f148b995aee4023e2bc769" \
```

```
-X GET http://localhost:8080/zstack/v1/vcenters/datacenters/9e52bc6c8c214456a9ee7c11e6375709
```

可查询字段

运行zstack-cli命令行工具，输入QueryVCenterDatacenter并按Tab键查看所有可查询字段以及可跨表查询的资源名。

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
"error": {
    "code": "SYS.1001",
    "description": "A message or a operation timeout",
    "details": "Create VM on KVM timeout after 300s"
}
}
```

SDK示例

Java SDK

```
QueryVCenterDatacenterAction action = new QueryVCenterDatacenterAction();
action.conditions = asList("uuid=60f406ebd44c4362911a5e7905bdac62");
action.sessionId = "d751323ed815434c9c8897eebf0bc6f0";
QueryVCenterDatacenterAction.Result res = action.call();
```

Python SDK

```
QueryVCenterDatacenterAction action = QueryVCenterDatacenterAction()
action.conditions = ["uuid=83936d668acd4c53b70596dd9cf647ac"]
action.sessionId = "e1cf6dfc43354570a9ae9ee8694c2f1e"
QueryVCenterDatacenterAction.Result res = action.call()
```

4.1.7 查询vCenter集群(QueryVCenterCluster)

API请求

URLs

```
GET zstack/v1/vcenters/clusters
GET zstack/v1/vcenters/clusters/{uuid}
```

Headers

```
Authorization: OAuth the-session-uuid
```

Curl示例

```
curl -H "Content-Type: application/json" \
```

```
-H "Authorization: OAuth b4ad190088e14bb89ad9594fd311a9b9" \
-X GET http://localhost:8080/zstack/v1/vcenters/clusters?q=uuid=82019f6ef752427480de4
812339149bd
```

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth e2545e88cc674f52bb6d9dd3954e8ea7" \
-X GET http://localhost:8080/zstack/v1/vcenters/clusters/ccf3e1c048d84ebbad90802a7bd0907c
```

可查询字段

运行zstack-cli命令行工具，输入QueryVCenterCluster并按Tab键查看所有可查询字段以及可跨表查询的资源名。

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
"error": {
    "code": "SYS.1001",
    "description": "A message or a operation timeout",
    "details": "Create VM on KVM timeout after 300s"
}
}
```

SDK示例

Java SDK

```
QueryVCenterClusterAction action = new QueryVCenterClusterAction();
action.conditions = asList("uuid=1197f1e666b24da9b96ca15b76d23029");
action.sessionId = "833bf9348c13481792af75ade4d2d0fb";
QueryVCenterClusterAction.Result res = action.call();
```

Python SDK

```
QueryVCenterClusterAction action = QueryVCenterClusterAction()
action.conditions = ["uuid=99e19ea5de35490f8f78f2b2c54e867b"]
action.sessionId = "bbfbeaea067840879cbc285146514152"
QueryVCenterClusterAction.Result res = action.call()
```

4.1.8 查询vCenter主存储(QueryVCenterPrimaryStorage)

API请求

URLs

```
GET zstack/v1/vcenters/primary-storage
```

```
GET zstack/v1/vcenters/primary-storage/{uuid}
```

Headers

```
Authorization: OAuth the-session-uuid
```

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth 0d623b631fd7402d83b9ff2b25bdd831" \
-X GET http://localhost:8080/zstack/v1/vcenters/primary-storage?q=uuid=981850c9581d4f
fb9df096112f902031
```

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth c9d5f42a19d745c6928da0cd5325e1d6" \
-X GET http://localhost:8080/zstack/v1/vcenters/primary-storage/d51815cfb0974e168f185
78618474f75
```

可查询字段

运行zstack-cli命令行工具，输入QueryVCenterPrimaryStorage并按Tab键查看所有可查询字段以及可跨表查询的资源名。

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
"error": {
    "code": "SYS.1001",
    "description": "A message or a operation timeout",
    "details": "Create VM on KVM timeout after 300s"
}
}
```

SDK示例

Java SDK

```
QueryVCenterPrimaryStorageAction action = new QueryVCenterPrimaryStorageAction();
action.conditions = asList("uuid=252afc9b92b34337adda34baa66f41c4");
action.sessionId = "0574d9e353cb434cb8873871d6c312ba";
QueryVCenterPrimaryStorageAction.Result res = action.call();
```

Python SDK

```
QueryVCenterPrimaryStorageAction action = QueryVCenterPrimaryStorageAction()
action.conditions = ["uuid=9fad4c72758b46d38c0866e460ef4b46"]
action.sessionId = "ceba885251524d33a45220db9b465a9e"
```

```
QueryVCenterPrimaryStorageAction.Result res = action.call()
```

4.1.9 查询vCenter镜像服务器(QueryVCenterBackupStorage)

API请求

URLs

```
GET zstack/v1/vcenters/backup-storage
GET zstack/v1/vcenters/backup-storage/{uuid}
```

Headers

```
Authorization: OAuth the-session-uuid
```

Curl示例

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth 7ef3836bac52419aad77c40b44fffb75" \
-X GET http://localhost:8080/zstack/v1/vcenters/backup-storage?q=uuid=5557b285420b48
b0b8c9bcea75b8b118
```

```
curl -H "Content-Type: application/json" \
-H "Authorization: OAuth 5176427af8404b40b3fdc2faa26790d1" \
-X GET http://localhost:8080/zstack/v1/vcenters/backup-storage/59ca8b1caccf41eca787b
0fbd354bde9
```

可查询字段

运行zstack-cli命令行工具，输入QueryVCenterBackupStorage并按Tab键查看所有可查询字段以及可跨表查询的资源名。

API返回

该API成功时返回一个空的JSON结构{}，出错时返回的JSON结构包含一个error字段，例如：

```
{
"error": {
    "code": "SYS.1001",
    "description": "A message or a operation timeout",
    "details": "Create VM on KVM timeout after 300s"
}
}
```

SDK示例

Java SDK

```
QueryVCenterBackupStorageAction action = new QueryVCenterBackupStorageAction();
action.conditions = asList("uuid=7557d599e0204b11b4dbf0d1dca47218");
action.sessionId = "76854653db20477199d441bc07a74552";
```

```
QueryVCenterBackupStorageAction.Result res = action.call();
```

Python SDK

```
QueryVCenterBackupStorageAction action = QueryVCenterBackupStorageAction()
action.conditions = ["uuid=ea80ba3ee9de4938a148d5b29fdc1780"]
action.sessionId = "365e725adbea460b9352adf7848a7026"
QueryVCenterBackupStorageAction.Result res = action.call()
```

术语表

区域（Zone）

ZStack中最大的一个资源定义，包括集群、二层网络、主存储等资源。

集群（Cluster）

一个集群是类似物理主机（Host）组成的逻辑组。在同一个集群中的物理主机必须安装相同的操作系统（虚拟机管理程序，Hypervisor），拥有相同的二层网络连接，可以访问相同的主存储。在实际的数据中心，一个集群通常对应一个机架（Rack）。

管理节点（Management Node）

安装系统的物理主机，提供UI管理、云平台部署功能。

计算节点（Compute Node）

也称之为物理主机（或物理机），为云主机实例提供计算、网络、存储等资源的物理主机。

主存储（Primary Storage）

用于存储云主机磁盘文件的存储服务器。支持本地存储、NFS、 Ceph、FusionStor、Shared Mount Point等类型。

镜像服务器（Backup Storage）

也称之为备份存储服务器，主要用于保存镜像模板文件。建议单独部署镜像服务器。

镜像仓库（Image Store）

镜像服务器的一种类型，可以为正在运行的云主机快速创建镜像，高效管理云主机镜像的版本变迁以及发布，实现快速上传、下载镜像，镜像快照，以及导出镜像的操作。

云主机（VM Instance）

运行在物理机上的虚拟机实例，具有独立的IP地址，可以访问公共网络，运行应用服务。

镜像（Image）

云主机或云盘使用的镜像模板文件，镜像模板包括系统云盘镜像和数据云盘镜像。

云盘（Volume）

云主机的数据盘，给云主机提供额外的存储空间，共享云盘可挂载到一个或多个云主机共同使用。

计算规格（Instance Offering）

启动云主机涉及到的CPU数量、内存、网络设置等规格定义。

云盘规格（Disk Offering）

创建云盘容量大小的规格定义。

二层网络（L2 Network）

二层网络对应于一个二层广播域，进行二层相关的隔离。一般用物理网络的设备名称标识。

三层网络（L3 Network）

云主机使用的网络配置，包括IP地址范围、网关、DNS等。

公有网络（Public Network）

由因特网信息中心分配的公有IP地址或者可以连接到外部互联网的IP地址。

私有网络（Private Network）

云主机连接和使用的内部网络。

L2NoVlanNetwork

物理主机的网络连接不采用Vlan设置。

L2VlanNetwork

物理主机节点的网络连接采用Vlan设置，Vlan需要在交换机端提前进行设置。

VXLAN网络池（VXLAN Network Pool）

VXLAN网络中的 Underlay 网络，一个 VXLAN 网络池可以创建多个 VXLAN Overlay 网络（即 VXLAN 网络），这些 Overlay 网络运行在同一组 Underlay 网络设施上。

VXLAN网络（VXLAN）

使用 VXLAN 协议封装的二层网络，单个 VXLAN 网络需从属于一个大的 VXLAN 网络池，不同 VXLAN 网络间相互二层隔离。

云路由（vRouter）

云路由通过定制的Linux云主机来实现的多种网络服务。

安全组（Security Group）

针对云主机进行第三层网络的防火墙控制，对IP地址、网络包类型或网络包流向等可以设置不同的安全规则。

弹性IP（EIP）

公有网络接入到私有网络的IP地址。

快照（Snapshot）

某一个时间点上某一个磁盘的数据备份。包括自动快照和手动快照两种类型。