Oracle® ZFS Storage Appliance RESTful API 指南，发行版 OS8.8.0

文件号码 **E97785-01**
2018 年 11 月

ORACLE

文件号码 E97785-01

文档可访问性

有关 Oracle 对可访问性的承诺，请访问 Oracle Accessibility Program 网站 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc。

获得 Oracle 支持

购买了支持服务的 Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

目录

Oracle ZFS Storage Appliance RESTful API 入门

Oracle ZFS Storage Appliance 可通过网络提供高效的文件和块数据服务。本指南介绍可用于管理设备的 Oracle ZFS Storage Appliance RESTful 应用编程接口 (Application Programming Interface, API)。RESTful 体系结构基于分层的客户机/服务器模型，此模型允许通过标准集线器、路由器和其他网络系统在没有客户机配置的情况下透明地重定向服务。

RESTful API 验证

Oracle ZFS Storage Appliance RESTful API 使用的验证凭证与浏览器用户界面 (browser user interface, BUI) 和命令行界面 (command-line interface, CLI) 相同。来自外部客户机的所有请求都单独使用设备凭证进行验证并在端口 215 上通过 HTTPS 连接来执行。RESTful API 支持 HTTPS 会话具有超时值 15 分钟，用户可定义该值。

可以采用以下验证形式之一：

- **基本验证**－每个请求都必须包含用户登录。

  HTTP 头示例：

  ```
  Authorization:  Basic abcefgMWE
  ```

- **用户验证**－使用 BUI 或 CLI 登录凭证进行验证。在这种情况下，X-Auth-User 头必须包含登录名，而 X-Auth-Key 头必须包含登录密码。

  HTTP 头示例：

  ```
  X-Auth-User:  login-name
  X-Auth-Key:  password-xxx
  ```

- **会话验证**－当会话通过验证后，可使用会话头继续运行命令，直至会话到期。会话到期后，在接受命令之前，必须再次执行验证。

  会话头示例：

  ```
  X-Auth-Session:  guigqpQRE4g89ngb
  ```

RESTful API 版本

设备给定发行版的 RESTful API 版本具有与设备软件版本匹配的全局版本号。所有请求的响应头中都会返回此版本号：

```
X-Zfssa-Version:  nas.2013.1.1
```

服务版本

每个服务都有一个作为统一资源标识符 (Uniform Resource Identifier, URI) 的一部分、用于访问服务的版本号。版本有主要版本号和次要版本号。请求必须提供主要版本号，但次要版本号是可选的，如果未提供次要版本号，则默认值为 `0`。请求中的主要版本号必须与服务的主要版本号匹配。请求中的次要版本号必须与服务的次要版本号匹配。

例如，下表显示了当客户机请求某个运行 `2.1` 版本的服务时，是否可以在客户机请求中使用指定的版本。

请求版本	允许
`1`	否：主要版本与服务运行的版本不匹配。
`2`	是：主要版本匹配，次要版本（默认值为 `0`）向后兼容。
`2.1`	是：主要版本值和次要版本值与服务运行的版本相匹配。
`2.2`	否：主要版本匹配，但次要版本比服务运行的版本新。

更改以下属性时，无需更改任何服务 API 版本。必须使用设备版本号和型号来确定哪些属性可用。这些属性更改还会反映在 CLI 和 BUI 中，并且是该设备实例的功能指示。

- 新的输出属性（不会删除旧属性）。
- 在现有命令中添加的新的输入属性，这些属性具有默认值，以使此命令执行其先前版本中所执行的操作。

由于新版本的向后兼容命令可以返回其他属性，因此应对客户机进行编码以忽略新属性。如果对服务 API 进行向后兼容的更改，则次要版本号将递增。

- 向现有服务添加新命令。
- 向服务命令添加新的查询参数。

如果对服务 API 进行不兼容的更改，则主要版本号将递增。

- 删除命令查询参数。
- 从现有服务中删除命令。

设备软件的主要版本可能包括不兼容的版本更改。执行主要更新期间，给定服务可能有也可能没有较早的版本。每个命令响应都必须包含 HTTP 头和给定模块的设备 API 的当前版本：

```
X-Zfssa-Nas-Api:  1.1
```

常用 RESTful 操作

下表显示了给定资源的常用 RESTful 操作。

表 1 常用 RESTful 操作

请求	路径	说明
GET	resources	列出所有资源
GET	resources/*name*	获取描述选定资源的 JSON 对象
POST	resources	创建新资源
PUT	resources/*name*	修改选定资源
DELETE	resources/*name*	删除选定资源

HTTP 响应正文

将按照 RFC 4627 中定义的 JSON 格式对所有响应数据进行编码。除非另有说明，否则针对单个资源的命令都将返回一个将该资源属性作为其名称的 JSON 结果对象。每个命令部分都将记录此 JSON 结果对象中返回了哪些属性名称。

除非另有说明，否则创建 (POST) 和修改 (PUT) 命令都会返回已创建或修改的资源的属性。内容应该与 GET 请求返回的值匹配。

示例正文：

```
{
    "resource_name": {
        "href": "path/to/this/resource",
        "property_01": "value_01",
        "property_02": "value_01
    }
}
```

某些 GET 命令返回资源列表。

```
{
    "resource_list_name": [
        {
            "href": "path/to/resource_01",
            "property_01": "value_01"
        }, {
            "href": "path/to/resource_02",
            "property_02", "value_02"
```

```
        }
    ]
}
```

注 - 在本文档中，对于命令所显示的 JSON 返回结果，已经通过添加回车和空格对这些结果设置了格式以增强可读性。实际输出不包含此格式设置。

HTTP 响应头

所有用于发送数据的设备服务命令都使用 JSON 数据格式，并需要以下头值：

```
Accept:  application/json
Content-Type:  application/json
```

响应头包括以下信息：

```
Date: Tue, 23 Jul 2013 13:07:37 GMT X-Zfs-Sa-Appliance-Api: 1.0 Content-Type: application/
json Content-Length: 357
```

对于列表式结果，在数据发回之前，内容长度可能是未知的。如果未提供内容长度，则客户机必须读取响应正文直至 EOF（End of File，文件结束符），以便读取所有返回的数据。

查询参数

一些请求使用可选查询参数，这些参数修改或增强返回的数据。有关详细信息，请参见各个资源的文档。不是每个资源都支持所有查询参数。本节仅记录了当资源实施指定的查询参数时将使用的常用查询参数。

表 2 常用查询参数

参数	说明
`props=true`	列出资源的属性元数据。默认值为 `false`。
`limit=`*n*	限制返回的列表元素数量。
`start=`*n*	用作返回的元素数据开头的索引号（或时间）。
`depth=`*n*	索引号指定返回数据的详细程度。
`match_`*property-name*`=`*value*	列出与指定属性名称和值匹配的数据。

查询参数：props

`props` 查询参数可用于 `GET`、`POST` 或 `PUT` 命令，以便最终用户访问元数据。最终用户通过将查询参数 `props` 设置为 `true` 来请求此功能。对于 `GET` 和 `PUT` 操作，返回的 JSON

对象将包含所需的数据以及属性的元数据列表。对于 POST，将仅返回元数据以帮助用户正确创建资源。

表 3 属性元数据值

属性	说明
name	属性名称
label	属性描述
immutable	此标志指示属性不可修改
type	属性类型，例如字符串、整数或布尔型。
choices	对于枚举属性，则为可用值的数组

查询参数：limit

limit 查询可用于返回大量元素的众多 GET 命令，以限制返回元素的最大数量。

查询参数：start

对于支持时间值的资源，index 可能是一个时间值（例如 20170531T01:13:58）并且必须以 UTC 时间表示。

查询参数：depth

depth 查询参数可用于检索资源列表的 GET 命令。该参数用于指定返回列表的详细程度。depth 的数值越大，返回的信息越详细。例如：

- /api/...?depth=0－返回节点属性和子节点名称。
- /api/...?depth=1－返回节点的属性、子节点的名称和属性、孙节点的名称。
- /api/...?depth=2－返回节点的属性、子节点的名称和属性以及孙节点的 depth=0 输出。

注 - 使用 /api/log/v1 列出日志以及使用 /api/storage/v1 列出池、项目、文件系统和 LUN 时不支持 depth 查询参数。

使用查询参数 depth 的请求示例：

```
GET /api/user/v1/users?depth=2 HTTP/1.1
Host: zfs-storage.example.com
X-Auth-User: root
X-Auth-Key: password-xxx
```

在此示例中，将按 `depth=2` 这一详细程度返回用户列表。

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1558
X-Zfssa-Access-Api: 1.0

{"users":
    [{
 "name": "root",
 "properties": {
  "logname": "root",
  "fullname": "Super-User",
  "initial_password": "password",
  "require_annotation": false
 },
 "children": [{
  "name": "preferences",
  "properties": {
   "locale": "C",
   "login_screen": "status/dashboard",
   "session_timeout": 15,
   "advanced_analytics": false
  },
  "children": [{
   "name": "keys",
   "properties": {},
   "children": [],
   "list": []
  }],
  "list": []
 }],
 "list": [],
 "href": "/api/user/v1/users/root"
    },
    {
 "name": "admin2",
 "properties": {
  "logname": "admin2",
  "fullname": "Administrator",
  "initial_password": "password",
  "require_annotation": false,
  "roles": ["basic"],
  "kiosk_mode": false,
  "kiosk_screen": "status/dashboard"
 },
 "children": [{
  "name": "exceptions",
  "properties": {},
  "children": [],
  "list": [{
   "name": "auth-000",
   "properties": {
    "scope": "stat",
    "drilldowns": "*",
    "allow_create": false,
    "allow_read": true
   },
   "children": [],
   "list": []
   },
   {
```

```
   "name": "auth-001",
   "properties": {
    "scope": "ad",
    "name": "*",
    "allow_domain": true,
    "allow_workgroup": false
   },
   "children": [],
   "list": []
  }]
 }, {
  "name": "preferences",
  "properties": {
   "locale": "C",
   "login_screen": "status/dashboard",
   "session_timeout": 15,
   "advanced_analytics": false
  },
  "children": [{
   "name": "keys",
   "properties": {},
   "children": [],
   "list": ["key-000"]
  }],
  "list": []
 }],
 "list": [],
 "href": "/api/user/v1/users/admin2"
    }]
}
```


查询参数：match

`match_`*property-name*`=`*value* 查询参数可用于检索资源列表的 `GET` 命令。它将返回与指定属性名称和值匹配的数据列表。例如：

- `/api/...?depth=0&match_kiosk_mode=true`—如果 `kiosk_mode` 为 `true`，返回过滤后的列表，其中包含子节点的名称。
- `/api/...?depth=1&match_kiosk_mode=true`—如果 `kiosk_mode` 为 `true`，返回过滤后的列表，其中包含的详细信息具体到 `depth=1`。
- `/api/...?depth=2&match_Fullname='Super*'&kiosk_mode=true`—如果包含 `Super` 和 `kiosk_mode` 的 `fullname` 为 `true`，返回过滤后的列表，其中包含的详细信息具体到 `depth=2`。

注 - 使用 `/api/log/v1` 列出日志以及使用 `/api/storage/v1` 列出池、项目、文件系统和 LUN 时不支持 `match_`*property-name*`=`*value* 查询参数。

设备错误

发生错误时将返回一个指示错误的 HTTP 状态代码以及以下故障响应有效载荷。

JSON 故障响应：

```
{
    fault: {
        message: 'ERR_INVALID_ARG',
        details: 'Error Details...',
        code: 500
    }
}
```

表 4 常见错误代码

错误	代码	说明
ERR_INVALID_ARG	400	输入参数无效
ERR_UNKNOWN_ARG	400	额外的未处理输入参数
ERR_MISSING_ARG	400	缺少必要的输入参数
ERR_UNAUTHORIZED	401	此用户未获得执行命令的授权
ERR_DENIED	403	操作被拒绝
ERR_STATE_CHANGED		系统状态冲突
ERR_NOT_FOUND	404	未找到请求项
ERR_OBJECT_EXISTS	409	请求创建已存在的对象
ERR_CONFIRM_REQUIRED	409	请求需要 `?confirm=true` 查询参数才能完成
ERR_OVER_LIMIT	413	输入请求太大，无法处理
ERR_UNSUPPORTED_MEDIA	415	请求的介质类型不受请求支持
ERR_NOT_IMPLEMENTED	501	操作未实施
ERR_BUSY	503	因资源有限，服务不可用

访问设置

协议版本和关联密码命令管理用于访问设备的 SSL/TLS 协议版本和密码。

安全协议和密码设置

默认情况下，SSL/TLS 协议版本 TLSv1.1、TLSv1.2 及其关联密码处于启用状态。您可以通过向 HTTPS 服务发送 PUT 请求以设置 tls_version 属性来启用 TLSv1.0。

请求示例：

```
PUT /api/service/v1/services/https HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json

{ "tls_version":  ["TLSv1.0", "TLSv1.1", "TLSv1.2"] }
```

结果示例（为了提高可读性，已人为地进行了断行）：

```
HTTP/1.1 202 Accepted
Content-Length: 1265
X-Zfssa-Service-Api: 1.1
X-Zfssa-Api-Version: 1.0
Content-Type: application/json; charset=utf-8

{
"service": {
"href": "/api/service/v1/services/https",
"<status>": "online",
"tls_version": "TLSv1 TLSv1.1 TLSv1.2",
"ciphers": "SRP-DSS-AES-256-CBC-SHA:SRP-RSA-AES-256-CBC-SHA:SRP-AES-256-CBC-SHA:
DH-DSS-AES256-GCM-SHA384:DHE-DSS-AES256-GCM-SHA384:DH-RSA-AES256-GCM-SHA384:DHE-
RSA-AES256-GCM-SHA384:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA256:DH-RSA-AES256-
SHA256:DH-DSS-AES256-SHA256:DHE-RSA-AES256-SHA:DHE-DSS-AES256-SHA:DH-RSA-AES256-
SHA:DH-DSS-AES256-SHA:DHE-RSA-CAMELLIA256-SHA:DHE-DSS-CAMELLIA256-SHA:DH-RSA-
CAMELLIA256-SHA:DH-DSS-CAMELLIA256-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-
SHA:CAMELLIA256-SHA:SRP-DSS-AES-128-CBC-SHA:SRP-RSA-AES-128-CBC-SHA:SRP-AES-128-
CBC-SHA:DH-DSS-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:DH-RSA-AES128-GCM-
SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-SHA256:DHE-DSS-AES128-SHA256:
DH-RSA-AES128-SHA256:DH-DSS-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA:
DH-RSA-AES128-SHA:DH-DSS-AES128-SHA:DHE-RSA-CAMELLIA128-SHA:DHE-DSS-CAMELLIA128-
SHA:DH-RSA-CAMELLIA128-SHA:DH-DSS-CAMELLIA128-SHA:AES128-GCM-SHA256:AES128-SHA256:
AES128-SHA:CAMELLIA128-SHA:SRP-DSS-3DES-EDE-CBC-SHA:SRP-RSA-3DES-EDE-CBC-SHA:SRP-
3DES-EDE-CBC-SHA:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DH-RSA-DES-CBC3-SHA:
DH-DSS-DES-CBC3-SHA:DES-CBC3-SHA"
}
}
```

要仅启用 TLSv1.0，请将 ciphers 属性设置为仅用于 TLSv1.0 的密码的列表。

请求示例（为了提高可读性，已人为地进行了断行）：

```
PUT /api/service/v1/services/https HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json

{
"tls_version":  ["TLSv1.0"] ,
"ciphers" : ["SRP-DSS-AES-256-CBC-SHA", "SRP-RSA-AES-256-CBC-SHA", "SRP-AES-256-CBC-SHA",
"DHE-RSA-AES256-SHA", "DHE-DSS-AES256-SHA", "DH-RSA-AES256-SHA", "DH-DSS-AES256-SHA",
"DHE-RSA-CAMELLIA256-SHA", "DHE-DSS-CAMELLIA256-SHA", "DH-RSA-CAMELLIA256-SHA",
"DH-DSS-CAMELLIA256-SHA", "AES256-SHA", "CAMELLIA256-SHA", "SRP-DSS-AES-128-CBC-SHA",
"SRP-RSA-AES-128-CBC-SHA", "SRP-AES-128-CBC-SHA", "DHE-RSA-AES128-SHA", "DHE-DSS-AES128-
SHA",
"DH-RSA-AES128-SHA", "DH-DSS-AES128-SHA", "DHE-RSA-CAMELLIA128-SHA", "DHE-DSS-CAMELLIA128-
SHA",
"DH-RSA-CAMELLIA128-SHA", "DH-DSS-CAMELLIA128-SHA", "AES128-SHA", "CAMELLIA128-SHA",
"SRP-DSS-3DES-EDE-CBC-SHA", "SRP-RSA-3DES-EDE-CBC-SHA", "SRP-3DES-EDE-CBC-SHA",
"EDH-RSA-DES-CBC3-SHA", "EDH-DSS-DES-CBC3-SHA", "DH-RSA-DES-CBC3-SHA", "DH-DSS-DES-CBC3-
SHA",
"DES-CBC3-SHA"]
}
```

结果示例（为了提高可读性，已人为地进行了断行）：

```
HTTP/1.1 202 Accepted
Content-Length: 809
X-Zfssa-Service-Api: 1.1
X-Zfssa-Api-Version: 1.0
Content-Type: application/json; charset=utf-8
```

```
{
"service": {
"href": "/api/service/v1/services/https",
"<status>": "online",
"tls_version": "TLSv1",
"ciphers": "SRP-DSS-AES-256-CBC-SHA:SRP-RSA-AES-256-CBC-SHA:SRP-AES-256-CBC-SHA:
DHE-RSA-AES256-SHA:DHE-DSS-AES256-SHA:DH-RSA-AES256-SHA:DH-DSS-AES256-SHA:DHE-RSA-
CAMELLIA256-SHA:DHE-DSS-CAMELLIA256-SHA:DH-RSA-CAMELLIA256-SHA:DH-DSS-CAMELLIA256-
SHA:AES256-SHA:CAMELLIA256-SHA:SRP-DSS-AES-128-CBC-SHA:SRP-RSA-AES-128-CBC-SHA:SRP-
AES-128-CBC-SHA:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA:DH-RSA-AES128-SHA:DH-DSS-
AES128-SHA:DHE-RSA-CAMELLIA128-SHA:DHE-DSS-CAMELLIA128-SHA:DH-RSA-CAMELLIA128-SHA:
DH-DSS-CAMELLIA128-SHA:AES128-SHA:CAMELLIA128-SHA:SRP-DSS-3DES-EDE-CBC-SHA:SRP-RSA-
3DES-EDE-CBC-SHA:SRP-3DES-EDE-CBC-SHA:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DH-
RSA-DES-CBC3-SHA:DH-DSS-DES-CBC3-SHA:DES-CBC3-SHA"
}
}
```

注 - 要避免被阻止使用 RESTful API 或 BUI，除非另有需要或者有 Oracle 支持人员指导，否则请保留 `tls_version` 和 `ciphers` 属性的默认设置。

使用 RESTful API

访问服务是 Oracle ZFS Storage Appliance 上所有 RESTful API 服务的入口点。本服务用于验证用户凭证和列出可用的 RESTful API 服务，包括其服务版本和访问点。

访问服务

要访问此服务，请使用此 URL：`http://`*hostname*`:215/api/access/v1`。

要访问其他服务，请使用本访问服务进行登录以获取可用服务的位置和版本，然后使用返回的 URI 访问这些服务。服务位置可能因当前设备配置或发行版级别而异。

表 5 访问服务命令

请求	路径	说明
GET	/api/access/v1	列出 RESTful API 服务访问点
POST	/api/access/v1	创建登录会话
DELETE	/api/access/v1	注销会话

列出服务

list services 命令列出可用服务访问 URI。如果不需要登录会话，则可使用 list services 以及相应的凭证来列出可用的服务访问 URI。此命令列出该设备上的所有可用的 RESTful API 服务和版本。

请求示例：

```
GET /api/access/v1 HTTP/1.1
Host: zfs-storage.example.com
X-Auth-User: admin1
X-Auth-Key: password
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 190
```

```
X-Zfssa-Access-Api: 1.0

{
    "access": {
        "services": [{
            "version": "1.0",
            "name": "appliance",
            "uri": "https://zfs-storage.example.com:215/api/appliance/v1"
        }, {
            "version": "1.0",
            "name": "nas",
            "uri": "https://zfs-storage.example.com:215/api/nas/v1"
        }, {
            "version": "1.0",
            "name": "replication",
            "uri": "https://zfs-storage.example.com:215/api/replication/v1"
        }, {
            "version": "1.0",
            "name": "san",
            "uri": "https://zfs-storage.example.com:215/api/san/v1"
        } ... ]
    }
}
```

获取服务命令

`get service` 命令返回该服务的相关信息，包括所有可用命令的列表。

请求示例：

```
GET /api/appliance/v1 HTTP/1.1
Host: zfs-storage.example.com
X-Auth-Session: guigqpQRE4g89ngb
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 204
X-Zfssa-Access-Api: 1.0

{
    "service": {
        "name": "appliance",
        "methods": [
            {
                "description": "Get appliance RESTful services",
                "path": "/apis",
                "request": "GET"
            },
            {
                "description": "Get appliance RESTful service properties",
                "path": "/apis/<api:path>",
                "request": "GET"
            },
            {
                "description": "Create a new alert threshold watch",
                "path": "/alerts/thresholds",
                "request": "POST"
```

```
            }, ... ]
    }
}
```

验证会话

可通过发送 POST 请求从访问服务获取验证会话 ID。所有其他服务可将此验证会话 ID 用作为身份凭证。当超过用户的会话超时属性所设置的超时期限后，此验证 ID 将失效。默认值通常为 15 分钟。DELETE 请求可用于注销会话 ID 并使会话 ID 无效。

当客户机可以重新发送每个请求及其验证信息时，无需验证会话。由于 RESTful API 操作没有状态，因此仅存储验证 ID。

登录会话

空白 POST 请求会请求新的登录对话。成功后，将返回 201 状态的 HTTP 以及具有单个属性 "access" 的 JSON 对象，此属性包含可用的 RESTful 的 API 服务列表。

登录请求示例：

```
POST /api/access/v1 HTTP/1.1
Host: zfs-storage.example.com
X-Auth-User: root
X-Auth-Key: password-xxx
```

成功登录后将返回 HTTP 状态 201 (Created)，以及通过 X-Auth-Session HTTP 头的会话 ID。响应正文包含通过此登录可访问的服务的列表。

响应标题：

```
HTTP/1.1 201 Created
X-Auth-Session: guigqpQRE4g89ngb
Content-Type: application/json
Content-Length: 378
X-Zfssa-Access-Api: 1.0

{
    "access": {
        "services":[{
            ...
        }]
    }
}
```

注销会话

空 DELETE 发送请求以注销并使会话失效。

注销请求示例：

```
DELETE /api/access/v1 HTTP/1.1
X-Auth-Session: guigqpQRE4g89ngb
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Access-Api: 1.0
```

RESTful API 警报服务

警报 RESTful API 服务允许您配置警报阈值以及对已发布警报的响应。

警报服务命令

下表显示了警报服务命令。

表 6 警报服务命令

请求	附加到路径 /api/alert/v1	说明
GET	仅使用 /api/alert/v1	列出警报服务命令
POST	/thresholds	创建新的警报阈值监视
GET	/thresholds/*threshold*	获取指定的警报阈值监视属性
GET	/thresholds	列出所有警报阈值监视对象
PUT	/thresholds/*threshold*	修改指定的警报阈值监视对象
DELETE	/thresholds/*threshold*	销毁指定的阈值对象
POST	/actions	创建新的警报操作
GET	/actions/*actions*	获取指定的警报操作属性
GET	/actions	列出所有警报操作对象
PUT	/actions/*actions*	修改指定的警报操作对象
DELETE	/actions/*actions*	销毁指定的操作对象
POST	/actions/*actions*	创建新的警报操作的操作
GET	/actions/*actions*/*action*	获取指定的警报操作的操作属性
PUT	/actions/*actions*/*action*	修改指定的警报操作的操作对象
DELETE	/actions/*actions*/*action*	销毁指定的操作对象
GET	/events	侦听新的警报事件

警报阀值

可设置阈值以创建定制警报监视。下表列出了用于管理警报阈值的典型属性。有关完整参考，请参见 CLI 帮助。

表 7 警报阀值

属性	类型	说明
uuid	Default	监视的唯一标识符 ("immutable")
statname	AnalyticsStatistics	要监视的统计数据 ["cpu.utilization"、"arc.accesses"、"arc.size"、"arc.l2_bytes"、"arc.l2_accesses"、"arc.l2_size"、"syscap.bytesused"、"yscap.percentused"、"metacap.bytesused"、"metacap.percentused"、"repl.bytes"、"repl.ops"、"shadow.kilobytes"、"shadow.ops"、"shadow.requests"、"io.bytes"、"io.ops"、"datalink.kilobytes"、"nic.kilobytes"、"net.kilobytes"、"ftp.kilobytes"、"fc.bytes"、"fc.ops"、"http.reqs"、"ndmp.bytes"、"ndmp.diskkb"、"ndmp.ops"、"nfs2.bytes"、"nfs2.ops"、"nfs3.bytes"、"nfs3.ops"、"nfs4.bytes"、"nfs4.ops"、"nfs4-1.bytes"、"nfs4-1.ops"、"sftp.kilobytes"、"smb.ops"、"srp.bytes"、"srp.ops"、"iscsi.bytes"、"iscsi.ops"]
type	ChooseOne	当 stat 超出限制 (normal) 或低于限制 (inverted) 时是否要发布警报 ["normal"、"inverted"]
limit	PositiveInteger	限制统计信息的值
minpost	Duration	发布警报前必须保持的最短时间条件
days	ChooseOne	仅在特定日期发布警报 ["all"、"weekdays"、"weekends"]
window_start	TimeOfDay	仅在 window_start 与 window_end 之间发布警报 ["none"、"00:00"、"00:30"、"01:00"、"01:30"、"02:00"、"02:30"、"03:00"、"03:30"、"04:00"、"04:30"、"05:00"、"05:30"、"06:00"、"06:30"、"07:00"、"07:30"、"08:00"、"08:30"、"09:00"、"09:30"、"10:00"、"10:30"、"11:00"、"11:30"、"12:00"、"12:30"、"13:00"、"13:30"、"14:00"、"14:30"、"15:00"、"15:30"、"16:00"、"16:30"、"17:00"、"17:30"、"18:00"、"18:30"、"19:00"、"19:30"、"20:00"、"20:30"、"21:00"、"21:30"、"22:00"、"22:30"、"23:00"、"23:30"]
window_end	TimeOfDay	仅在 window_start 与 window_end 之间发布警报 ["none"、"00:00"、"00:30"、"01:00"、"01:30"、"02:00"、"02:30"、"03:00"、"03:30"、"04:00"、"04:30"、"05:00"、"05:30"、"06:00"、"06:30"、"07:00"、"07:30"、"08:00"、"08:30"、"09:00"、"09:30"、"10:00"、"10:30"、"11:00"、"11:30"、"12:00"、"12:30"、"13:00"、"13:30"、"14:00"、"14:30"、"15:00"、"15:30"、"16:00"、"16:30"、"17:00"、"17:30"、"18:00"、"18:30"、"19:00"、"19:30"、"20:00"、"20:30"、"21:00"、"21:30"、"22:00"、"22:30"、"23:00"、"23:30"] ("immutable")
frequency	Duration	重新发布之前的最短时间
minclear	Duration	重新发布 "all clear" 警报之前的最短正常时间

列出警报阈值

列出所有配置的警报阈值。

请求示例：

```
GET /api/alert/v1/thresholds HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Date: Tue, 27 Aug 2013 17:38:40 GMT
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 689

{
    "thresholds": [
        {
            "days": "all",
            "frequency": 300,
            "href": "/api/alert/v1/thresholds/
                     bec758cb-346e-6a7d-c211-b320c09ef6a6",
            "limit": 500,
            "minclear": 300,
            "minpost": 300,
            "statname": "cpu.utilization",
            "threshold": "threshold-000",
            "type": "normal",
            "uuid": "bec758cb-346e-6a7d-c211-b320c09ef6a6",
            "window_end": 0,
            "window_start": -1
        },
        {
            "days": "all",
            "frequency": 300,
            "href": "/api/alert/v1/thresholds/
                     475799d8-32c8-6ff6-882c-aa3b66e3a5a2",
            "limit": 100000,
            "minclear": 600,
            "minpost": 300,
            "statname": "datalink.kilobytes",
            "threshold": "threshold-001",
            "type": "normal",
            "uuid": "475799d8-32c8-6ff6-882c-aa3b66e3a5a2",
            "window_end": 300,
            "window_start": 1200
        }
    ]
}
```

获取警报阈值

列出单个警报阈值的属性。

请求示例：

```
GET /api/alert/v1/thresholds/1b15d405-75c4-4c0c-e0f6-8a108165b874
    HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 363

{
    "threshold": {
        "days": "weekdays",
        "frequency": 300,
        "href": "/api/alert/v1/thresholds/
                 1b15d405-75c4-4c0c-e0f6-8a108165b874",
        "limit": 100000,
        "minclear": 300,
        "minpost": 300,
        "statname": "datalink.kilobytes",
        "type": "normal",
        "uuid": "1b15d405-75c4-4c0c-e0f6-8a108165b874",
        "window_end": 0,
        "window_start": -1
    }
}
```

创建警报阈值

创建警报阈值。

请求示例：

```
POST /api/alert/v1/thresholds HTTP/1.1
Host: zfs-storage.example.com
X-Auth-User: root
X-Auth-Key: password
Content-Type: application/json
Content-Length: 50

{"statname": "datalink.kilobytes", "limit": 100000}
```

响应示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 321
Location: /api/alert/v1/thresholds
          /1b15d405-75c4-4c0c-e0f6-8a108165b874

{
    "threshold": {
        "href": "/api/alert/v1/alerts/thresholds
```

```
                /1b15d405-75c4-4c0c-e0f6-8a108165b874",
        ...
    }
}
```

修改警报阈值

修改指定的警报阈值的任何属性。

请求示例：

```
PUT /api/alert/v1/thresholds/1b15d405-75c4-4c0c-e0f6-8a108165b874
    HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215

{"days":"weekdays"}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 326

{
    "threshold": {
        "days": "weekdays",
        ...
    }
}
```

删除警报阈值

删除指定的警报阈值。

请求示例：

```
DELETE /api/alert/v1/thresholds/475799d8-32c8-6ff6-882c-aa3b66e3a5a2
       HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

警报操作

`category` 属性确定要定义的警报操作的类型。每个类别都定义了其自己的属性集。

支持的类别为：

- ad
- all
- appliance_software
- backup
- cluster
- custom
- hardware
- hardware_faults
- ndmp
- network
- replication
- replication_source
- replication_target
- restore
- scrk
- shadow
- smf
- thresholds
- zfs_pool

表 8 警报操作 "ad"

属性	类型	说明
active_directory_degraded	Boolean	过滤器应匹配 active_directory_degraded 事件：`true` 或 `false`
smb_kerberos_client_authentication_degraded	Boolean	过滤器应匹配 mb_kerberos_client_authentication_degraded 事件：`true` 或 `false`

表 9 警报操作 "all"

属性	类型	说明
all_defects	Boolean	过滤器应匹配 all_defects 事件
service_alerts	Boolean	过滤器应匹配 service_alerts 事件：`true` 或 `false`
all_hardware_faults	Boolean	过滤器应匹配 all_hardware_faults 事件：`true` 或 `false`

表 10 警报操作 "appliance_software"

属性	类型	说明
obstacles_to_system_software_update	Boolean	过滤器应匹配 obstacles_to_system_software_update 事件：`true` 或 `false`
operating_system_kernel_panic	Boolean	过滤器应匹配 operating_system_kernel_panic 事件：`true` 或 `false`

表 11 警报操作 "backup"

属性	类型	说明
backup_finished	Boolean	过滤器应匹配 backup_finished 事件：`true` 或 `false`
backup_started	Boolean	过滤器应匹配 backup_started 事件：`true` 或 `false`

表 12 警报操作 "cluster"

属性	类型	说明
cluster_i/o_link_down	Boolean	过滤器应匹配 cluster_i/o_link_down 事件：`true` 或 `false`
cluster_i/o_link_failed	Boolean	过滤器应匹配 cluster_i/o_link_failed 事件：`true` 或 `false`
cluster_i/o_link_up	Boolean	过滤器应匹配 cluster_i/o_link_up 事件：`true` 或 `false`
unexpected_peer_error_occurred	Boolean	过滤器应匹配 unexpected_peer_error_occurred 事件：`true` 或 `false`
communication_to_peer_lost	Boolean	过滤器应匹配 communication_to_peer_lost 事件：`true` 或 `false`
cluster_peer_panicked	Boolean	过滤器应匹配 cluster_peer_panicked 事件：`true` 或 `false`
failed_to_set_sp_root_password_on_cluster_peer	Boolean	过滤器应匹配 failed_to_set_sp_root_password_on_cluster_peer 事件：`true` 或 `false`
cluster_rejoin_failed_on_peer	Boolean	过滤器应匹配 cluster_rejoin_failed_on_peer 事件：`true` 或 `false`
cluster_rejoin_mismatch_on_peer	Boolean	过滤器应匹配 cluster_rejoin_mismatch_on_peer 事件：`true` 或 `false`
cluster_rejoin_completed_on_peer	Boolean	过滤器应匹配 cluster_rejoin_completed_on_peer 事件：`true` 或 `false`
cluster_peer_lost_communication_token	Boolean	过滤器应匹配 cluster_peer_lost_communication_token 事件：`true` 或 `false`
cluster_rejoin _failed	Boolean	过滤器应匹配 cluster_rejoin_failed 事件：`true` 或 `false`

属性	类型	说明
cluster_rejoin_mismatch	Boolean	过滤器应匹配 cluster_rejoin_mismatch 事件：`true` 或 `false`
cluster_rejoin_completed	Boolean	过滤器应匹配 cluster_rejoin_completed 事件：`true` 或 `false`
cluster_takeover_complete	Boolean	过滤器应匹配 cluster_takeover_complete 事件：`true` 或 `false`
resource_import_failed_during_cluster_takeover	Boolean	过滤器应匹配 resource_import_failed_during_cluster_takeover 事件：`true` 或 `false`
local_cluster_communication_token_lost	Boolean	过滤器应匹配 local_cluster_communication_token_lost 事件：`true` 或 `false`

表 13 警报操作 "custom"

属性	类型	说明
patterns	Default	FMA 事件模式

表 14 警报操作 "hardware"

属性	类型	说明
fibre_channel_port_down	Boolean	过滤器应匹配 fibre_channel_port_down 事件：`true` 或 `false`
multiple_transient_fibre_channel_port_status_changes	Boolean	过滤器应匹配 multiple_transient_fibre_channel_port_status_changes 事件：`true` 或 `false`
transient_fibre_channel_port_status_change	Boolean	过滤器应匹配 transient_fibre_channel_port_status_change 事件：`true` 或 `false`
fibre_channel_port_up	Boolean	过滤器应匹配 fibre_channel_port_up 事件：`true` 或 `false`
network_port_down	Boolean	过滤器应匹配 network_port_down 事件：`true` 或 `false`
network_port_up	Boolean	过滤器应匹配 network_port_up 事件：`true` 或 `false`
chassis_connected_to_system	Boolean	过滤器应匹配 chassis_connected_to_system 事件：`true` 或 `false`
chassis_removed	Boolean	过滤器应匹配 chassis_removed 事件：`true` 或 `false`
hardware_component_inserted	Boolean	过滤器应匹配 hardware_component_inserted 事件：`true` 或 `false`
hardware_component_removed	Boolean	过滤器应匹配 hardware_component_removed 事件：`true` 或 `false`
disk_inserted	Boolean	过滤器应匹配 disk_inserted 事件：`true` 或 `false`
disk_removed	Boolean	过滤器应匹配 disk_removed 事件：`true` 或 `false`
i/o_path_added	Boolean	过滤器应匹配 i/o_path_added 事件：`true` 或 `false`
i/o_path_removed	Boolean	过滤器应匹配 i/o_path_removed 事件：`true` 或 `false`

属性	类型	说明
service_processor_offline_or_unavailable	Boolean	过滤器应匹配 service_processor_offline_or_unavailable 事件：`true` 或 `false`
service_processor_online_after_outage	Boolean	过滤器应匹配 service_processor_online_after_outag 事件：`true` 或 `false`
failed_to_set_root_password_on_service_processor	Boolean	过滤器应匹配 failed_to_set_root_password_on_service_processor 事件：`true` 或 `false`

表 15 警报操作 "hardware_faults"

属性	类型	说明
all_hardware_faults	Boolean	过滤器应匹配 all_hardware_faults 事件：`true` 或 `false`

表 16 警报操作 "ndmp"

属性	类型	说明
invalid_ndmp_restore	Boolean	过滤器应匹配 invalid_ndmp_restore 事件：`true` 或 `false`
backup_finished	Boolean	过滤器应匹配 backup_finished 事件：`true` 或 `false`
backup_started	Boolean	过滤器应匹配 backup_started 事件：`true` 或 `false`
restore_finished	Boolean	过滤器应匹配 restore_finished 事件：`true` 或 `false`
restore_started	Boolean	过滤器应匹配 restore_started 事件：`true` 或 `false`

表 17 警报操作 "network"

属性	类型	说明
datalink_failed	Boolean	过滤器应匹配 datalink_failed 事件 ： `true` 或 `false`
datalink_ok	Boolean	过滤器应匹配 datalink_ok 事件：`true` 或 `false`
network_port_down	Boolean	过滤器应匹配 network_port_down 事件：`true` 或 `false`
network_port_up	Boolean	过滤器应匹配 network_port_up 事件：`true` 或 `false`
ip_address_conflict	Boolean	过滤器应匹配 ip_address_conflict 事件：`true` 或 `false`
ip_address_conflict_resolved	Boolean	过滤器应匹配 ip_address_conflict_resolved 事件：`true` 或 `false`
ip_interface_degraded	Boolean	过滤器应匹配 ip_interface_degraded 事件：`true` 或 `false`

属性	类型	说明
ip_interface_failed	Boolean	过滤器应匹配 ip_interface_failed 事件：`true` 或 `false`
ip_interface_ok	Boolean	过滤器应匹配 ip_interface_ok 事件：`true` 或 `false`

表 18 警报操作 "replication"

属性	类型	说明
receive_failed_(unsupported _version)	Boolean	过滤器应匹配 receive_failed_(unsupported_version) 事件：`true` 或 `false`
receive_failed_(cancelled)	Boolean	过滤器应匹配 receive_failed_(cancelled) 事件：`true` 或 `false`
receive_failed_(all_others)	Boolean	过滤器应匹配 receive_failed_(all_others) 事件：`true` 或 `false`
receive_failed_(out_of_space)	Boolean	过滤器应匹配 receive_failed_(out_of_space) 事件：`true` 或 `false`
receive_failed_(package_not _upgraded)	Boolean	过滤器应匹配 receive_failed_(package_not_upgraded) 事件：`true` 或 `false`
receive_finished	Boolean	过滤器应匹配 receive_finished 事件：`true` 或 `false`
receive_started	Boolean	过滤器应匹配 receive_started 事件：`true` 或 `false`
send_failed_(unsupported_version)	Boolean	过滤器应匹配 send_failed_(unsupported_version) 事件：`true` 或 `false`
send_failed_(cancelled)	Boolean	过滤器应匹配 send_failed_(cancelled) 事件：`true` 或 `false`
send_failed_(all_others)	Boolean	过滤器应匹配 send_failed_(all_others) 事件：`true` 或 `false`
send_failed_(connectivity)	Boolean	过滤器应匹配 send_failed_(connectivity) 事件：`true` 或 `false`
send_failed_(out_of_space)	Boolean	过滤器应匹配 send_failed_(out_of_space) 事件：`true` 或 `false`
send_failed_(remote _verification)	Boolean	过滤器应匹配 send_failed_(remote_verification) 事件：`true` 或 `false`
send_finished	Boolean	过滤器应匹配 send_finished 事件：`true` 或 `false`
send_skipped_(already_running)	Boolean	过滤器应匹配 send_skipped_(already_running) 事件：`true` 或 `false`
send_started	Boolean	过滤器应匹配 send_started 事件：`true` 或 `false`

表 19 警报操作 "replication_source"

属性	类型	说明
send_failed_(unsupported_version)	Boolean	过滤器应匹配 send_failed_(unsupported_version) 事件：`true` 或 `false`
send_failed_(cancelled)	Boolean	过滤器应匹配 send_failed_(cancelled) 事件：`true` 或 `false`
send_failed_(all_others)	Boolean	过滤器应匹配 send_failed_(all_others) 事件：`true` 或 `false`
send_failed_(connectivity)	Boolean	过滤器应匹配 send_failed_(connectivity) 事件：`true` 或 `false`
send_failed_(out_of_space)	Boolean	过滤器应匹配 send_failed_(out_of_space) 事件：`true` 或 `false`
send_failed_(remote_verification)	Boolean	过滤器应匹配 send_failed_(remote_verification) 事件：`true` 或 `false`
send_finished	Boolean	过滤器应匹配 send_finished 事件：`true` 或 `false`
send_skipped_(already_running)	Boolean	过滤器应匹配 send_skipped_(already_running) 事件：`true` 或 `false`
send_started	Boolean	过滤器应匹配 send_started 事件：`true` 或 `false`

表 20 警报操作 "replication_target"

属性	类型	说明
receive_failed_(unsupported_verrsion)	Boolean	过滤器应匹配 receive_failed_(unsupported_version) 事件：`true` 或 `false`
receive_failed_(cancelled)	Boolean	过滤器应匹配 receive_failed_(cancelled) 事件：`true` 或 `false`
receive_failed _(all_others)	Boolean	过滤器应匹配 receive_failed_(all_others) 事件：`true` 或 `false`
receive_failed _(out_of_space)	Boolean	过滤器应匹配 receive_failed_(out_of_space) 事件：`true` 或 `false`
receive_failed_(package_not_upgraded)	Boolean	过滤器应匹配 receive_failed_(package_not_upgraded) 事件：`true` 或 `false`
receive_finished	Boolean	过滤器应匹配 receive_finished 事件：`true` 或 `false`
receive_started	Boolean	过滤器应匹配 receive_started 事件：`true` 或 `false`

表 21 警报操作 "restore"

属性	类型	说明
restore_finished	Boolean	过滤器应匹配 restore_finished 事件：`true` 或 `false`

属性	类型	说明
restore_started	Boolean	过滤器应匹配 restore_started 事件：`true` 或 `false`

表 22 警报操作 "scrk"

属性	类型	说明
support_bundle_build_failed	Boolean	过滤器应匹配 support_bundle_build_failed 事件：`true` 或 `false`
support_bundle_sent	Boolean	过滤器应匹配 support_bundle_sent 事件：`true` 或 `false`
support_bundle _upload_failed	Boolean	过滤器应匹配 support_bundle_upload_failed 事件：`true` 或 `false`
an_update_is_available_on_my_oracle_support.	Boolean	过滤器应匹配 an_update_is_available_on_my_oracle_support. 事件：`true` 或 `false`
no_updates_available.	Boolean	过滤器应匹配 no_updates_available. 事件：`true` 或 `false`
the_appliance_failed_to_verify_if_an_update_is_available	Boolean	过滤器应匹配 the_appliance_failed_to_verify_if_an_update_is_available 事件：`true` 或 `false`

表 23 警报操作 "shadow"

属性	类型	说明
shadow_migration_complete	Boolean	过滤器应匹配 shadow_migration_complete 事件：`true` 或 `false`

表 24 警报操作 "smf"

属性	类型	说明
service_failures	Boolean	过滤器应匹配 service_failures 事件：`true` 或 `false`

表 25 警报操作 "thresholds"

属性	类型	说明
thresholdid	Default	应匹配其警报的监视的 UUID

表 26 警报操作 "zfs_pool"

属性	类型	说明
resilver_finished	Boolean	过滤器应匹配 resilver_finished 事件：`true` 或 `false`
resilver_started	Boolean	过滤器应匹配 resilver_started 事件：`true` 或 `false`

属性	类型	说明
scrub_finished	Boolean	过滤器应匹配 scrub_finished 事件：`true` 或 `false`
scrub_started	Boolean	过滤器应匹配 scrub_started 事件：`true` 或 `false`
hot_spare_activated	Boolean	过滤器应匹配 hot_spare_activated 事件：`true` 或 `false`

列出警报操作

列出警报操作命令可列出所有警报操作。要获取单个资源的数据，请向给定警报操作资源的 href 属性发送 HTTP `GET` 请求。

获取警报操作的请求示例：

```
GET /api/alert/v1/actions HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 1395

{
    "actions": [
        {
            "action": "actions-000",
            "category": "smf",
            "href": "/api/alert/v1/actions/actions-000",
            "service_failures": true
        },
        {
            "action": "actions-001",
            "category": "scrk",
            "href": "/api/alert/v1/actions/actions-001",
            "action-000": {
                    "handler": "snmp_trap",
                    "href": "/api/alert/v1/alerts/actions/actions-001
                             /action-000"
            },
            "action-001": {
                    "address": "admin@example.com",
                    "handler": "email",
                    "href": "/api/alert/v1/actions/actions-001
                             /action-001",
                    "subject": "Phone Home Alert"
            },
            "support_bundle_build_failed": true,
            "support_bundle_sent": true,
            "support_bundle_upload_failed": true
        },
        {
```

```
            "action": "actions-002",
            "category": "thresholds",
            "href": "/api/alert/v1/actions/actions-002",
            "action-000": {
                    "address": "admin@example.com",
                    "handler": "email",
                    "href": "/api/alert/v1/actions/actions-002
                             /action-000",
                    "subject": "CPU Busy Alert"
            },
            "thresholdid": "b182ca05-53d3-6604-b874-ec353335704d"
        }
    ]
}
```

获取警报操作

此命令类似于列出警报操作命令，但此命令仅返回指定的警报操作。

请求示例：

```
GET /api/alert/v1/actions/actions-002 HTTP/1.1
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 331

{
    "action": {
        "category": "thresholds",
        "href": "/api/alert/v1/actions/actions-002",
        "action-000": {
                "address": "admin@example.com",
                "handler": "email",
                "href": "/api/alert/v1/alerts/actions/actions-002
                         /action-000",
                "subject": "CPU Busy"
        },
        "thresholdid": "b182ca05-53d3-6604-b874-ec353335704d"
    }
}
```

创建警报操作

当您创建一个包含 JSON 对象的警报操作 `POST` 请求时，必须向 /api/alert/v1/alerts/actions 发送操作属性。必须设置 `category` 属性以选择要创建的操作类型。有关给定系统上的所有可用 `category` 值，请参见 CLI 文档。

下面是典型的 `category` 值：

```
"ad"
"all"
```

```
"appliance_software"
"backup"
"cluster"
"custom"
"hardware"
"hardware_faults"
"ndmp"
"network"
"replication"
"replication_source"
"replication_target"
"restore"
"scrk"
"shadow",
"smf"
"thresholds"
"zfs_pool"
```

请求示例：

```
POST /api/alert/v1/actions HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-Session: uerqghq84vbdv
Content-Type: application/json
Content-Length: 30

{"category": "hardware_faults"}
```

响应示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 118
Location: /api/alert/v1/actions/actions-006

{
    "action": {
        "href": "/api/alert/v1/actions/actions-006",
        "category": "hardware_faults",
        "all_hardware_faults": true
    }
}
```

修改警报操作

可通过发送 HTTP `PUT` 请求来修改由 `list` 命令返回的一些属性。

请求示例：

```
PUT /api/alert/v1/actions/actions-001 HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-Session: uerqghq84vbdv
Content-Type: application/json
Content-Length: 30

{"support_bundle_sent": false}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 195

{
    "action": {
        "href": "/api/alert/v1/actions/actions-001",
        "category": "scrk",
        "support_bundle_build_failed": true,
        "support_bundle_sent": false,
        "support_bundle_upload_failed": true
    }
}
```

删除警报操作

向任何警报操作 href 或操作 href 发送 HTTP `DELETE` 请求可删除指定的资源。成功删除的响应为 HTTP 状态 204 (No Content)。

请求示例：

```
DELETE /api/alert/v1/actions/actions-003 HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

警报操作项目

各个操作项目将被添加到每个警报操作列表中。

创建警报项目

此操作可将警报操作添加到现有警报操作组中。

请求示例：

```
POST /api/alert/v1/actions/actions-001 HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-Session: uerqghq84vbdv
Content-Type: application/json
Content-Length: 68

{"address": "admin@example.com", "handler": "email", "subject":"CPU Busy"}
```

响应示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 177
Location: /api/alert/v1/actions/actions-001/action-001

{
    "action": {
        "href": "/api/alert/v1/actions/actions-001
                 /action-001",
        "handler": "email",
        "address": "admin@example.com",
        "subject": "CPU Busy"
    }
}
```

修改警报操作

此操作可修改现有警报操作。

请求示例：

```
PUT /api/alert/v1/actions/actions-001/action-000 HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-Session: uerqghq84vbdv
Content-Type: application/json
Content-Length: 28

{"address": "admin@example.com"}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 176
X-Zfssa-Version: user/generic@2013.06.08,1-0

{
    "action": {
        "href": "/api/alert/v1/actions/actions-001
                 /action-000",
        "handler": "email",
        "address": "admin@example.com",
        "subject": "CPU Busy"
    }
}
```

删除警报操作项目

对于给定的警报操作，可删除单个操作。要删除一个参数，请向此操作 href 属性发送 DELETE 请求。

删除操作的请求示例：

```
DELETE /api/alert/v1/actions/actions-001/action-000 HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-Session: uerqghq84vbdv

HTTP/1.1 204 No Content
```

Analytics 服务

使用 Analytics，可以用图表实时显示各种统计信息以及记录数据以供将来检索。您可以执行长期监视，也可以执行短期分析。Analytics 使用 DTrace 来动态创建定制的统计信息，从而对操作系统堆栈的各个层进行详细分析。

Analytics 命令

可获取以下 Analytics 服务，网址为 `http://`*hostname*`/api/analytics/v1.0/`。

表 27 Analytics 命令

请求	附加到路径 /analytics/v1	说明
GET	仅使用 /analytics/v1	列出 Analytics 服务信息
POST	/worksheets	创建新的 Analytics 数据集
GET	/worksheets/*worksheet*	获取指定的 Analytics 数据集属性
GET	/worksheets	列出所有 Analytics 数据集对象
PUT	/worksheets/*worksheet*	修改指定的 Analytics 数据集对象
DELETE	/worksheets/*worksheet*	销毁指定的工作表对象
PUT	/worksheets/*worksheet*/suspend	暂停所有工作表数据集
PUT	/worksheets/*worksheet*/resume	恢复所有工作表数据集
POST	/worksheets/*worksheet*/datasets	创建新的工作表数据集
GET	/worksheets/*worksheet*/datasets/*dataset*	获取指定的工作表数据集属性
GET	/worksheets/*worksheet*/datasets	列出所有工作表数据集对象
PUT	/worksheets/*worksheet*/datasets/*dataset*	修改指定的工作表数据集对象
DELETE	/worksheets/*worksheet*/datasets/*dataset*	销毁指定的数据集对象
POST	/datasets	创建新的 Analytics 数据集
GET	/datasets/*dataset*	获取指定的 Analytics 数据集属性
GET	/datasets	列出所有 Analytics 数据集对象
PUT	/datasets/*dataset*	修改指定的 Analytics 数据集对象
DELETE	/datasets/*dataset*	销毁指定的数据集对象
PUT	/datasets	暂停或恢复所有数据集

请求	附加到路径 /analytics/v1	说明
PUT	/datasets/*dataset*/data	保存此数据集（如果未保存）
DELETE	/datasets/*dataset*/data	将给定 [粒度] 的数据从此数据集中删除
GET	/settings	列出 Analytics 设置
PUT	/settings	修改 Analytics 设置

Analytics 设置

下表中描述的属性允许您收集所有分析数据、设置数据要保留的小时数以及设置主机名查找策略。

属性	说明
`retain_second_data`	每秒数据的保留间隔（小时）
`retain_minute_data`	每分钟数据的保留间隔（小时）
`retain_hour_data`	每小时数据的保留间隔（小时）
`hostname_lookup`	主机名查找策略

获取设置

此命令获取分析属性的当前值。

请求示例：

```
GET /api/analytics/v1/settings HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 131
X-Zfssa-Analytics-Api: 1.0

{
    "settings": {
        "href": "/api/analytics/v1/settings",
        "retain_hour_data": 600,
        "retain_minute_data": 400,
        "retain_second_data": 200,
        "hostname_lookup": true
    }
}
```

修改设置

修改设置命令用于修改 Analytics 设置，例如数据保留值和主机名查找策略。

请求示例：

```
PUT /api/analytics/v1/settings HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Content-Type: application/json
Content-Length: 60

{"retain_hour_data":600, "retain_minute_data":400, "retain_second_data":200,
 "hostname_lookup":true}
```

结果示例：

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 101
X-Zfssa-Analytics-Api: 1.0

{
    "settings": {
        "href": "/api/analytics/v1/settings",
        "retain_hour_data": 600,
        "retain_minute_data": 400,
        "retain_second_data": 200,
        "hostname_lookup": true
    }
}
```

Analytics 工作表

工作表是为统计信息绘制图形的 BUI 屏幕。可以同时绘制多项统计信息，并可以为工作表指定一个标题并保存该工作表供以后查看。保存工作表这一操作将自动对所有打开的统计信息执行归档操作，这意味着将继续读取并永久归档任何打开的统计信息。工作表命令可用于管理从 BUI 中获得的工作表。

下表显示了 Analytics 工作表中使用的属性。

属性	说明
ctime	创建此工作表的时间和日期
mtime	上次修改此工作表的时间和日期
name	此工作表的名称
owner	此工作表的所有者
uuid	此工作表的通用唯一标识符

列出工作表

列出当前配置的所有 Analytics 工作表。

请求示例：

```
GET /api/analytics/v1/worksheets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 237
X-Zfssa-Analytics-Api: 1.0

{
    "worksheets": [{
        "href": "/api/analytics/v1/worksheets/ab59bcbc...",
        "uuid": "ab59bcbc-080a-cf1a-98c9-9f485bc3a43d"
    }, {
        "href": "/api/analytics/v1/worksheets/bb3ee729...",
        "uuid": "bb3ee729-080a-cf1a-98c9-9f485bc3a43d"
    }]
}
```

获取 Analytics 工作表

获取单个 Analytics 工作表。

请求示例：

```
GET /api/analytics/v1/worksheets/ab59bcbc-080a-cf1a-98c9-9f485bc3a43d
    HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 237
X-Zfssa-Analytics-Api: 1.0

{
    "worksheet": {
        "ctime": "Thu Jun 13 2013 02:17:14 GMT+0000 (UTC)",
        "href": "/api/analytics/v1/worksheets
                 /ab59bcbc-080a-cf1a-98c9-9f485bc3a43d",
        "mtime": "Sun Jun 23 2013 16:22:01 GMT+0000 (UTC)",
        "name": "myworksheet",
        "owner": "root",
        "uuid": "ab59bcbc-080a-cf1a-98c9-9f485bc3a43d"
```

```
    }
}
```

创建工作表

创建新的 Analytics 工作表。

请求示例：

```
POST /api/analytics/v1/worksheets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
Content-Length: 26

{"name": "myworksheet"}
```

结果示例：

```
HTTP/1.1 201 Created
Content-Length: 280
Location: /api/analytics/v1/worksheets/bb3ee729-4480-4609-89b2-fae2dc016bec

{
    "worksheet": {
        "uuid": "bb3ee729-4480-4609-89b2-fae2dc016bec",
        "name": "myworksheet",
        "owner": "root",
        "ctime": "Fri Aug 23 2013 20:35:00 GMT+0000 (UTC)",
        "mtime": "Fri Aug 23 2013 20:35:00 GMT+0000 (UTC)",
        "href": "/api/analytics/v1/worksheets
                 /bb3ee729-4480-4609-89b2-fae2dc016bec"
    }
}
```

重命名工作表

重命名保存的工作表。

请求示例：

```
PUT /api/analytics/v1/worksheets/a442e761-4048-4738-b95f-be0824d7ed09
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
Content-Length: 26

{"name": "test"}
```

响应示例：

```
HTTP/1.1 202 Accepted
Date: Tue, 20 Dec 2016 00:33:06 GMT
Server: TwistedWeb/192.0.2
Content-Length: 279
X-Zfssa-Version: user/generic@2013.06.05.7.0,1-1.12
```

```
X-Zfssa-Analytics-Api: 1.1
X-Zfssa-Api-Version: 1.0
Content-Type: application/json; charset=utf-8

{
    "worksheet": {
        "href": "/api/analytics/v1/worksheets/a442e761-4048-4738-b95f-be0824d7ed09",
        "uuid": "a442e761-4048-4738-b95f-be0824d7ed09",
        "name": "test",
        "owner": "root",
        "ctime": "Wed Dec 14 2016 03:58:28 GMT+0000 (UTC)",
        "mtime": "Tue Dec 20 2016 00:25:57 GMT+0000 (UTC)"
    }
}
```

销毁工作表

销毁 Analytics 工作表。在此示例中，工作表名称用作工作表标识符，但也可以使用 href 中标识的 uuid。此命令的行为与可销毁工作表的 CLI 命令的行为相匹配。

请求示例：

```
DELETE /api/analytics/v1/worksheets/name=myworksheet HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
Content-Length: 26
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Analytics-Api: 1.0
```

列出工作表数据集

列出指定的工作表中的所有数据集。

下表显示了数据集配置中使用的属性。

属性	说明
name	此数据集的底层统计信息的名称
drilldowns	下钻当前突出显示（如果有）
seconds	此数据集显示的秒数

请求示例：

```
GET /api/analytics/v1/worksheets/name=myworksheet/datasets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
```

```
Host: zfs-storage.example.com:215
Accept: application/json
```

添加工作表数据集

创建工作表数据集。

请求示例：

```
POST /api/analytics/v1/worksheets/name=myworksheet/datasets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
Content-Length: 26

{"name": "nfs4.ops", "seconds": 300}
```

结果示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
X-Zfssa-Analytics-Api: 1.0
Location: /api/analytics/v1/worksheets/name=me/datasets/nfs4.ops
Content-Length: 162

{
    "dataset": {
        "href": "/api/analytics/v1/worksheets/name=me/datasets/dataset-008",
        "name": "nfs4.ops",
        "width": 0,
        "drilldowns": [],
        "seconds": 300,
        "time": ""
    }
}
```

修改工作表数据集

修改现有的工作表数据集。

请求示例：

```
PUT /api/analytics/v1/worksheets/name=myworksheet/datasets/dataset-008
    HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
Content-Length: 26

{"seconds": 60}
```

结果示例：

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 161
```

```
X-Zfssa-Analytics-Api: 1.0

{
    "dataset": {
        "href": "/api/analytics/v1/worksheets/name=me/datasets/dataset-008",
        "name": "nfs4.ops",
        "width": 0,
        "drilldowns": [],
        "seconds": 60,
        "time": ""
    }
}
```

Analytics 数据集

Analytics 数据集使用以下属性。除 suspended 以外的所有其他属性都不可变。

属性	说明
name	此数据集的底层统计信息的名称
grouping	此统计信息的所属组
explanation	底层统计信息的说明
incore	内核中的数据集数据的字节数
size	磁盘上的数据集数据的字节数
suspended	指示数据集当前是否处于挂起状态的布尔值
activity	待定数据集活动标志

可用数据集：

- arc.accesses[hit/miss]
- arc.l2_accesses[hit/miss]
- arc.l2_size
- arc.size
- arc.size[component]
- cpu.utilization
- cpu.utilization[mode]
- dnlc.accesses[hit/miss]
- fc.bytes
- fc.ops
- ftp.kilobytes
- http.reqs
- io.bytes

- io.bytes[op]
- io.disks[utilization=95][disk]
- io.ops
- io.ops[disk]
- io.ops[op]
- iscsi.bytes
- iscsi.ops
- metacap.bytesused
- metacap.percentused
- ndmp.diskkb
- nfs2.ops
- nfs2.ops[op]
- nfs3.ops
- nfs3.ops[op]
- nfs4.ops
- nfs4.ops[op]
- nfs4-1.ops
- nfs4-1.bytes
- nic.kilobytes
- nic.kilobytes[device]
- nic.kilobytes[direction]
- sftp.kilobytes
- smb.ops
- smb.ops[op]

列出数据集

列出所有配置的 Analytic 数据集。

请求示例：

```
GET /api/analytics/v1/datasets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 237
```

```
X-Zfssa-Analytics-Api: 1.0

{
    "datasets": [{
        "dataset": "dataset-000",
        "href": "/api/analytics/v1/datasets/arc.accesses[hit/miss]",
        "name": "arc.accesses[hit/miss]"
    }, {
        "dataset": "dataset-001",
        "href": "/api/analytics/v1/datasets/arc.l2_accesses[hit/miss]",
        "name": "arc.l2_accesses[hit/miss]",
    }, {
        "dataset": "dataset-002",
        "href": "/api/analytics/v1/datasets/arc.l2_size",
        "name": "arc.l2_size",
    }, {
        "dataset": "dataset-003",
        "href": "/api/analytics/v1/datasets/arc.size",
        "name": "arc.size",
    }, {
        "dataset": "dataset-004",
        "href": "/api/analytics/v1/datasets/arc.size[component]",
        "name": "arc.size[component]",
    }, {
        ...
    }]
}
```

获取数据集

获取指定的数据集的属性。

请求示例：

```
GET /api/analytics/v1/datasets/nfs4.ops HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 237
X-Zfssa-Analytics-Api: 1.0

{
    "dataset": {
        "activity": "none",
        "dataset": "dataset-030",
        "explanation": "NFSv4 operations per second",
        "grouping": "Protocol",
        "href": "/api/analytics/v1/datasets/nfs4.ops",
        "incore": 296128,
        "name": "nfs4.ops",
        "size": 53211540,
        "suspended": false
    }
}
```

创建数据集

创建新的数据集。

请求示例：

```
POST /api/analytics/v1/datasets HTTP/1.1
X-Auth-User: root
X-Auth-Key: password
Content-Type: application/json
Content-Length: 26

{"statistic": "test.sine"}
```

结果示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 200
Location: /api/analytics/v1/datasets/test.sine

{
    "dataset":{
        "href": "/api/analytics/v1/datasets",
        "name": "test.sine",
        "grouping": "Test",
        "explanation": "sine per second",
        "incore": 34752,
        "size": 31912,
        "suspended": false,
        "activity": "none"
    }
}
```

修改数据集

修改数据集命令用于暂停或恢复单个数据集的数据收集。

暂停请求示例：

```
POST /api/analytics/v1/datasets/nfs4.ops

{"suspended":true}
```

恢复请求示例：

```
POST /api/analytics/v1/datasets/nfs4.ops
        {"suspended":false}
```

响应示例：

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 228
X-Zfssa-Analytics-Api: 1.0
```

```
{
    "dataset" {
        ...
        "suspended": false
    }
}
```

销毁数据集

销毁数据集。

请求示例：

```
DELETE /api/analytics/v1/datasets/test.sine HTTP/1.1
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Analytics-Api: 1.0
```

保存数据集

保存数据集。

请求示例：

```
PUT /api/analytics/v1/datasets/nfs4.ops/data
```

响应示例：

```
HTTP/1.1 202 Accepted
```

删改数据集数据

下表显示了在删改数据集时使用的查询参数。

参数	说明
`granularity`	删改粒度。可以按粒度值 `second`、`minute` 或 `hour` 删改数据集内的数据。
`endtime`	删改在给定 `endtime` 之前收集的数据。`endtime` 是一个 ISO 8601 时间/日期字符串（例如 20130531T01:13:58）。

请求示例：

```
DELETE /api/analytics/v1/datasets/nfs4.ops/data?granularity=hour
```

响应示例：

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

获取数据集数据

从 Analytic 数据集获取数据。支持检索每秒的数据和细粒度数据。

下表显示了用于获取数据集数据的基于时间的查询参数。

参数	说明
start	开始收集示例数据的时间。默认值为当前时间。
seconds	收集样例数据的秒数。默认值是 1。如果指定了 span 和 interval 参数，则会忽略 seconds 参数。
span	收集样例数据的持续时间：minute、hour、day、week、month 或 year。
granularity	在给定范围内提供数据点平均值时采用的粒度：minute、hour、day、week、month 或 year。

如果未提供 start 参数，则开始时间设置为当前时间减去指定样例数据的秒数 (seconds)。start 时间不能是将来时间。如果收集数据所需的秒数会导致数据返回时间晚于当前时间，服务器将等待各个样例收集完毕，然后再返回数据。

要检索细粒度数据，请使用 span 和 granularity 参数的组合。使用了 span 和 granularity 时，将忽略 seconds 参数。如果 span 和 granularity 中的任一参数输入不正确，则会忽略此请求并改用 seconds 参数。如果请求不正确或不受支持，将显示错误消息 "Input span and granularity are not supported"。

下面的列表描述了如何将 span 和 granularity 参数组合使用：

- 如果 span 为 minute，则 granularity 只能是 minute。
- 如果 span 为 hour，则 granularity 可以是 minute 或 hour。
- 如果 span 为 day，则 granularity 可以是 minute、hour 或 day。
- 如果 span 为 week，则 granularity 可以是 hour、day 或 week。
- 如果 span 为 month，则 granularity 可以是 day、week 或 month。
- 如果 span 为 year，则 granularity 可以是 week、month 或 year。

下表显示了返回的数据集数据属性。

属性	说明
startTime	返回的第一个样例的时间

属性	说明
sample	返回的第一个样例的样例索引
data	样例数据组
min	在指定粒度内的每秒最小值
max	在指定粒度内的每秒最大值

startTime 属性可为以下格式之一：

- ISO 8601 时间/日期字符串（例如，20130531T01:13:58）
- 样例索引号
- 文本字符串 now

用于收集三秒内实时数据的请求示例：

```
GET /api/analytics/v1/datasets/nfs4.ops%5Bfile%5D/data?start=now&seconds=3 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: text/x-yaml
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: text/x-yaml
X-Zfssa-Analytics-Api: 1.0
Transfer-Encoding: chunked

---
data:
  - sample: 239024557
    data:
        value:      5
    startTime:      20130912T21:42:38
    samples:        239024558

  - sample: 239024558
    data:
        value:      15
    startTime:      20130912T21:42:39
    samples:        239024559

  - sample: 239024559
    data:
        value:      25
    startTime:      20130912T21:42:40
    samples:        239024560

size:       3
---
```

用于收集一周内每一天的实时数据的请求示例：

```
GET /api/analytics/v1/datasets/nfs4.ops%5Bfile%5D/data?
start=239024557&span=week&granularity=day
    HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
```

```
Accept: text/x-yaml
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: text/x-yaml
X-Zfssa-Analytics-Api: 1.0
Transfer-Encoding: chunked

---
data:
  - sample: 239024557
    data:
       value:      5
    max:        79
    min:        0
    startTime:      20130912T21:42:38
    samples:        240074328

  - sample: 239110957
    data:
        value:      15
     max:        150
     min:        1
     startTime:      20130913T21:42:38
     samples:        240074328

    ...

   - sample: 239629357
     data:
         value:      25
     max:        120
     min:        2
     startTime:      20130914T21:42:38
     samples:        240074328

size:       7
```

硬件服务

本节介绍硬件群集、机箱和组件的管理。

群集

群集命令用于设置群集和管理群集资源。

表 28 群集命令

请求	附加到路径 /hardware/v1	说明
GET	/cluster	获取群集属性和群集资源列表
GET	/cluster/resources/*resource:path*	获取指定的群集资源的属性
PUT	/cluster/resources/*resource:path*	修改指定的群集资源
PUT	/cluster/failback	对分配给群集对等设备的所有资源都执行故障恢复
PUT	/cluster/takeover	接管所有分配给群集对等设备的资源
PUT	/cluster/unconfigure	取消群集设备的单机模式配置
GET	/cluster/links	获取群集卡链路状态
PUT	/cluster/setup	执行初始群集设置

获取群集属性

获取当前的群集配置状态和资源属性。

请求示例：

```
GET /api/hardware/v1/cluster HTTP/1.1
Authorization: Basic abcd45sMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 529
```

```
X-Zfssa-Api: 1.0

{
    "cluster": {
        "description": "Clustering is not configured",
        "peer_asn": "",
        "peer_description": "",
        "peer_hostname": "",
        "peer_state": "",
        "resources": {
            "net/ixgbe0": {
                "details": ["ipaddr-1"],
                "href": "/hardware/v1/cluster/resources/resources/net/ixgbe0",
                "owner": "admin1",
                "type": "singleton",
                "user_label": "Untitled Interface"
            },
            "zfs/zfs-storage-1": {
                "details": ["821G"],
                "href": "/hardware/v1/cluster/resources/resources/zfs/zfs-storage-1",
                "owner": "admin1",
                "type": "singleton",
                "user_label": ""
            }
        },
        "state": "AKCS_UNCONFIGURED"
    }
}
```

获取群集资源

根据群集资源中的 href 属性，可以获取该单个群集资源的数据。上例中有两个可用资源：/hardware/v1/cluster/resources/resources/zfs/zfs-storage-1 和 /hardware/v1/cluster/resources/resources/net/ixgbe0。

修改群集资源

当系统完成群集设置后，可以使用此命令修改各个群集资源的属性。有关更多信息，请参见 CLI "configuration cluster resources"。

群集命令

群集支持的命令有 `failback`、`takeover` 和 `unconfigure`。所有命令都会向群集资源发出 `PUT` 请求，并附上该命令的名称。成功后，这些命令都返回 HTTP 状态 202 (Accepted)。

故障恢复操作以异步方式执行。当 REST 客户机使用 PUT 命令发送故障恢复请求时，会在成功收到请求之后返回 HTTP 状态 202 (Accepted)。客户机将需要通过侦听警报或轮询群集状态来监视故障恢复进度。

请求示例：

```
PUT /api/hardware/v1/cluster/failback HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

如果群集未处于正确状态，无法接受命令，则返回 HTTP 状态 409 (Conflict)。

群集链路

此命令返回群集卡的当前链路状态。输出与 `aksh` 命令 "configuration cluster links" 的输出相同。建议在运行群集设置之前运行此命令，以确保群集布线没有问题。运行设置之前，所有链路都应处于 AKCIOS_ACTIVE 状态。

请求示例：

```
GET /api/hardware/v1/cluster/links HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 181

{
    "links": {
        "clustron2_embedded:0/clustron_uart:0 = AKCIOS_TIMEDOUT\n
         clustron2_embedded:0/clustron_uart:1 = AKCIOS_TIMEDOU\n
         clustron2_embedded:0/dlpi:0 = AKCIOS_TIMEDOUT"
    }
}
```

设置群集

`setup cluster` 命令可为系统设置初始群集。所有群集链路都应处于 AKCIOS_ACTIVE 状态，且应打开对等设备系统的电源但不进行配置，否则此命令将失败。

请求示例：

```
PUT /api/hardware/v1/cluster/setup HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

```
{"nodename": "zfs-storage-2", "password": "password"}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

机箱

硬件命令用于获取设备硬件机箱和组件的列表。

表 29 硬件命令

请求	附加到路径 /hardware/v1.0	说明
GET	/chassis	列出硬件机箱
GET	/chassis/*chassis*	获取指定的硬件机箱属性
PUT	/chassis/*chassis*	修改指定的硬件机箱属性
GET	/chassis/*chassis*/*fru_type*	列出硬件机箱组件
GET	/chassis/*chassis*/*fru_type*/*fru*	获取指定的机箱组件属性
PUT	/chassis/*chassis*/*fru_type*/*fru*	修改硬件机箱组件属性

列出机箱

get 机箱命令不会使用任何参数，会返回系统机箱对象的列表。命令成功执行后，将返回 HTTP 状态 200 (OK)。

属性	类型	说明
name	string	机箱名称
model	string	机箱型号
manufacturer	string	机箱制造商
serial	string	机箱序列号
revision	string	机箱修订级别
part	string	机箱更换部件号
type	string	机箱存储类型
faulted	boolean	故障指示灯
fru	string	机箱的 FMRI 表示形式
uuid	string	机箱 uuid 标识符

请求示例：

```
GET /api/hardware/v1/chassis HTTP/1.1
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Length: 788
Content-Type: application/json
X-Zfssa-Appliance-Api: 1.0

{
    "hardware": [{
        "faulted": false,
        "href": "/api/hardware/v1/chassis/chassis-000",
        "manufacturer": "Oracle",
        "model": "Oracle ZFS Storage ZS3-1",
        "name": "cairo",
        "rpm": "--",
        "serial": "1211FM200C",
        "type": "system"
    }, {
        "faulted": false,
        "href": "/api/hardware/v1/chassis/chassis-001",
        "locate": false,
        "manufacturer": "Oracle",
        "model": "Oracle Storage DE2-24C",
        "name": "1235FM4002",
        "part": "7046842",
        "path": 2,
        "revision": "0010",
        "rpm": 7200,
        "serial": "1235FM4002",
        "type": "storage"
    }, {
        "faulted": false,
        "href": "/api/hardware/v1/chassis/chassis-002",
        "locate": false,
        "manufacturer": "Oracle",
        "model": "Oracle Storage DE2-24P",
        "name": "50050cc10c206b96",
        "part": "7046836",
        "path": 2,
        "revision": "0010",
        "rpm": 10000,
        "serial": "50050cc10c206b96",
        "type": "storage"
    }]
}
```

获取机箱组件

此命令返回指定机箱中的所有硬件组件。命令成功执行后，将返回 HTTP 状态 200 (OK)。

请求示例：

```
GET /api/nas/v1/chassis/chassis-001 HTTP/1.1
Host: zfs-storage.example.com
```

```
Accept: application/json
```

响应示例：

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

{
    "chassis": {
        "type": "storage"
        "faulted": false,
        "href": "/api/hardware/v1/chassis/chassis-001",
        "locate": false,
        "manufacturer": "Oracle",
        "model": "Oracle Storage DE2-24C",
        "name": "1235FM4002",
        "part": "7046842",
        "path": 2,
        "revision": "0010",
        "rpm": 7200,
        "serial": "1235FM4002",
        "disk": [{
            "device": "c0t5000CCA01A76A2B8d0",
            "faulted": false,
            "href": "/api/hardware/v1/chassis/chassis-001/disk/disk-000",
            "interface": "SAS",
            "label": "HDD 0",
            "locate": false,
            "manufacturer": "HITACHI",
            "model": "H7230AS60SUN3.0T",
            "pathcount": 4,
            "present": true,
            "revision": "A310",
            "rpm": 7200,
            "serial": "001210R37LVD        YHJ37LVD",
            "size": 3000592982016,
            "type": "data",
            "use": "peer"
        }, {
            "href": "/api/hardware/v1/chassis/chassis-001/disk/disk-001",
            ...
        }, {
            "href": "/api/hardware/v1/chassis/chassis-001/disk/disk-002",
            ...
        },  ...  {
            "href": "/api/hardware/v1/chassis/chassis-001/disk/disk-023",
             ...
        }],
    "fan": [
        {
            "href": "/api/hardware/v1/chassis/chassis-001/fan/fan-000",
            ...
        },  ...  {
            "href": "/api/hardware/v1/chassis/chassis-001/fan/fan-007",
        }],
    "psu": [
        {
            "href": "/api/hardware/v1/chassis/chassis-001/psu/psu-000",
            ...
        }, {
            "href": "/api/hardware/v1/chassis/chassis-001/psu/psu-001",
        }, {
            "href": "/api/hardware/v1/chassis/chassis-001/psu/psu-002",
        }, {
```

```
            "href": "/api/hardware/v1/chassis/chassis-001/psu/psu-003",
        }],
    "slot": [{
            "href": "/api/hardware/v1/chassis/chassis-001/slot/slot-000",
        }, {
            "href": "/api/hardware/v1/chassis/chassis-001/slot/slot-001",
        }],
    }
}
```

获取硬件组件

此命令返回单个硬件组件中的属性。命令成功执行后，将返回 HTTP 状态 200 (OK)。响应对象包含下表中包含的组件属性。

属性	类型	说明
device	string	FRU 设备 ID
faulted	boolean	此标志指示 FRU 是否发生故障
fru	string	FRU 的 FMRI 表示形式
interface	string	FRU 接口类型
label	string	FRU 位置标签
locate	boolean	标志上的定位指示灯
manufacturer	string	FRU 制造商
model	string	FRU 型号
part	string	FRU 部件号
present	boolean	FRU 线上状态指示灯
rpm	number	磁盘片的 RMP（仅适用于磁盘）
serial	string	FRU 序列号
size	number	FRU 大小（容量）
type	string	组件类型
use	string	组件使用情况枚举

请求示例：

```
GET /api/hardware/v1/chassis/chassis-001/disk/disk-011 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "disk": {
```

```
        "device": "c0t5000CCA01A764FB0d0",
        "faulted": false,
        "href": "/api/hardware/v1/chassis/chassis-001/disk/disk-011",
        "interface": "SAS",
        "label": "HDD 11",
        "locate": false,
        "manufacturer": "HITACHI",
        "model": "H7230AS60SUN3.0T",
        "pathcount": 4,
        "present": true,
        "revision": "A310",
        "rpm": 7200,
        "serial": "001210R322ED        YHJ322ED",
        "size": 3000592982016,
        "type": "data",
        "use": "peer"
    }
}
```

修改组件属性

可使用 PUT 请求在选定硬件组件上设置属性。成功的请求会返回 HTTP 状态 201 (Accepted) 以及使用 JSON 格式的组件属性。

请求示例：

```
PUT /api/hardware/v1/chassis/chassis-001/disk/disk-011 HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-User: root
X-Auth-Key: password
Accept: application/json
Content-Type: application/json
Content-Length: 16

{"locate": true}
```

JSON 响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Length: 403
Content-Type: application/json

{
    "disk": {
        "href": "/api/hardware/v1/chassis/chassis-001/disk/disk-011",
        ...,
        "locate": true
    }
}
```

日志命令

日志命令用于管理 CLI "maintenance logs" 菜单下的可用日志。有关各个服务日志的信息，请参见服务 API。

管理日志命令

下表显示了如何调用管理日志命令。

表 30 管理日志命令

请求	附加到路径 /api/log/v1	说明
GET	仅使用 /api/log/v1	列出日志服务命令
GET	/logs	列出所有日志类型
GET	/logs/?start=*index/time*&limit=*entry limit*	获取选定范围的日志条目
GET	/logs/alert	列出所有警报日志
GET	/logs/alert?start=*index/time*&limit=*entry limit*	获取选定范围的日志条目
GET	/collect	下载所有日志条目的集合
GET	/collect?start=*index/time*&limit=*entry limit*	下载选定范围中日志条目的集合

列出日志

此命令列出设备上的所有可用日志。每个日志都会返回该日志中的条目数量和其中最后一个条目的时间戳。

注 - 不支持 `depth` 查询参数和 `match_property-name=value` 查询参数。

请求示例：

```
GET /api/log/v1/logs HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 532
X-Zfssa-Api: 1.0

{
    "logs": [
        {
            "href": "/api/log/v1/logs/fault",
            "name": "faults",
            "size": 16,
            "updated": "20130614T22:51:48"
        },
        {
            "href": "/api/log/v1/logs/audit",
            "name": "audits",
            "size": 460149,
            "updated": "20130730T22:10:41"
        },
        {
            "href": "/api/log/v1/logs/alert",
            "name": "alerts",
            "size": 13054,
            "updated": "20130728T00:06:10"
        },
        {
            "href": "/api/log/v1/logs/phone-home",
            "name": "phone-home",
            "size": 249,
            "updated": "20130730T03:22:35"
        },
        {
            "href": "/api/log/v1/logs/system",
            "name": "system",
            "size": 344,
            "updated": "20130724T03:21:55"
        }
    ]
}
```

获取日志条目

可以从指定的设备日志中返回日志条目。每个日志条目都会返回该条目的日期/时间以及日志特定的内容属性。

注 - 根据日志的数量，较旧的日志条目可能会因内存限制而不可用。BUI 和 CLI 中也有同样的限制。要获取所有系统日志，请使用“管理日志命令” [71]中描述的 `collect` 函数。

参数	说明
`start=`*index*	开始根据给定索引/时间返回日志
`limit=`*number*	限制返回的日志条目数

起始索引默认值为 0，表示返回生成的第一个日志。此值不允许为负值以及大于或等于日志大小的值。起始索引也可以是时间字符串；例如 20130724T03:21:55。

注 - REST 仅接受 UTC 时间。不接受早于当前日期一个月以上的时间值。旧日志的检索必须使用起始值的索引名称。限制值会限制针对给定请求返回的日志数量。返回的日志数量不超过给定的限制值。

请求示例：

```
GET /api/log/v1/logs/audit?limit=4&start=1000 HTTP/1.1
Authorization: Basic abcd45sMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
X-Zfssa-Api: development
Transfer-Encoding: chunked

{
    "logs": [
        {
            "address": "192.0.2.0",
            "annotation": "",
            "summary": "User logged in",
            "timestamp": "20131022T22:54:19",
            "user": "root"
        }, {
            "address": "192.0.2.0",
            "annotation": "",
            "summary": "Destroyed share \"zfs-storage-1:tst.volumes.py.34111.project/tst.
volumes.py.34111.lun.7\"",
            "timestamp": "20131022T22:52:34",
            "user": "root"
        }, {
            "summary": "Joined workgroup \"RESTTESTWG\"",
            "timestamp": "20131022T22:54:23",
            "user": "<system>"
        }, {
            "address": "192.0.2.0",
            "annotation": "",
            "summary": "User logged in",
            "timestamp": "20131022T22:54:19",
            "user": "root"
        }
    ]
}
```

下载日志

下载日志命令返回一个使用 gzip 压缩的 tar 文件，其中包含所有系统日志。文件配置名称设置为 logs.tar.gz。由于数据是实时创建并以流的形式处理的，因此无法恢复下载。

下载日志

如果仅需下载一个日志类型，则可将其名称附加到 `collect` 资源，如表中所示。日志的文本将以流的形式返回到客户机。如果请求进行 `gzip` 压缩，则将用 `gzip` 压缩文本流。其他压缩类型不受支持，并会被忽略。

网络命令

本节介绍的网络命令用于查看网络地址和设备，以及配置网络数据链路、接口和路由。

联网配置

联网配置功能允许您针对物理网络端口创建各种高级联网设置，包括链路聚合、虚拟 NIC (virtual NIC, VNIC)、虚拟 LAN (virtual LAN, VLAN) 和多路径组。然后可以为这些抽象内容定义任意数量的 IPv4 和 IPv6 地址，以便用于连接系统上的各种数据服务。

系统的网络配置有四个组成部分：

- **设备**－对应于您的物理网络连接或 InfiniBand 上的 IP (IP on InfiniBand, IPoIB) 分区的物理网络端口。
- **数据链路**－发送和接收数据包的基本结构。数据链路可以与设备（即物理网络端口）或 IB 分区一一对应，或者您可以定义由其他设备和数据链路组成的聚合、VLAN 和 VNIC 数据链路。
- **接口**－IP 配置和寻址的基本结构。每个 IP 接口都与一个数据链路关联，或者定义为包含其他接口的 IP 多路径 (IP MultiPathing, IPMP) 组。
- **路由**－IP 路由配置，用于控制系统对 IP 数据包的定向方式。

在此模型中，网络设备表示可用的硬件；它们没有可配置的设置。数据链路是第 2 层实体，必须创建它们，以便将 LACP 等设置应用到这些网络设备。接口是第 3 层实体，包含通过数据链路提供的 IP 设置。此模型将网络接口设置分为两个部分：数据链路对应第 2 层设置，接口对应第 3 层设置。

网络数据链路

网络数据链路命令可用于管理设备上的数据链路。您可以列出、修改、创建和删除数据链路资源。

表 31 网络数据链路命令

请求	附加到路径 /network/v1	说明
POST	/datalinks	创建新的网络数据链路

请求	附加到路径 /network/v1	说明
GET	/datalinks/*datalink*	获取指定的网络数据链路属性
GET	/datalinks	列出所有网络数据链路对象
PUT	/datalinks/*datalink*	修改指定的网络数据链路对象
DELETE	/datalinks/*datalink*	销毁指定的数据链路对象

表 32 物理设备数据链路属性

属性	类型	说明
`class`	string	“device” (“immutable”)
`label`	NetworkLabel	标签
`links`	ChooseOne	链路 ["igb1"、"igb0"、"ixgbe2"、"ixgbe3"、"igb4"、"igb3"、"ixgbe1"、"igb2"、"igb5"]
`jumbo`	Boolean	使用巨型帧 ["true"、"false"] ("deprecated")
`mtu`	PositiveInteger	最大传输单元 (Max transmission unit, MTU)
`speed`	ChooseOne	链路速度 ["auto"、"10"、"100"、"1000"、"10000"]
`duplex`	ChooseOne	链路双工 ["auto"、"half"、"full"]

表 33 VNIC 设备数据链路属性

属性	类型	说明
`class`	string	"vnic" ("immutable")
`label`	NetworkLabel	标签
`links`	ChooseOne	链路 ["ixgbe0"]
`mtu`	PositiveInteger	最大传输单元 (Max transmission unit, MTU)
`id`	VLAN	VLAN ID

表 34 VLAN 设备数据链路属性

属性	类型	说明
`class`	string	"vlan" ("immutable")
`label`	NetworkLabel	标签
`links`	ChooseOne	链路 ["ixgbe0"]
`mtu`	PositiveInteger	最大传输单元 (Max transmission unit, MTU)
`id`	VLAN	VLAN ID

表 35 基于聚合的设备数据链路属性

属性	类型	说明
`class`	string	"aggregation" ("immutable")
`label`	NetworkLabel	标签

属性	类型	说明
links	ChooseN	链路 ["igb1"、"igb0"、"ixgbe2"、"ixgbe3"、"igb4"、"igb3"、"ixgbe1"、"igb2"、"igb5"]
jumbo	Boolean	使用巨型帧 ["true"、"false"] ("deprecated")
mtu	PositiveInteger	最大传输单元 (Max transmission unit, MTU)
policy	ChooseOne	策略 ["L2"、"L3"、"L4"、"L2+L3"、"L2+L4"、"L3+L4"]
mode	ChooseOne	模式 ["active"、"passive"、"off"]
timer	ChooseOne	计时器 ["short"、"long"]
key	Integer	聚合键 ("immutable")

表 36 基于 IP 分区的设备数据链路属性

属性	类型	说明
class	string	"partition" ("immutable")
label	NetworkLabel	标签
links	ChooseOne	链路
pkey	Pkey	分区键
linkmode	ChooseOne	链路模式 ["cm"、"ud"]

列出网络数据链路

列出设备上的所有已配置数据链路。数据链路列表中的各个对象都包含用于获取单个数据链路资源的相关操作的 href 属性以及数据链路属性。

请求示例：

```
GET /api/network/v1/datalinks HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

JSON 数据示例：

```
{
    "datalinks": [{
        "href": "/api/network/v1/datalinks/ixgbe0",
        ...
    }, {
        "href": "/api/network/v1/datalinks/ixgbe1",
        ...
    }, {
        "href": "/api/network/v1/datalinks/ixgbe2",
        ...
    }, {
        "href": "/api/network/v1/datalinks/ixgbe3",
        ...
    }]
}
```

获取网络数据链路

GET 方法返回一个 JSON 对象，其中包含数据链路属性以及数据链路对象的列表。

```
GET /api/network/v1/datalinks/ixgbe0 HTTP/1.1 Host: zfs-storage.example.com

Accept: application/json
```

JSON 数据示例：

```
{
    "datalink": {
        "class": "device",
        "datalink": "ixgbe0",
        "duplex": "auto",
        "href": "/api/network/v1/datalinks/ixgbe0",
        "jumbo": false,
        "label": "Untitled Datalink",
        "links": [
            "ixgbe0"
        ],
        "mac": "0:21:28:a1:d9:68",
        "mtu": 1500,
        "speed": "auto"
    }
}
```

创建网络数据链路

POST 命令可创建新的数据链路。创建新的数据链路时需要的一个额外属性是 class 属性，此属性定义了要创建的数据链路的类。数据链路类是在创建数据链路期间定义的，它可以是以下类型之一：

- device－创建基于设备的数据链路
- vnic－创建基于 VNIC 的数据链路
- vlan－创建基于 VLAN 的数据链路
- aggregation－创建基于聚合的数据链路
- 分区－创建 IB 分区数据链路

这些属性映射到 "configuration net datalinks" 菜单中提供的相同 CLI 属性。

请求示例：

```
POST /api/network/v1/datalinks HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-User: root
X-Auth-Key: password
Content-Type: application/json
Content-Length: 78

{
```

```
    "class": "device",
    "jumbo": true,
    "links": ["ixgbe2"],
    "label": "TestDataLink"
}
```

响应示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Location: /api/network/v1/datalinks/ixgbe2
```

修改网络数据链路

PUT 方法可用于修改数据链路属性。有关设置数据链路的详细信息，请参见 CLI 文档。

请求示例：

```
PUT /api/network/v1/datalinks/ixgbe2 HTTP/1.1

{"jumbo": true}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 219

{
    "datalink": {
        "href": "/api/network/v1/datalinks/ixgbe2",
        "class": "device",
        "label": "MyDataLink",
        "links": ["ixgbe2"],
        "mac": "0:21:28:a1:d9:6a",
        "mtu": 9000,
        "duplex": "auto",
        "jumbo": true,
        "speed": "auto"
    }
}
```

删除网络数据链路

此命令可从系统中删除数据链路。使用 href 路径删除指定的数据链路。

请求示例：

```
DELETE /api/network/v1/datalinks/ixgbe2 HTTP/1.1
```

响应示例：

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

网络设备

这些命令列出系统上的物理网络设备。物理网络设备上没有可修改的属性。

表 37 网络设备命令

请求	附加到路径 /network/v1	说明
GET	/devices/*device*	获取指定的网络设备属性
GET	/devices	列出所有网络设备对象

表 38 网络设备属性

属性	说明
active	布尔标志，指示设备是否处于活动状态
duplex	设备的双工状态
factory_mac	出厂 MAC 地址
media	设备介质
speed	设备速度，单位为兆位/秒
up	布尔标志，指示设备是否正常运行

列出网络设备

此命令列出所有网络设备。

请求示例：

```
GET /api/network/v1/devices HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 412
X-Zfssa-Gns-Api: 1.0

{
    "devices": [{
        "href": "/api/network/v1/devices/ixgbe0",
        ....
    }, {
```

```
        "href": "/api/network/v1/devices/ixgbe1",
        ...
    }, {
        "href": "/api/network/v1/devices/ixgbe2",
        ...
    }, {
        "href": "/api/network/v1/devices/ixgbe3",
        ...
    }]
}
```

获取网络设备

此命令获取单个网络设备中的属性。

请求示例：

```
GET /api/network/v1/devices/ixgbe0 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 178
X-Zfssa-Gns-Api: 1.0

{
    "devices": {
        "active": false,
        "device": "ixgbe0",
        "duplex": "full-duplex",
        "factory_mac": "0:21:28:a1:d9:68",
        "href": "/api/network/v1/devices/ixgbe0",
        "media": "Ethernet",
        "speed": "1000 Mbit/s",
        "up": true
    }
}
```

网络接口

表 39 网络接口命令

请求	附加到路径 /api/network/v1	说明
POST	/interfaces	创建新的网络接口
GET	/interfaces/*interface*	获取指定的网络接口属性
GET	/interfaces	列出所有网络接口对象
PUT	/interfaces/*interface*	修改指定的网络接口对象

请求	附加到路径 /api/network/v1	说明
DELETE	/interfaces/*interface*	销毁指定的接口对象

表 40 网络接口属性

属性	说明
admin	此标志指示是否可在此接口上进行管理
class	类类型（"ip"、"ipmp"）（创建后不可变）
curaddrs	当前 IP 地址（不可变）
enable	此标志指示此接口是否已启用
label	接口的用户标签
links	为此接口选择网络链路
state	接口状态（不可变）
v4addrs	IPv4 地址
v6dhcp	IPv4 DHCP 标志
v6addrs	IPv6 地址
v6dhcp	IPv6 DHCP 标志

列出网络接口

此命令列出所有已配置的网络接口。

请求示例：

```
GET /api/network/v1/interfaces HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 369

{
    "interfaces": {[
        "href": "/api/network/v1/interfaces/ixgbe0",
        "v4addrs": ["ipaddr-1"]
        ...
    }, {
        "href": "/api/network/v1/interfaces/ixgbe1",
        "v4addrs": ["ipaddr-2"]
        ...
    }, {
        "href": "/api/network/v1/interfaces/ixgbe2",
        "v4addrs": ["ipaddr-3"]
        ...
    }, {
```

```
        "href": "/api/network/v1/interfaces/ixgbe3",
        "v4addrs": ["ipaddr-4"]
        ...
    }]
}
```

获取网络接口

此命令获取指定网络接口的完整属性列表。

请求示例：

```
GET /api/network/v1/interfaces/ixgbe0 HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 292

{
    "interface": {
        "admin": true,
        "class": "ip",
        "curaddrs": ["ipaddr-1"],
        "enable": true,
        "href": "/api/network/v1/interfaces/ixgbe0",
        "interface": "ixgbe0",
        "label": "Untitled Interface",
        "links": ["ixgbe0"],
        "state": "up",
        "v4addrs": ["ipaddr-1"],
        "v4dhcp": false,
        "v6addrs": [],
        "v6dhcp": false
    }
}
```

创建网络接口

此命令创建新的网络接口。

请求示例：

```
POST /api/network/v1/interfaces HTTP/1.1
Host: zfs-storage.example.com:215
X-Auth-User: root
X-Auth-Key: password
Content-Type: application/json
Content-Length: 78
```

```
{
    "class": "ip",
    "links": ["ixgbe3"],
    "v4addrs":"192.0.2.0/24"
}
```

响应示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Location: /api/network/v1/interfaces/ixgbe3
```

修改网络接口

此命令修改现有的网络接口。

请求示例：

```
PUT /api/network/v1/interfaces/ixgbe3 HTTP/1.1

{
    "v4addrs": ["192.0.2.0/24"],
    "interface": "Demo Rest"
}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 219

{
    "admin": true,
    "class": "ip",
    "curaddrs": ["192.0.2.0/24"],
    "enable": true,
    "href": "/api/network/v1/interfaces/ixgbe3",
    "interface": "ixgbe3",
    "label": "Demo Rest",
    "links": ["ixgbe3"],
    "state": "failed",
    "v4addrs": ["192.0.2.0/24"]
    "v4dhcp": false,
    "v6addrs": [],
    "v6dhcp": false
}
```

删除网络接口

此命令删除现有的网络接口。

注 - 删除某个接口后，将同时删除与该接口关联的所有路由。

请求示例：

```
DELETE /api/network/v1/interfaces/ixgbe3 HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
```

结果示例：

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

网络路由

这些命令管理网络路由。

表 41 管理网络路由

请求	附加到路径 /api/network/v1	说明
POST	/routes	创建新的网络路由
GET	/routes/*route*	获取指定的网络路由属性
GET	/routes	列出所有网络路由对象
DELETE	/routes/*route*	销毁指定的路由对象
GET	/routing	获取网络路由属性
PUT	/routing	修改网络路由属性

表 42 管理网络路由属性

属性	说明
`type`	路由类型，例如 "system" 或 "static"（不可变）
`family`	地址族（IPv4 或 IPv6）
`destination`	路由目标地址
`gateway`	网关地址
`interface`	网络数据链路接口

各个路由的 href 路径都使用在 CLI 中设置的路由 ID 集，但当修改路由时，这些值也会发生变化。API 支持使用路由中的唯一属性选择单个路由。语法为 `routes/name=`*value*，对应于 `routes/route-###`。

列出路由

列出在设备上创建的所有网络路由。

请求示例：

```
GET /api/network/v1/routes HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 192

{
    "routes": [{
        "destination": "ipaddr-0",
        "family": "IPv4",
        "gateway": "ipaddr-1",
        "href": "/api/network/v1/routing/route-000",
        "interface": "ixgbe0",
        "mask": 0,
        "route": "route-000",
        "type": "static"
    }, {
        "destination": "ipaddr-2",
        "family": "IPv4",
        "gateway": "ipaddr-3",
        "href": "/api/network/v1/routes/route-001",
        "interface": "ixgbe0",
        "mask": 24,
        "route": "route-001",
        "type": "system"
    }]
}
```

获取路由

获取单个路由的属性。

请求示例：

```
GET /api/network/v1/routes/destination=ipaddr-1 HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 192

{
    "route": {
        "destination": "ipaddr-1",
        "family": "IPv4",
        "gateway": "ipaddr-2",
        "href": "/api/network/v1/routes/route-001",
        "interface": "ixgbe0",
```

```
        "mask": 24,
        "route": "route-001",
        "type": "system"
    }
}
```

添加路由

创建新的网络路由。如果向系统添加其他路由，则路由 href 值会发生更改。创建路由时将不会返回路由信息，因为返回的属性将与输入属性相同。成功创建路由后会返回 HTTP 状态 204 (Created)。

创建静态路由的请求示例：

```
POST /api/network/v1/routes HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
Content-Type: application/json
Content-Length: 164

{
    "family": "IPv4",
    "destination": "ipaddr-0",
    "mask": "0",
    "gateway": "ipaddr-1",
    "interface": "ixgbe0"
}
```

结果示例：

```
HTTP/1.1 201 Created
```

删除路由

删除现有网络路由。

请求示例：

```
DELETE /api/network/v1/routes/route-001 HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfs-storage.example.com:215
```

结果示例：

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

RESTful API 问题服务

RESTful API 问题服务用于查看和管理由设备故障管理器发现的问题。

问题服务命令

表 43 问题服务命令

请求	附加到路径 /problem/v1	说明
GET	仅使用 /problem/v1	列出问题服务命令
GET	/problems	列出所有当前问题
GET	/problems/*problem*	获取具有指定 uuid 的问题的详细属性
PUT	/problems/*problem*/markrepaired	将指定问题 uuid 标记为已修复

列出所有问题

此命令列出设备上当前存在的所有问题。命令成功执行后，将返回 HTTP 状态 200 (OK)。

请求示例：

```
GET /api/problem/v1/problems HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "problems": [{
        "code": "AK-8003-Y6",
            "description": "The device configuration for JBOD
                       '1204FMD063' is invalid.",
            "impact": "The disks contained within the enclosure
                   cannot be used as part of a storage pool.",
        "uuid": "0d30be41-b50d-4d03-ddb4-edb69ee080f8",
        "repairable": false,
        "type": "Defect",
```

```
            "timestamp": "2013-2-21 17:37:12",
            "severity": "Major",
        "components": [{
                "certainty": 100,
                    "status": "degraded",
                    "uuid": "b4fd328f-92d6-4f0e-fb86-e3967a5473e7",
                "chassis": "1204FMD063",
                "label": "hc://:chassis-mfg=SUN
                    :chassis-name=SUN-Storage-J4410
                    :chassis-part=unknown
                    :chassis-serial=1204FMD063
                                :fru-serial=1204FMD063
                    :fru-part=7041262
                                :fru-revision=3529/ses-enclosure=0",
                "revision": "3529",
                    "part": "7041262",
                    "model": "Sun Disk Shelf (SAS-2)",
                    "serial": "1204FMD063",
                    "manufacturer": "Sun Microsystems, Inc."
        }]
    }]
}
```

列出一个问题

此命令会列出一个问题。命令成功执行后，将返回 HTTP 状态 200 (OK)。

列出问题命令使用 `uuid` 输入参数，该参数是单个问题的 UUID。

请求示例：

```
GET /api/problem/v1.0/problems/0d30be41-b50d-4d03-ddb4-edb69ee080f8
    HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "problem": {
        "uuid": "0d30be41-b50d-4d03-ddb4-edb69ee080f8",
        ...
    }
}
```

修复问题

修复问题命令将问题标记为已修复。

修复问题命令使用 `uuid` 输入参数，该参数是要标记为已修复的问题的 UUID。

请求示例：

```
PUT /api/problem/v1/problems/0d30be41-b50d-4d03-ddb4-edb69ee080f8/repaired
    HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功响应会返回 HTTP 状态 202 (Accepted)：

```
HTTP/1.1 202 Accepted
```

RESTful API 角色服务

角色是可分配给用户的一组特权。可能需要创建授权级别不同的管理员和操作员角色。可为员工分配满足其需求的任何角色，而不为其分配不必要的特权。使用角色比使用共享的管理员密码（例如向所有人提供 root 用户密码）更安全。角色限制用户仅具有必要的授权，并且在“审计”日志中将其操作归于各自的用户名。默认情况下，存在称为 "Basic administration"（基本管理）的角色，它包含非常基本的授权。

使用 RESTful API 角色服务可管理系统角色和授权。

角色服务命令概述

以下列表显示角色命令。

表 44 角色服务命令

请求	附加到路径 /role/v1	说明
GET	仅使用 /role/v1	列出角色服务命令
GET	/roles/*role*	获取指定的管理角色属性
GET	/roles	列出所有管理角色对象
PUT	/roles/*role*	修改指定的管理角色对象
DELETE	/roles/*role*	销毁指定的角色对象
POST	/roles	创建新角色或克隆现有角色
PUT	/roles/*role*/revoke	从所有用户中删除指定的角色
POST	/roles/*role*/authorizations	创建新角色授权
GET	/roles/*role*/authorizations/*auth*	获取指定的角色授权属性
GET	/roles/*role*/authorizations	列出所有角色授权对象
PUT	/roles/*role*/authorizations/*auth*	修改指定的角色授权对象
DELETE	/roles/*role*/authorizations/*auth*	销毁指定的授权对象

列出角色

每个角色都具有以下摘要属性。有关角色属性的完整说明，请参见 CLI 帮助。

表 45 角色属性

属性	类型	说明
name	string	角色名称（创建后不可变）
description	string	角色描述

请求示例：

```
GET /api/role/v1/roles HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
{
    "roles": [{
        "description": "Basic administration",
        "href": "/api/role/v1/roles/basic",
        "name": "basic",
        "role": "basic"
    }, {
        "description": "a",
        "href": "/api/role/v1/roles/rola",
        "name": "rola",
        "role": "rola"
    }]
}
```

获取角色

检索单个角色的属性。要返回属性元数据，请将 `props` 查询参数设置为 `true`。

请求示例：

```
GET /api/role/v1/roles/basic?props=true HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 390
```

```
{
    "props": [{
        "immutable": true,
        "label": "Role name",
        "name": "name",
        "type": "String"
    }, {
        "label": "A description of this role",
        "name": "description",
        "type": "String"
    }],
    "role": {
        "authorizations": [],
        "description": "Basic administration",
        "href": "/api/role/v1/roles/basic",
        "name": "basic"
    }
}
```

创建角色

此命令可创建新角色。

表 46 创建新角色属性

属性	类型	说明
name	string	新角色的名称（必需）
clone	string	要克隆原始属性的角色的名称（可选）
description	string	角色描述（必需）

请求示例：

```
POST /api/role/v1/roles HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 71

{"name":"role_workflow", "description":"Role to run workflows"}
```

结果示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 143
Location: /api/role/v1/roles/role_workflow

{
    "role": {
        "authorizations": [],
        "description": "Role to run workflows",
        "href": "/api/role/v1/roles/role_workflow",
```

```
        "name": "role_workflow"
    }
}
```

修改角色

可在创建角色后修改角色属性。

请求示例：

```
PUT /api/role/v1/roles/role_workflow HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 54

{"description":"Role allowing user to run workflows!"}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 158

 {
    "role": {
        "authorizations": [],
        "description": "Role allowing user to run workflows!",
        "href": "/api/role/v1/roles/role_workflow",
        "name": "role_workflow"
    }
}
```

撤销角色

从所有用户中撤销角色。

请求示例：

```
PUT /api/role/v1/role_worksheets/revoke HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 0
```

删除角色

从系统中删除角色。如果仍然将角色分配给一个或多个用户，请将 ?confirm=true 添加到 DELETE 命令中。

请求示例：

```
DELETE /api/role/v1/roles/rola?confirm=true HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

结果示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

列出角色授权

列出选定角色的授权。

请求示例：

```
GET /api/role/v1/roles/role_workflow/authorizations HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
{
    "authorizations": [{
        "allow_modify": false,
        "allow_read": true,
        "auth": "auth-000",
        "href": "/api/role/v1/roles/role_workflow/authorizations/auth-000",
        "owner": "*",
        "scope": "workflow",
        "uuid": "*"
    }]
}
```

创建角色授权

创建新的角色授权。输入属性与 CLI 中定义的属性相同。每个授权都具有所定义的 scope 属性。可根据此输入范围设置其他属性。范围值包括：

```
ad          cluster    keystore   role       stmf       user
alert       dataset    nas        schema     svc        workflow
appliance   hardware   net        stat       update     worksheet
```

请求示例：

```
POST /api/role/v1/roles/role_workflow/authorizations HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 41
{"scope": "workflow", "allow_read": true}
```

结果示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 171
Location: /api/role/v1/roles/role_workflow/authorizations/auth-000

{
    "auth": {
        "allow_modify": false,
        "allow_read": true,
        "href": "/api/role/v1/roles/role_workflow/authorizations/auth-000",
        "owner": "*",
        "scope": "workflow",
        "uuid": "*"
    }
}
```

修改角色授权

可修改角色授权属性。

请求示例：

```
PUT /api/role/v1/roles/role_workflow/authorizations/auth-000 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 29

{"allow_modify": true}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 171

{
    "auth": {
        "allow_modify": true,
        "allow_read": true,
        "href": "/api/role/v1/roles/role_workflow/authorizations/auth-000",
        "owner": "*",
        "scope": "workflow",
```

```
        "uuid": "*"
    }
}
```

删除角色授权

删除角色授权。

请求示例：

```
DELETE /api/role/v1/roles/role_workflow/authorizations/auth-000 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

结果示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

RESTful API SAN 服务

RESTful API SAN 服务允许您将设备连接到存储区域网络 (Storage Area Network, SAN)。

SAN 概述

SAN 具有以下基本组件：

- 访问网络存储的客户机
- 提供网络存储的存储设备
- 将客户机链接到存储的网络

不管网络上使用的是何种协议，这三个组件都保持不变。在某些情况下，网络甚至可能是启动器和目标之间的一条线缆，但是大多数情况下，涉及某些类型的交换。RESTful API SAN 服务管理各个受支持协议的四种 SAN 资源类型：

- **启动器**－能够启动 SCSI 会话、发送 SCSI 命令和 I/O 请求的应用程序或生产系统端点。启动器也通过唯一寻址方法进行标识。
- **启动器组**－一组启动器。启动器组与逻辑单元号 (Logical Unit Numbers, LUN) 关联后，仅该组中的启动器可以访问该 LUN。
- **目标**－一个存储系统端点，提供一个服务以处理来自启动器的 SCSI 命令和 I/O 请求。目标由存储系统管理员创建，并通过唯一寻址方法标识。在配置后，目标将包含零个或更多个逻辑单元。
- **目标组**－一组目标。LUN 针对一个特定目标组中的所有目标导出。

SAN 启动器

以下是用于管理 SAN 启动器的命令。

这些命令使用以下 URI 参数：

protocol 启动器的 NAS 协议：`fc`、`iscsi` 或 `srp`

initiator 启动器的 IQN、WWN 或 EUI

表 47 启动器命令

请求	附加到路径 /san/v1.0	说明
GET	/*protocol*/initiators	针对给定协议 (`fc`, `iscsi`, `srp`) 对象列出所有 SAN 启动器
GET	/*protocol*/initiators/*initiator*	针对给定协议 (`fc`, `iscsi`, `srp`) 属性获取指定的 SAN 启动器
POST	/*protocol*/initiators	针对给定协议 (`fc`, `iscsi`, `srp`) 创建新的 SAN 启动器
PUT	/*protocol*/initiators/*initiator*	针对给定协议 (`fc`, `iscsi`, `srp`) 对象修改指定的 SAN 启动器
DELETE	/*protocol*/initiators/*initiator*	销毁指定的启动器对象

许多启动器命令使用下表中列出的属性作为返回值。创建和修改命令也使用这些属性作为输入值。

表 48 启动器属性

属性	协议	说明
`alias`	all	此启动器的别名
`initiator`	`fc`	此启动器的端口全局名称 (world wide name, WWN)
`iqn`	`iscsi`	此启动器的 iSCSI 限定名称
`chapuser`	`iscsi`	质询握手 `auth` 协议 (Challenge Handshake Auth Protocol, CHAP) 用户名
`chapsecret`	`iscsi`	质询握手 `auth` 协议 (Challenge Handshake Auth Protocol, CHAP) 密钥
`initiator`	`srp`	扩展唯一标识符 (Extended Unique Identifier, EUI)

列出启动器

列出在指定协议类型的设备上配置的所有启动器。响应正文包含名为 "initiators" 的启动器属性数组（使用 JSON 格式）。

用来列出 iSCSI 启动器的请求示例：

```
GET /api/san/v1/iscsi/initiators HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json

{
    "initiators": [{
        "alias": "init-02",
        "href": "/api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:02:02",
        "initiator": "iqn.zfs-storage.example.com.sun:02:02",
        "chapsecret": "",
        "chapuser": ""
    },{
        "alias": "init-01",
        "initiator": "iqn.zfs-storage.example.com.sun:02:01",
        "href": "/api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:02:01",
        "chapsecret": "",
        "chapuser": ""
    }]
}
```

获取启动器详细信息

列出单个 iSCSI 启动器的详细信息。响应正文包含作为名为 "initiator" 的对象的 iSCSI 启动器属性（使用 JSON 格式）。

请求示例：

```
GET /api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:02:01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "initiator": {
        "alias": "init-01",
        "href": "/api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:02:01"
        "initiator": "iqn.zfs-storage.example.com.sun:02:01",
        "chapsecret": "",
        "chapuser": ""
    }
}
```

创建启动器

创建新的 iSCSI 启动器。您必须提供 iSCSI 限定名 (iSCSI Qualified Name, IQN)。请求正文包含 iSCSI 启动器属性（使用 JSON 格式）。响应包含 HTTP 头中新 iSCSI 启动器的位置 URI，以及响应成功时返回的状态码 201 (Created)。响应正文包含作为名为 "initiator" 的对象的 iSCSI 启动器属性（使用 JSON 格式）。

请求示例：

```
POST /api/san/v1.0/iscsi/initiators HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "initiator": "iqn.zfs-storage.example.com.sun:02:02",
    "alias":"init-02"
}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 181
X-Zfssa-San-Api: 1.0
Location: /api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:02:02

{
    "initiator": {
        "alias": "init-02",
        "href": "/api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:02:02",
        "initiator": "iqn.zfs-storage.example.com.sun:02:02",
        "chapsecret": "",
        "chapuser": ""
    }
}
```

修改启动器

此命令可修改现有启动器。请求正文包含应修改的启动器属性（使用 JSON 格式）。URI 中提供此启动器的 IQN。成功后，将返回 HTTP 状态 202 (Accepted)。响应正文包含作为名为 `initiator` 的对象的新 iSCSI 启动器属性（使用 JSON 格式）。

请求示例：

```
PUT /api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:01 HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "alias":"init-01-secure",
    "chapuser":"admin4",
    "chapsecret":"secret"
}
```

响应示例：

```
HTTP/1.1 202 Accepted
Content-Length: 167
Content-Type: application/json
X-Zfs-Sa-Nas-Api: 1.0

{
    "initiator": {
        "alias": "init-01-secure",
        "href": "/api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:01",
```

```
        "iqn": "iqn.zfs-storage.example.com.sun:1",
        "chapsecret": "secret",
        "chapuser": "admin4"
    }
}
```

删除启动器

从设备中删除启动器。

请求示例：

```
DELETE /api/san/v1/iscsi/initiators/iqn.zfs-storage.example.com.sun:01 HTTP/1.1
Host: zfs-storage.example.com:215
```

成功删除后，将返回 HTTP 代码 204 (No Content)：

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

启动器组

iSCSI 启动器命令用于管理设备上的 iSCSI 启动器和 iSCSI 启动器组。下表中列出了可用命令。

这些命令使用以下 URI 参数：

protocol 启动器的 NAS 协议：`fc`、`iscsi` 或 `srp`

name 启动器组的名称

每个启动器组都有 `name` 属性和 `initiators` 属性，后者包含启动器组中的启动器的列表。

表 49 启动器组命令

请求	附加到路径 /san/v1.0	说明
GET	/*protocol*/initiator-groups	针对给定协议 (`fc`, `iscsi`, `srp`) 对象列出所有 SAN 启动器组
GET	/*protocol*/initiator-groups/*name*	针对给定协议 (`fc`, `iscsi`, `srp`) 属性获取指定的 SAN 启动器组
POST	/*protocol*/initiator-groups	针对给定协议 (`fc`, `iscsi`, `srp`) 创建新的 SAN 启动器组
PUT	/*protocol*/initiator-groups/*name*	针对给定协议 (`fc`, `iscsi`, `srp`) 对象修改指定的 SAN 启动器组

请求	附加到路径 /san/v1.0	说明
DELETE	/*protocol*/initiator-groups/*name*	销毁指定的名称对象

列出启动器组

列出所有可用的 iSCSI 启动器组。成功后，将返回 HTTP 状态 200 (OK)，并且正文包含属性名为 "groups" 的 JSON 对象，此对象包含启动器组对象的数组。

请求示例：

```
GET /api/san/v1/iscsi/initiator-groups HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "groups": [{
        "href": "/san/v1/iscsi/initiator-groups/p1-initiators-0",
        "initiators": ["iqn.zfs-storage.example.com.sun:0"],
        "name": "p1-initiators-0"
    }, {
        "href": "/san/v1/iscsi/initiator-groups/p1-initiators-1",
        "initiators": ["iqn.zfs-storage.example.com.sun:1"],
        "name": "p1-initiators-1"
    }]
}
```

获取启动器组详细信息

从单个 iSCSI 启动器组中获取详细信息。可根据列出启动器组命令中返回的 href 属性访问此组。

请求示例：

```
GET /api/san/v1/iscsi/initiator-groups/test-group HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "group": {
        "href": "/api/san/v1/iscsi/initiator-groups/test-group"
        "initiators": ["iqn.zfs-storage.example.com.sun:02:01"],
        "name": "test-group"
```

```
    }
}
```

创建启动器组

创建无成员的 iSCSI 启动器组。请求正文包含带单个 name 参数的 JSON 对象，此参数包含组名称。

表 50 启动器组创建属性

属性	类型	说明
name	string	启动器组的名称
initiators	array	现有启动器 IQN 属性的数组

请求示例：

```
POST /api/san/v1/iscsi/initiator-groups HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Content-Length: 64
Accept: application/json

{
   "name":"group-01",
   "initiators": ["iqn.zfs-storage.example.com.sun:02"]
}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /api/san/v1/iscsi/initiator-groups/test-group

{
    "group": {
        "href": "/api/san/v1/iscsi/initiator-groups/test-group",
        "initiators": ["iqn.zfs-storage.example.com.sun:02"],
        "name": "group-01"
    }
}
```

删除启动器组

从设备中删除启动器组。

请求示例：

```
DELETE /api/san/v1.0/iscsi/initiator-groups/group-01 HTTP/1.1
Host: zfs-storage.example.com:215
```

成功删除后将返回 HTTP 状态 204 (No Content)：

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

目标

iSCSI 目标命令用于管理 iSCSI 目标和 iSCSI 目标组。下表中列出了可用命令。

这些目标命令使用以下 URI 参数：

protocol SAN 协议：`fc`、`iscsi` 或 `srp`

target 目标 ID：IQN、WWN 或 EUI

表 51 目标命令

请求	附加到路径 /san/v1.0	说明
GET	/*protocol*/targets	针对给定协议 (`fc`, `iscsi`, `srp`) 对象列出所有 SAN 目标
GET	/*protocol*/targets/*target*	针对给定协议 (`fc`, `iscsi`, `srp`) 属性获取指定的 SAN 目标
POST	/*protocol*/targets	针对给定协议 (`fc`, `iscsi`, `srp`) 创建新的 SAN 目标
PUT	/*protocol*/targets/*target*	针对给定协议 (`fc`, `iscsi`, `srp`) 对象修改指定的 SAN 目标
DELETE	/*protocol*/targets/*target*	销毁指定的目标对象

获取目标命令将返回目标属性。创建目标命令和修改目标命令使用下表中列出的属性作为输入。

表 52 目标输入属性

属性	协议	说明
`alias`	iscsi	简单的人工可读名称
`iqn`	iscsi	iSCSI 限定名称
`state`	iscsi	iSCSI 目标的状态（"online"、"offline"）
`auth`	iscsi	可选验证类型（"none"、"chap"）
`targetchapuser`	iscsi	可选 CHAP 用户验证
`targetchapsecret`	iscsi	可选 CHAP 密钥验证
`interfaces`	iscsi	目标可用的网络接口的列表
`wwn`	fc	此目标的全局名称
`port`	fc	此端口的物理位置

属性	协议	说明
mode	fc	此端口的模式（启动器或目标）
speed	fc	此端口的协商速率
discovered_ports	fc	已发现的远程启动器端口的数量
alias	srp	SRP 目标的别名
eui	srp	此目标的扩展唯一标识符

以下属性用于获取 iSCSI 目标组信息。

表 53 目标组属性

属性	类型	说明
protocol	string	目标组协议：FC、iSCSI 或 SRP
name	string	iSCSI 目标组名称
targets	array	iSCSI 目标 IQN 组成员的列表

列出目标

列出设备上可用的指定协议的所有 SAN 目标。

请求示例：

```
GET /api/san/v1/iscsi/targets HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1337

{
    "size": 7,
     "targets": [{
        "alias": "tst.volumes.py.12866.target",
        "href": "/api/san/v1/iscsi/targets/iqn.zfs-storage.example.com.sun:02:
            72b6fa9a-96c4-e511-db19-aadb9bac2052",
        "iqn": "iqn.zfs-storage.example.com.sun:02:72b6fa9a-96c4-
            e511-db19-aadb9bac2052",
        ...
    }, {
        "alias": "tst.volumes.py.96238.target",
         "href": "/api/san/v1/iscsi/targets/iqn.zfs-storage.example.com.sun:02:
            31d26d2e-6aa0-6054-fe58-8b1fb508b008",
        "iqn": "iqn.zfs-storage.example.com.sun:31d26d2e-6aa0-6054-fe58-8b1fb508b008",
        ...
    }
    ...]
```

```
}
```

获取目标详细信息

从单个目标中获取属性。可使用 "iqn" 属性或使用 "alias=*alias*" 选择目标。

请求示例：

```
GET /api/san/v1/iscsi/targets/alias=test-target HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 251

{
    "target": {
        "alias": "test-target",
        "auth": "none",
        "href": "/api/san/v1/iscsi/targets/alias=test-target",
        "interfaces": ["ixgbe0"],
        "iqn": "iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008",
        "targetchapsecret": "",
        "targetchapuser": ""
    }
}
```

创建目标

创建新的目标。请求正文包含一个带有 `name` 属性的 JSON 对象，此属性是新的 iSCSI 目标组的名称。

请求示例：

```
POST /api/san/v1/iscsi/targets HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
Content-Type: application/json
Content-Length: 23
Accept: application/json

{"alias": "test-target"}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 233
X-Zfssa-San-Api: 1.0
Location: /api/san/v1/iscsi/targets/iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-
fe58-8b1fb508b008
```

```
{
    "target": {
        "href": "/api/san/v1/iscsi/targets/iqn.zfs-
storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008",
        "alias": "test-target",
        "iqn": "iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008",
        "auth": "none",
        "targetchapuser": "",
        "targetchapsecret": "",
        "interfaces": ["ixgbe0"]
    }
}
```

修改目标

修改现有 iSCSI 目标。请求正文包含带有已修改的 iSCSI 目标属性的 JSON 对象。成功后，将返回 HTTP 状态 202 (Accepted)。响应正文包含在 JSON 对象中编码的目标的结果 iSCSI 目标属性。

请求示例：

```
PUT /api/san/v1/iscsi/targets/alias=test-target HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
Host: zfs-storage.example.com
Content-Type: application/json
Content-Length: 54
Accept: application/json

{"targetchapsecret":"secret", "auth":"chap",
 "targetchapuser":"admin5"}
```

响应示例：

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 189
X-Zfssa-San-Api: 1.0

{
    "target": {
        "href": "/api/san/v1/iscsi/targets/alias=test-target",
        "auth": "chap",
        "targetchapsecret": "secret",
        "alias": "test-target",
        "iqn": "iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008",
        "targetchapuser": "admin5",
        "interfaces": ["ixgbe0"]
    }
}
```

删除目标

从系统中删除 SAN 目标。

请求示例：

```
DELETE /api/san/v1/iscsi/targets/iqn.zfs-storage.example.com.sun:02:e7e688b1 HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
```

成功删除后，将返回 HTTP 代码 204 (No Content)：

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

目标组

目标组是目标的集合。下表中列出了目标组命令。

目标组命令使用以下 URI 参数：

protocol 启动器的 NAS 协议：`fc`、`iscsi` 或 `srp`

target-group 目标组的名称

表 54 目标组命令

请求	附加到路径 /san/v1.0	说明
GET	/*protocol*/target-groups	针对给定协议（`fc`、`iscsi` 或 `srp`）对象列出所有 SAN 目标组
GET	/*protocol*/target-groups/*target-group*	针对给定协议（`fc`、`iscsi` 或 `srp`）属性获取指定的 SAN 目标组
POST	/*protocol*/target-groups	针对给定协议（`fc`、`iscsi` 或 `srp`）创建新的 SAN 目标组
PUT	/*protocol*/target-groups/*target-group*	针对给定协议（`fc`、`iscsi` 或 `srp`）对象修改指定的 SAN 目标组
DELETE	/*protocol*/target-groups/*target-group*	销毁指定的目标组对象

列出目标组

列出设备上所有可用的目标组。成功后，将返回 HTTP 状态 200 (OK)，并且正文包含属性名为 `groups` 的 JSON 对象，此对象包含目标组对象的数组。

请求示例：

```
GET /api/san/v1/iscsi/target-groups
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 237

{
    "groups": [{
        "href": "/api/san/v1/iscsi/target-groups/test-group",
        "name": "test-group",
        "targets": [
            "iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008"
        ]
    }, {
        "href": "/api/san/v1/iscsi/target-groups/alt-group",
        ...
    }]
}
```

获取目标组

获取单个目标组。此请求使用作为目标组名称的单个 URI 参数。响应正文包含名为 `group` 的 JSON 对象属性，此对象属性包含目标组属性。

请求示例：

```
GET /api/san/v1/iscsi/target-groups/test-group
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE=
Accept: application/json
```

响应示例：

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

{
    "group": {
        "href": "/api/san/v1/iscsi/target-groups/test-group",
        "name": "test-group",
        "targets": [
            "iqn.zfs-storage.example.com.sun:02:0d5a0ed8-44b6-49f8-a594-872bf787ca5a"]
    }
}
```

创建目标组

创建新的 iSCSI 目标组。请求正文是带有单个 `name` 属性的 JSON 对象，此属性是此新组的名称。

请求示例：

```
POST /api/san/v1/iscsi/target-groups HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcd123MWE
Accept: application/json
```

```
Content-Type: application/json
Content-Length: 97

{"name":"test-group",
 "targets": ["iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008"]}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 154
X-Zfssa-San-Api: 1.0
Location: /api/san/v1/iscsi/target-groups/test-group

{
    "group": {
        "href": "/api/san/v1/iscsi/target-groups/test-group",
        "name": "test-group",
        "targets": [
            "iqn.zfs-storage.example.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1fb508b008"]
    }
}
```

删除目标组

删除现有目标组。

请求示例：

```
DELETE /api/nas/v1.0/iscsi/target-groups/test-group
```

成功删除后将返回 HTTP 状态 204 (No Content)：

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

服务命令

服务 RESTful API 用于列出和管理设备上运行的软件服务。

服务命令

以下服务命令可用。

表 55 服务命令

请求	附加到路径 /service/v1	说明
GET	仅使用 /service/v1	列出服务命令
GET	/services	列出所有服务
GET	/services/*service*	获取指定服务的配置和状态
PUT	/services/*service*	修改指定服务的配置
PUT	/services/*service*/enable	启用指定的服务
PUT	/services/*service*/disable	禁用指定的服务

列出服务

此命令返回存储设备上可用的可配置服务及其启用状态的列表。命令成功执行后，将返回 HTTP 状态 200 (OK)。

请求示例：

```
GET /api/service/v1/services HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
X-Zfssa-Service-Api: 1.0

{
    "services": [{
```

```
    "<status>": "disabled",
    "href": "/api/service/v1/services/ad",
    "name": "ad"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/smb",
    "log": {
        "href": "/api/log/v1/logs/network-smb:default",
        "size": 2
    },
    "name": "smb"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/dns",
    "log": {
        "href": "/api/log/v1/logs/network-dns-client:default",
        "size": 4
    },
    "name": "dns"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/dynrouting",
    "log": {
        "href": "/api/log/v1/logs/network-routing-route:default",
        "size": 81
    },
    "name": "dynrouting"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/ftp",
    "log": {
        "href": "/api/log/v1/logs/network-ftp:proftpd",
        "size": 40
    },
    "name": "ftp"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/http",
    "name": "http"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/identity",
    "log": {
        "href": "/api/log/v1/logs/system-identity:node",
        "size": 4
    },
    "name": "identity"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/idmap",
    "log": {
        "href": "/api/log/v1/logs/system-idmap:default",
        "size": 15
    },
    "name": "idmap"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/ipmp",
    "log": {
        "href": "/api/log/v1/logs/network-ipmp:default",
        "size": 3
    },
    "name": "ipmp"
}, {
```

```
    "<status>": "online",
    "href": "/api/service/v1/services/iscsi",
    "log": {
        "href": "/api/log/v1/logs/network-iscsi-target:default",
        "size": 3
    },
    "name": "iscsi"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/ldap",
    "name": "ldap"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/ndmp",
    "log": {
        "href": "/api/log/v1/logs/system-ndmpd:default",
        "size": 11
    },
    "name": "ndmp"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/nfs",
    "log": {
        "href": "/api/log/v1/logs/appliance-kit-nfsconf:default",
        "size": 6
    },
    "name": "nfs"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/nis",
    "log": {
        "href": "/api/log/v1/logs/network-nis-domain:default",
        "size": 3
    },
    "name": "nis"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/ntp",
    "name": "ntp"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/replication",
    "name": "replication"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/rest",
    "log": {
        "href": "/api/log/v1/logs/appliance-kit-akrestd:default",
        "size": 10
    },
    "name": "rest"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/scrk",
    "name": "scrk"
}, {
    "<status>": "disabled",
    "href": "/api/service/v1/services/sftp",
    "name": "sftp"
}, {
    "<status>": "online",
    "href": "/api/service/v1/services/shadow",
    "name": "shadow"
}, {
```

```
            "<status>": "online",
            "href": "/api/service/v1/services/smtp",
            "log": {
                "href": "/api/log/v1/logs/network-smtp:sendmail",
                "size": 6
            },
            "name": "smtp"
        }, {
            "<status>": "disabled",
            "href": "/api/service/v1/services/snmp",
            "name": "snmp"
        }, {
            "<status>": "disabled",
            "href": "/api/service/v1/services/srp",
            "name": "srp"
        }, {
            "<status>": "online",
            "href": "/api/service/v1/services/ssh",
            "log": {
                "href": "/api/log/v1/logs/network-ssh:default",
                "size": 3
            },
            "name": "ssh"
        }, {
            "<status>": "disabled",
            "href": "/api/service/v1/services/syslog",
            "name": "syslog"
        }, {
            "<status>": "online",
            "href": "/api/service/v1/services/tags",
            "name": "tags"
        }, {
            "<status>": "disabled",
            "href": "/api/service/v1/services/tftp",
            "name": "tftp"
        }, {
            "<status>": "disabled",
            "href": "/api/service/v1/services/vscan",
            "log": {
                "href": "/api/log/v1/logs/vscan",
                "size": 0
            },
            "name": "vscan"
        }]
}
```

获取服务

此命令从单个服务中获取详细信息，其中包括此服务的状态和配置。

请求示例：

```
GET /api/service/v1/services/ndmp HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json

{
    "service": {
        "cram_md5_password": "",
        "cram_md5_username": "",
        "dar_support": true,
        "default_pools": [],
        "drive_type": "sysv",
        "href": "/api/service/v1/services/ndmp",
        "ignore_ctime": false,
        "name": "ndmp",
        "restore_fullpath": false,
        "status": "online",
        "tcp_port": 10000,
        "version": 4,
        "zfs_force_override": "off",
        "zfs_token_support": false
    }
}
```

更改服务状态

此命令更改给定服务的状态。将使用以下 URI 参数：

service 服务的名称

state 新服务状态：`enable` 或 `disable`

请求示例：

```
PUT /api/service/v1/services/replication/enable HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功响应会返回 HTTP 状态 202 (Accepted)。也可通过向服务发送 JSON 请求来启用或禁用此服务。

使用 JSON 的请求示例：

```
PUT /api/service/v1/services/replication HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
Content-Type: application/json
Content-Length: 22

{"<status>": "enable"}
```

要禁用此服务，请发送以下 JSON：

```
{"<status>": "disable"}
```

修改服务配置

可通过发送 PUT 请求以及在其头中定义的新属性值来修改指定服务的配置属性。一些服务可能具有子资源，您也可以根据子资源中定义的 href 来修改这些服务。

请求示例：

```
PUT /api/service/v1/services/sftp HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json

{"port": 218}
```

成功响应会返回 HTTP 状态 202 (Accepted)：

```
HTTP/1.1 202 Accepted
Content-Length: 162
Content-Type: application/json; charset=utf-8
X-Zfssa-Service-Api: 1.0

{
    "service": {
        "<status>": "disabled",
        "href": "/api/service/v1/services/sftp",
        "keys": [],
        "listen_port": 218,
        "logging_verbosity": "INFO",
        "root_login": false
    }
}
```

服务资源

一些服务具有子资源。查看针对各个服务或针对服务命令列表返回的数据，以了解有哪些子资源可用。

表 56 服务子资源命令

请求	路径	说明
GET	/services/*service*/*resource*	列出服务子资源
PUT	/services/*service*/*resource*/*href*	修改子资源
POST	/services/*service*/*resource*	创建新的子资源
DELETE	/services/*service*/*resource*/*href*	销毁子资源

这些命令使用的模式与其他 RESTful API 命令相同，其中，GET 用于列出或获取指定的子资源类型，POST 用于创建新的子资源类型，PUT 用于修改子资源，DELETE 用于销毁指定的子资源。

有关每个子资源及其可用的属性和命令的列表，请参见 CLI "configuration services"（配置服务）文档。

RESTful API 存储服务

RESTful API 存储服务用于查看配置以及管理存储池、项目、文件系统和 LUN 的各个方面。它还可管理快照和复制。

存储池操作

在 Oracle ZFS Storage Appliance 中，NAS 在池中进行配置，这些池在所有 LUN 和文件系统共享资源中都具有相同的数据冗余特性。在这个版本的 NAS API 中，池操作用于获取设备存储配置。

表 57 存储池命令

请求	附加到路径 /api/storage/v1	说明
GET	/pools	列出所有存储池
GET	/pools/*pool*	获取存储池详细信息
POST	/pools	配置新的存储池
PUT	/pools/*pool*	向池中添加存储或从中移除存储
PUT	/pools/*pool*/scrub	对指定的池启动数据清理
DELETE	/pools/*pool*/scrub	对指定的池停止任何数据清理
DELETE	/pools/*pool*	取消配置指定的存储池

列出池

此命令列出系统上所有存储池的属性。命令成功执行后，将返回 HTTP 状态 200 (OK)。HTTP 正文包含描述每个池的 JSON 对象的列表。下表显示了这些属性的名称。

注 - 不支持 `depth` 查询参数和 `match_`*property-name*`=`*value* 查询参数。

表 58 存储池属性

属性	类型	说明
`pool`	string	目标池名称

属性	类型	说明
profile	string	数据设备配置文件
state	string	池状态：online、offline、exported
asn	string	拥有此池的设备的序列号
peer	string	群集系统中对等节点的 ASN
owner	string	拥有此池的系统的主机名

请求示例：

```
GET /api/storage/v1/pools HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "pools": [{
        "profile": "mirror3",
            "name": "m1",
            "peer": "00000000-0000-0000-0000-000000000000",
        "state": "online",
        "owner": "zfs-storage",
        "asn": "2f4aeeb3-b670-ee53-e0a7-d8e0ae410749"
    }, {
        "profile": "raidz1",
        "name": "r1",
        "peer": "00000000-0000-0000-0000-000000000000",
        "state": "online",
        "owner": "zfs-storage",
        "asn": "2f4aeeb3-b670-ee53-e0a7-d8e0ae410749"
    }]
}
```

获取池

此命令返回单个存储池中的属性以及此池的存储使用情况信息。命令成功执行后，将返回 HTTP 状态 200 (OK)。

请求示例：

```
GET /api/storage/v1/pools/p1 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

```
{
    "pool": {
        "profile": "raidz1",
        "name": "p1",
        "usage": {
            "available": 57454799311352.0,
            "compression": 1.0,
            "dedupratio": 672791,
            "free": 57454799311352.0,
            "total": 74732430950400.0,
            "usage_child_reservation": 0.0,
            "usage_data": 16011663438848.0,
            "usage_metasize": 0.0,
            "usage_metaused": 0.0,
            "usage_replication": 1693675705344.0,
            "usage_reservation": 0.0,
            "usage_snapshots": 123913627136.0,
            "usage_total": 17829252771328.0,
            "used": 17829252771328.0
        },
        "peer": "00000000-0000-0000-0000-000000000000",
        "state": "online",
        "owner": "admin1",
        "asn": "2f4aeeb3-b670-ee53-e0a7-d8e0ae410749"
    }
}
```

配置池

配置池。有关创建池所需的参数，请参见 CLI 配置存储命令。可执行用于创建池的模拟运行请求，该请求会返回可用的属性名称和值。这是通过将 `props` 查询参数属性设置为 `true` 来执行的。

请求示例：

```
POST /api/storage/v1/pools?props=true HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbfsMWE=
Content-Type: application/json
Accept: application/json

{
    "name": "p1",
}
```

响应示例：

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

"props": [{
        "choices": ["custom" ],
        "label": "Chassis 0",
        "name": "0",
        "type": "ChooseOne"
    }, {
        "choices": ["custom"],
```

```
        "label": "Chassis 1",
        "name": "1",
        "type": "ChooseOne"
    }, {
        "choices": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
        "label": "Chassis 1 data",
        "name": "1-data",
        "type": "ChooseOne"
    }, {
        "choices": ["mirror", "mirror3", "raidz1",
            "raidz2", "raidz3_max", "stripe"],
        "label": "Data Profile",
        "name": "profile",
        "type": "ChooseOne"
    }]
}
```

请求示例（创建使用机箱 [1] 中 8 个磁盘的池）：

```
POST /api/storage/v1/pools HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbfsMWE=
Content-Type: application/json
Accept: application/json

{
    "name": "p1",
    "profile": "stripe",
    "1-data": 8
}
```

响应示例：

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

{
    "pool": {
        "asn": "314d252e-c42b-e844-dab1-a3bca680b563",
        "errors": [],
        "name": "p1",
        "owner": "zfs-storage",
        "peer": "00000000-0000-0000-0000-000000000000",
        "profile": "stripe",
        "status": "online",
        "usage": {
            "available": 1194000466944.0,
            "dedupratio": 100,
            "total": 1194000908288.0,
            "used": 441344.0
        }
    }
}
```

向池中添加存储

此命令类似于创建或配置池。添加存储会将其他存储设备添加到现有池中。发送 href *pool*/add，并在正文中包含要添加到池中的存储设备和所需的设备数量。

请求示例：

```
PUT /api/storage/v1/pools/p1/add HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbfsMWE=
Content-Type: application/json
Accept: application/json

{
    "2-data": 8
}
```

响应示例：

```
HTTP/1.1 202 Accepted
```

从池中移除存储

此命令类似于向池中添加存储。移除存储会从现有池中移除高速缓存和日志存储设备。发送 href *pool*/remove，并在正文中包含要从池中移除的存储设备和所需的设备类型、机箱编号和设备数量。

请求示例：

```
PUT /api/storage/v1/pools/p1/remove HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbfsMWE=
Content-Type: application/json
Accept: application/json
{
    "0-cache" : 2
}
```

响应示例：

```
HTTP/1.1 202 Accepted
```

要显示可移除设备的数量，请将查询参数 `props` 设置为 `true`。

请求示例：

```
PUT /api/storage/v1/pools/p1/remove?props=true HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbfsMWE=
Content-Type: application/json
Accept: application/json
```

响应示例：

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

{
      "props": [
          {
```

```
                "choices": [
                    "0",
                    "1",
                    "2"
                ],
                "type": "ChooseOne",
                "name": "0-cache",
                "label": "Chassis 0 cache"
            },
            {
                "choices": [
                    "0",
                    "1",
                    "2"
                ],
                "type": "ChooseOne",
                "name": "1-log",
                "label": "Chassis 1 log"
            }
        ]
}
```

池清理

发送 *pool*/scrub PUT 或 DELETE 请求可分别启动池清理操作或停止正在运行的清理作业。有关详细信息，请参见 CLI 命令 "configuration storage scrub"。

取消配置池

此命令可将池从系统中删除。

请求删除池：

```
DELETE /api/storage/v1/pools/p1 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbfsMWE=
```

响应示例：

```
HTTP/1.0 204 No Content
Date: Fri, 02 Aug 2013 22:31:06 GMT
X-Zfssa-Nas-Api: 1.0
Content-Length: 0
```

项目操作

所有项目操作都可限定到给定池。在所有项目中运行的命令会将 /projects 附加到 URI，而在单个项目中运行的命令则附加 /projects/*project*。

表 59 项目命令

请求	附加到路径 /api/storage/v1	说明
GET	/projects	列出所有项目
GET	/pools/*pool*/projects	列出项目
GET	/pools/*pool*/projects?snaps=true	列出所有项目（包括快照）
GET	/pools/*pool*/projects/*project*	获取项目详细信息
POST	/pools/*pool*/projects	创建项目
PUT	/pools/*pool*/projects/*project*	修改项目
DELETE	/pools/*pool*/projects/*project*	销毁项目
GET	/pools/*pool*/projects/*project*/usage/groups	获取项目组的使用情况
GET	/pools/*pool*/projects/*project*/usage/groups/*group*	获取指定组的项目使用情况
GET	/pools/*pool*/projects/*project*/usage/users	获取项目用户的使用情况
GET	/pools/*pool*/projects/*project*/usage/users/*user*	获取指定的用户的项目使用情况

下表显示了项目资源中的可编辑属性的列表。

表 60 项目属性

属性	类型	说明
`aclinherit`	string	ACL 继承行为（"discard"、"noallow"、"restricted"、"passthrough"、"passthrough-x"、"passthrough-mode-preserve"）
`aclmode`	string	模式更改时的 ACL 行为（"discard"、"mask"、"passthrough"）
`atime`	boolean	读取时更新访问时间标志
`canonical_name`	string	规范名称
`checksum`	string	块校验和（"fletcher2"、"fletcher4"、"sha256"）
`compression`	string	数据压缩设置（"off"、"lzjb"、"gzip-2"、"gzip"、"gzip-9"）
`copies`	number	其他复制副本的数量
`creation`	datetime	项目（或 LUN、文件系统）创建的日期和时间
`dedup`	boolean	重复数据删除标志
`default_group`	string	项目默认文件系统组："other"
`default_permissions`	string	项目默认文件系统权限 "700"
`default_sparse`	boolean	项目默认 LUN 稀疏数据标志
`default_user`	string	项目默认文件系统用户："nobody"
`default_volblocksize`	number	项目默认 LUN 块大小：8192
`default_volsize`	number	项目默认 LUN 大小
`exported`	boolean	已导出标志
`logbias`	string	同步写入偏向（"latency"、"throughput"）

属性	类型	说明
mountpoint	string	共享挂载点默认值 "/export/proj-01"
name	string	项目名称
nbmand	boolean	非阻塞强制性锁定标志
nodestroy	boolean	阻止销毁标志
quota	number	项目配额大小（单位为字节）
origin	string	克隆来源
pool	string	池名称
readonly	boolean	仅在此属性设置为 true 时才读取数据
recordsize	string	数据库记录大小为 "128k"
reservation	number	数据预留空间大小
rstchown	boolean	限制所有权更改标志
secondarycache	string	二级高速缓存使用情况（"all"、"metadata"、"none"）
sharedav	string	HTTP 共享资源（"off"、"rw"、"ro"）
shareftp	string	FTP 共享资源（"off"、"rw"、"ro"）
sharenfs	string	NFS 共享资源（"off"、"on"、"ro"、"rw"）
sharesftp	string	SFTP 共享资源（"off"、"rw"、"ro"）
sharesmb	string	SMB/CIFS 共享资源（"off"、"rw"、"ro"）
sharetftp	string	TFTP 共享资源（"off"、"rw"、"ro"）
snapdir	string	.zfs/snaphsot 可见性（"hidden"、"visible"）
snaplabel	string	调度的快照标签
vscan	boolean	病毒扫描标志

列出项目

此命令列出给定池中的所有项目。此请求使用作为存储池名称的单个 URI 参数。每个返回的项目都包含上述可修改属性的列表以及池名称、创建时间、装入状态、复制操作和数据使用情况。

注 - 不支持 `depth` 查询参数和 `match_`*property-name*`=`*value* 查询参数。

查询参数：`filter`——一个简单的字符串匹配过滤器，要求项目中的属性在其值中包含相同的过滤器字符串。

请求示例：

```
GET /api/storage/v1/pools/p1/projects HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功执行 `get` 后，将返回 HTTP 代码 200 (OK) 以及项目属性数组（使用 JSON 格式）。

结果示例：

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

{
    "projects": [{
        "name": "proj-01",
        ...
    }, {
        "name": "proj-02",
        ...
    }
}
```

系统也支持所有池中的所有项目的列表；URI 将仅包含 /projects 路径。

用于获取其属性中包含 backup 的所有项目的请求示例：

```
GET /projects?filter=backup HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

获取项目属性

此命令列出给定池中单个项目的属性。成功执行 get 后，将返回 HTTP 代码 200 (OK) 以及项目属性（使用 JSON 格式）。

用于列出 zfs-storage-1 池中名为 proj-01 的项目的请求示例：

```
GET /api/storage/v1/pools/p1/projects/proj-01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "project": {
        "default_volblocksize": 8192.0,
        "logbias": "latency",
        "creation": "20130411T20:02:35",
        "nodestroy": false,
        "dedup": false,
        "sharenfs": "on",
        "sharesmb": "off",
        "default_permissions": "700",
        "mountpoint": "/export",
        "snaplabel": "",
        "id": "042919bb-0882-d903-0000-000000000000",
        "readonly": false,
        "rrsrc_actions": [],
        "compression": "off",
        "sharetftp": "",
        "default_sparse": false,
```

```
        "snapdir": "hidden",
        "aclmode": "discard",
        "copies": 1,
        "aclinherit": "restricted",
        "shareftp": "",
        "canonical_name": "zfs-storage-1/local/default",
        "recordsize": 131072.0,
        "usage": {
            "available": 1758424767306.0,
            "loading": false,
            "quota": 0.0,
            "snapshots": 0.0,
           "compressratio": 100.0,
           "child_reservation": 0.0,
           "reservation": 0.0,
           "total": 45960.0,
            "data": 45960.0
        },
        "default_volsize": 0.0,
        "secondarycache": "all",
        "collection": "local",
        "exported": true,
        "vscan": false,
        "reservation": 0.0,
        "atime": true,
        "pool": "p1",
        "default_user": "nobody",
        "name": "default",
        "checksum": "fletcher4",
        "default_group": "other",
        "sharesftp": "",
        "nbmand": false,
        "sharedav": "",
        "rstchown": true
    }
}
```

创建项目

创建项目命令将在给定存储池中创建使用给定名称的项目。此请求使用作为存储池名称的单个 URI 参数。将返回具有默认属性的新项目。

JSON 正文请求参数：

- `name`－必须提供项目名称以创建项目。
- Project properties－任何项目属性都可设置为新项目的初始值。

创建名为 `proj-01` 的项目的请求示例：

```
POST /api/storage/v1/pools/p1/projects HTTP/1.1
Hosta: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "name": "proj-01",
    "sharenfs": "ro"
}
```

成功创建后，将返回 HTTP 状态 201 (Created)，且位置头包含新项目的 URI。正文包含所有项目属性（使用 JSON 格式）。

结果示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://zfs-storage.example.com:215
          /pools/p1/projects/proj-01

{
    "project": {
        "name": "proj-01",
        "href": "/api/storage/v1/pools/p1/projects/proj-01",
        "mountpoint": "/export/acme/zfs-storage-1",
        ...
    }
}
```

修改项目

此修改项目命令用于更改现有项目的属性。将使用以下 URI 参数：

pool 存储池名称

project 项目名称

请求参数－Project Properties－任何项目属性都可设置为新项目的初始值。

将项目名称从 `proj-01` 更改为 `new-name` 的请求示例：

```
POST /api/storage/v1/pools/p1/projects/proj-01 HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "name": "new-name",
    "sharenfs": "rw",
    "compression": "gzip-9"
}
```

成功响应后，将返回 HTTP 状态 202 (Accepted) 并列出所有项目属性。

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /api/storage/v1/pools/p1/projects/new-name

{
    "project": {
        "name": "new-name",
        "sharenfs": "rw",
        "compression: "gzip-9",
         ...
```

```
    }
}
```

删除项目

此删除项目命令用于删除给定池中的单个项目。将使用以下 URI 参数：

pool 存储池名称

project 项目名称

如需监视在接受了延迟更新异步数据集删除 (OS8.7.0) 之后，要在存储池中回收的空间量，请针对 pools/*pool* 输入 GET 命令。记下 `async_destroy_reclaim_space` 属性的空间量。操作完成时会显示 0（零）。

请求示例：

```
DELETE /api/storage/v1/pools/p1/projects/proj-01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

项目使用情况

获取请求项目使用情况资源可用于按用户或用户组获取项目的使用情况数据。

文件系统操作

文件系统操作命令可列出和管理文件系统共享资源。所有命令都可限定于给定的存储池或项目。

service_uri`/pools/`*pool*`/project/`*project*

表 61 文件系统命令

请求	附加到路径 /api/storage/v1	说明
GET	/filesystems	列出所有文件系统
GET	/pools/*pool*/projects/*project*/filesystems	列出指定的文件系统
GET	/pools/*pool*/projects/*project*/filesystems?snaps=true	列出所有文件系统（包括快照）
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*	获取文件系统详细信息
POST	/pools/*pool*/projects/*project*/filesystems	创建文件系统
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*	修改文件系统
DELETE	/pools/*pool*/projects/*project*/filesystems/*filesystem*	销毁文件系统

请求	附加到路径 /api/storage/v1	说明
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups	获取文件系统组使用情况
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups/*group*	获取指定的组的文件系统使用情况
POST	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups	创建文件系统组配额
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups/*name*	修改文件系统组配额
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users	获取文件系统用户使用情况
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users/*user*	获取指定的用户的文件系统使用情况
POST	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users	创建文件系统用户配额
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users/*name*	修改文件系统用户配额
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/shadow/errors	列出影子迁移错误

每个文件系统都包含此项目中的属性，并具有以下特定于文件系统的属性。

表 62 文件系统属性

属性	类型	说明
`casesensitivity`	string	"Case Sensitivity" 设置：`mixed`、`sensitive` 或 `insensitive`
`group`	string	组名
`normalization`	string	标准化
`permissions`	string	文件系统权限
`project`	string	项目名称
`quota_snap`	boolean	指示配额中包括快照的标志
`reservation_snap`	boolean	指示预留空间中包括快照的标志
`shadow`	string	数据迁移源
`errors`	string	数据迁移错误
`sharesmb_name`	string	SMB 共享资源的名称
`source`	object	项目继承属性
`usage`	object	文件系统使用情况信息
`user`	string	拥有共享资源的用户名
`utf8only`	boolean	拒绝非 UTF-8 的标志

列出文件系统

列出文件系统命令显示给定池或项目中的所有文件系统。

注 - 不支持 `depth` 查询参数和 `match_`*property-name*=*value* 查询参数。

查询参数：`filter`——一个简单的字符串匹配过滤器，要求项目中的属性在其值中包含相同的过滤器字符串。

列出文件系统命令使用以下 URI 参数：

pool 存储池名称

project 项目名称

请求示例：

```
GET /api/storage/v1/pools/p1/projects/proj-01/filesystems HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功执行请求后，将返回 HTTP 状态 200 (OK) 以及系统文件属性数组（使用 JSON 格式）。

结果示例：

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

{
    "filesystems": [{
        "name": "filesystem-01",
        "project": "proj-01",
        "pool": "p1",
        ...
    }, {
        "name": "filesystem-02",
        "project": "proj-01",
        "pool": "p1",
        ...
    }]
}
```

系统也支持所有池和项目中的所有文件系统的列表。在这种情况下，URI 将是 /api/storage/v1/filesystems。

获取所有文件系统并以 "abcd" 字符串作为其部分属性的请求示例：

```
GET /api/storage/v1/filesystems?filter=abcd HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

获取文件系统

获取文件系统命令返回给定池或项目中的单个文件系统的属性。将使用以下 URI 参数：

pool 存储池名称

project 项目名称

filesystem 文件系统名称

列出名为 `proj-01` 的项目的请求示例：

```
GET /api/storage/v1/pools/p1/projects/proj-01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功执行 `get` 后，将返回 HTTP 状态 200 (OK) 以及文件系统属性（使用 JSON 格式）。

响应示例：

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

{
    "filesystem": {
        "logbias": "latency",
        "creation": "20130423T21:30:34",
        "nodestroy": false,
        "dedup": false,
        "sharenfs": "on",
        "sharesmb": "off",
        "mountpoint": "/export/mnt1",
        "snaplabel": "",
        "id": "424ca2ec-b3fa-df86-0000-000000000000",
        "readonly": false,
        "rrsrc_actions": [],
        "compression": "off",
        "sharetftp": "",
        "source": {
            "logbias": "default",
            "dedup": "default",
            "sharenfs": "inherited",
            "sharesmb": "off",
            "mountpoint": "inherited",
            "rrsrc_actions": "local",
            "compression": "default",
            "sharetftp": "inherited",
            "snapdir": "default",
            "aclmode": "default",
            "copies": "default",
            "aclinherit": "default",
            "shareftp": "inherited",
            "readonly": "default",
            "secondarycache": "default",
            "exported": "inherited",
            "vscan": "default",
            "reservation": "local",
            "atime": "default",
            "recordsize": "default",
            "checksum": "inherited",
            "sharesftp": "inherited",
            "nbmand": "default",
            "rstchown": "default"
        },
        "snapdir": "hidden",
        "aclmode": "discard",
        "copies": 1,
        "aclinherit": "restricted",
        "shareftp": "",
        "canonical_name": "p1/local/default/mnt1",
        "recordsize": 131072.0,
        "usage": {
            "available": 880395477504.0,
            "loading": false,
            "quota": 0.0,
            "snapshots": 18432.0,
```

```
            "compressratio": 100.0,
            "reservation": 0.0,
            "total": 50176.0,
            "data": 31744.0
        },
        "secondarycache": "all",
        "collection": "local",
        "exported": true,
        "vscan": false,
        "reservation": 0.0,
        "shadow": "none",
        "atime": true,
        "pool": "p1",
        "quota_snap": true,
        "name": "mnt1",
        "checksum": "fletcher4",
        "project": "default",
        "sharesftp": "",
        "nbmand": false,
        "reservation_snap": true,
        "sharedav": "",
        "rstchown": true,
        "root_acl": {
             "owner@:cC:fd:deny",
             "everyone@:rw:fd:allow",
             "user:admin1:rw:allow",
        }
        "smbshareacl": {
             "owner@:cC:fd:deny",
             "everyone@:rw:fd:allow",
             "user:admin1:rw:allow",
        }
}
```

创建文件系统

创建文件系统命令可在给定存储池或项目中创建使用给定名称的文件系统。将返回带默认属性的新文件系统。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

filesystem 文件系统名称

请求参数：

- `name`－必须提供文件系统名称以创建新的文件系统。
- filesystem properties－文件系统属性或项目属性中列出的任何属性都可设置为初始值。

请求示例（创建名为 `share-01` 且由用户 `admin1` 拥有的文件系统）：

```
POST /api/storage/v1/pools/p1/projects/proj-01/filesystems HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "name": "share-01",
    "root_user": "admin1"
}
```

成功创建后，将返回 HTTP 状态 201 (Created)，且位置头包含新文件系统的 URI。正文包含所有文件系统属性（使用 JSON 格式）。

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /api/storage/v1/pools/p1/projects/proj-01/filesystems/share-01

{
    "filesystem": {
        "name": "share-01",
        "pool": "p1",
        "collection": "local",
        "project": "proj-01",
        "root_user": "admin1"
        ...
    }
}
```

修改文件系统

修改文件系统命令用于更改现有文件系统的属性。成功响应后，将返回 HTTP 状态 202 (Accepted) 并列出所有文件系统属性。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

filesystem 文件系统名称

请求参数：filesystem properties－可修改任何文件系统或项目属性。

将文件系统名称从 `share-01` 更改为 `new-name` 并将所有者更改为 `nobody` 的请求示例：

```
PUT /api/storage/v1/pools/p1/projects/proj-01/filesystems/share-01 HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "name": "new-name",
    "root_user": "nobody",
```

```
}
```

响应示例：

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Location: http://zfs-storage.example.com:215/pools/p1/projects/proj-01/filesystems/share-01

{
    "filesystem": {
        "name": "new-name",
        "pool": "p1",
        "collection": "local",
        "project": "proj-01",
        "root_user": "nobody"
        ...
    }
}
```

删除文件系统

删除文件系统命令用于删除给定池或项目中的单个文件系统。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

filesystem 文件系统名称

如需监视要在存储池中回收的空间量，请针对 pools/*pool* 输入 GET 命令。记下 `async_destroy_reclaim_space` 属性的空间量。操作完成时会显示 0（零）。

请求示例：

```
DELETE /api/storage/v1/pools/p1/projects/proj-01/filesystems/share-01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功删除后将返回 HTTP 状态 204 (No Content)。

响应示例：

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

文件系统配额和使用情况

可使用 `POST` 或 `PUT` 请求分别创建或修改用户或组配额。对文件系统使用资源的 `GET` 请求可用于按用户或组获取项目的使用情况数据。

LUN 操作

所有 LUN 或卷操作都可限定于给定的池或项目。以下 LUN 命令可用。

表 63 卷命令

请求	附加到路径 /api/storage/v1	说明
GET	/luns	列出所有 LUN
GET	/pools/*pool*/projects/*project*/luns	列出 LUN
GET	/pools/*pool*/projects/*project*/luns?snaps=true	列出所有 LUN（包括快照）
GET	/pools/*pool*/projects/*project*/luns/*lun*	获取 LUN 详细信息
POST	/pools/*pool*/projects/*project*/luns	创建 LUN
PUT	/pools/*pool*/projects/*project*/luns/*lun*	修改 LUN
DELETE	/pools/*pool*/projects/*project*/luns/*lun*	销毁 LUN

下表列出了 LUN 属性。卷也可继承或覆盖项目属性。

表 64 卷属性

属性	类型	说明
`assignednumber`	数字或数字列表	分配的 LU 编号。如果提供给多个启动器组，则类型为数字列表。 如果提供给多个启动器组，则 `assignednumber` 和 `initiatorgroups` 的排序一致。例如，`assignednumber` 列表中的第一个项目与 `initiatorgroups` 列表中的第一个项目相关。
`fixednumber`	boolean	将 LU 编号固定为当前值的标志。
`initiatorgroups`	字符串列表	启动器组。 如果将该 LUN 提供给多个启动器组，则 `assignednumber` 和 `initiatorgroups` 的排序一致。例如，`assignednumber` 列表中的第一个项目与 `initiatorgroups` 列表中的第一个项目相关。
`lunguid`	string	STMF GUID。
`lunumber`	数字或字符串	LU 编号。一个数字或 `auto`。
`project`	string	项目名称（不可变）。
`source`	object	列出属性源（`local` 或 `inherited`）。
`sparse`	boolean	启用瘦置备的标志。
`status`	string	逻辑单元状态（`online` 或 `offline`）。
`targetgroup`	string	目标组
`usage`	object	列出 LUN 使用情况统计信息
`volblocksize`	number	卷块大小

属性	类型	说明
volsize	number	卷大小
writecache	boolean	启用写入缓存的标志

可以从项目继承某些属性。源对象列出这些属性中的每一个，并标识属性是 LUN 的本地属性还是从项目继承的。默认情况下，项目会继承这些属性。在设置后，它们将成为 LUN 的本地属性。源对象不可变。要将源状态更改回 inherited，可取消设置属性。

取消设置压缩的 JSON 请求示例：

```
{"unset": ["compression"]}
```

列出 LUN

列出 LUN 命令会返回给定池或项目中可用 LUN 的列表。

注 - 不支持 `depth` 查询参数和 `match_`*property-name*`=`*value* 查询参数。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

filesystem 文件系统名称

列出 `proj-01` 项目中 LUN 的请求示例：

```
GET /api/storage/v1/pools/p1/projects/proj-01/luns HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功执行 `get` 后，将返回 HTTP 状态 200 (OK) 以及 LUN 属性（使用 JSON 格式）。

响应示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "luns": [{
        "id": "fa4ac6fb-0bcc-d2e3-0000-000000000000",
        "name": "vol-01"
        ...
    }, {
         "id": "690ae407-7c4d-b5d2-0000-000000000000",
         "name": "vol-01",
         ....
    }]
```

```
}
```

获取 LUN

获取 LUN 命令会返回给定池或项目中的单个 LUN 属性。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

lun LUN 名称

请求示例（获取名为 "vol-01" 的 LUN）：

```
GET /api/storage/v1/pools/p1/projects/proj-01/lun/vol-01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功执行 get 后，将返回 HTTP 状态 200 (OK) 以及 LUN 属性（使用 JSON 格式）。

响应示例：

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

{
    "lun": {
        "logbias": "latency",
        "creation": "20130423T21:31:17",
        "nodestroy": false,
        "dedup": false,
        "rrsrc_actions": [],
        "id": "e3045406-319b-cf7a-0000-000000000000",
        "writecache": false,
        "compression": "off",
        "copies": 1,
        "stmfguid": "600144F0D8E0AE4100005176FDA60001",
        "source": {
            "compression": "default",
            "checksum": "inherited",
            "logbias": "default",
            "dedup": "default",
            "copies": "default",
            "exported": "inherited",
            "rrsrc_actions": "inherited",
            "secondarycache": "default"
        },
        "canonical_name": "p1/local/default/disk1",
        "snaplabel": "",
        "usage": {
            "available": 881469214720.0,
            "loading": false,
            "snapshots": 0.0,
```

```
            "compressratio": 100.0,
            "total": 1073758208.0,
            "data": 1073758208.0
        },
        "secondarycache": "all",
        "collection": "local",
        "exported": true,
        "volsize": 1073741824.0,
        "pool": "p1",
        "volblocksize": 8192,
        "checksum": "fletcher4",
        "project": "default",
        "sparse": false
    }
}
```

创建新的 LUN

此命令可创建新的 LUN。您必须为新的 LUN 提供大小或克隆源。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

请求参数：

- name－必须提供 LUN 名称以创建新的 LUN。
- Volume properties－LUN 属性或项目属性中列出的任何属性都可设置为初始值。

请求示例：

```
POST /api/storage/v1/pools/p1/projects/proj-01/luns HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json

Request JSON:
{
        name : "vol-001",           // Volume name (required)

        size : 500000,              // New Volume size
        blocksize : 8192,           // New Volume block size
        sparse : true,              // New Volume sparse data flag

        initiatorgroup : 'default', // Initiator group name
        targetgroup : 'default',    // Target group name
        lunnumber : 'auto',         // Volume LUN number
        status : 'online',          // Initial Status ('online', 'offline')
        fixednumber : false,

        "source": {
            "snapshot_id" : "76b8950a-8594-4e5b-8dce-0dfa9c696358",
            "snapshot": "/pool-001/local/proj-001/snap-001"
        }
}
```

成功创建后，将返回 HTTP 状态 201 (Created)，且位置头包含新 LUN 的 URI。正文包含所有 LUN 属性（使用 JSON 格式）。

结果示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://zfs-storage.example.com:215
          /pools/p1/projects/proj-01/luns/vol-001

{
    "lun": {
        "name": "vol-001",
        ...
    }
}
```

修改 LUN

修改 LUN 命令用于更改现有 LUN 的属性。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

lun LUN 名称

请求参数：volume properties－可修改任何 LUN 或项目属性。

将 LUN 名称从 `vol-01` 更改为 `new-name` 的请求示例：

```
POST /api/storage/v1/pools/p1/projects/proj-01/luns/vol-01 HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
    "name": "new-name",
}
```

成功响应后，将返回 HTTP 状态 202 (Accepted) 并列出所有 LUN 属性。

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /api/storage/v1/pools/p1/projects/proj-01/luns/new-name

{
    "lun": {
        "name": "new-name",
```

```
        "pool": "p1",
        "collection": "local",
        "project": "proj-01",
        ...
    }
}
```

删除 Lun

删除 LUN 命令用于删除给定池或项目中的单个 LUN。

将使用以下 URI 参数：

pool 存储池名称

project 项目名称

lun LUN 名称

如需监视要在存储池中回收的空间量，请针对 pools/*pool* 输入 GET 命令。记下 `async_destroy_reclaim_space` 属性的空间量。操作完成时会显示 0（零）。

请求示例：

```
DELETE /pools/p1/projects/proj-01/luns/lun-01 HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

成功执行 `get` 后将返回 HTTP 状态 204 (No Content)。

响应示例：

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

快照和克隆操作

所有快照操作都可限定于给定的池或项目。快照操作也可限定于文件系统或 LUN 级别。

- 所有基于项目的快照操作的 URI 都以 `/api/storage/v1/pools/`*pool*`/projects/`*project* 开头
- 所有基于文件系统的快照操作的 URI 都以 `/api/storage/v1/pools/`*pool*`/projects/`*project*`/filesystems/`*filesystem* 开头
- 所有基于 LUN 的快照操作的 URI 都以 `/api/storage/v1/pools/`*pool*`/projects/`*project*`/luns/`*lun* 开头

表 65 快照和克隆命令

请求	附加到路径 /api/storage/v1	说明
GET	/snapshots	列出所有本地快照
GET	/pools/*pool*/projects?snaps=true	列出所有项目（包括快照）
GET	/pools/*pool*/projects/*project*/filesystems?snaps=true	列出所有文件系统（包括快照）
GET	/pools/*pool*/projects/*project*/luns?snaps=true	列出所有 LUN（包括快照）
GET	/pools/*pool*/projects/*project*/snapshots	列出项目的所有快照
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots	列出文件系统的所有快照
GET	/pools/*pool*/projects/*project*/luns/*lun*/snapshots	列出 LUN 的所有快照
GET	/pools/*pool*/projects/*project*/snapshots/*snapshot*	获取项目快照详细信息
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	获取文件系统快照详细信息
GET	/pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot*	获取 LUN 快照详细信息
POST	/pools/*pool*/projects/*project*/snapshots	创建项目快照
POST	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots	创建文件系统快照
POST	/pools/*pool*/projects/*project*/luns/*lun*/snapshots	创建 LUN 快照
PUT	/pools/*pool*/projects/*project*/snapshots/*snapshot*	修改项目快照
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	修改文件系统快照
PUT	/pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot*	修改 LUN 快照
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/clone	克隆文件系统快照
PUT	/pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot*/clone	克隆 LUN 快照
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/rollback	将数据回滚到给定文件系统快照
PUT	/pools/*pool*/projects/*project*/lun/*lun*/snapshots/*snapshot*/rollback	将数据回滚到给定 LUN 快照
DELETE	/pools/*pool*/projects/*project*/snapshots/*snapshot*	销毁项目快照
DELETE	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	销毁文件系统快照
DELETE	/pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot*	销毁 LUN 快照
GET	/pools/*pool*/projects/*project*/snapshots/*snapshot*/dependents	列出项目快照相关项
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/dependents	列出文件系统快照相关项
GET	/pools/*pool*/projects/*project*/lun/*lun*/snapshots/*snapshot*/dependents	列出 LUN 快照相关项
POST	/pools/*pool*/projects/*project*/automatic	创建新项目自动快照对象
POST	/pools/*pool*/projects/*project*/automatic?convert=true	创建新项目自动快照对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。
GET	/pools/*pool*/projects/*project*/automatic/*automatic*	获取指定的项目自动快照属性
GET	/pools/*pool*/projects/*project*/automatic	列出所有项目自动快照对象
PUT	/pools/*pool*/projects/*project*/automatic/*automatic*	修改指定的项目自动快照对象
PUT	/pools/*pool*/projects/*project*/automatic/*automatic*?convert=true	修改指定的项目自动快照调度对象。自动生成的、不符合新调度的现有快照将转换为手动快照。

请求	附加到路径 /api/storage/v1	说明
		排除 convert 属性会导致销毁自动生成的现有快照。
DELETE	/pools/*pool*/projects/*project*/automatic/*automatic*	销毁指定的自动对象
DELETE	/pools/*pool*/projects/*project*/automatic/*automatic*?convert=true	销毁指定的自动快照调度对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。
POST	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic	创建新文件系统自动快照对象
POST	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic?convert=true	创建新文件系统自动快照对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*	获取指定的文件系统自动快照属性
GET	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic	列出所有文件系统自动快照对象
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*	修改指定的文件系统自动快照对象
PUT	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*?convert=true	修改指定的文件系统自动快照调度对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。
DELETE	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*	销毁指定的自动快照调度对象。
DELETE	/pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*?convert=true	销毁指定的文件系统自动快照调度对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。
POST	/pools/*pool*/projects/*project*/luns/*lun*/automatic	创建新的 LUN 自动快照
POST	/pools/*pool*/projects/*project*/luns/*lun*/automatic?convert=true	创建新的 LUN 自动快照调度。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。
GET	/pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*	获取指定的 LUN 自动快照属性
GET	/pools/*pool*/projects/*project*/luns/*lun*/automatic	列出所有 LUN 自动快照对象
PUT	/pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*	修改指定的 LUN 自动快照对象
PUT	/pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*?convert=true	修改指定的 LUN 自动快照调度对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。

请求	附加到路径 /api/storage/v1	说明
DELETE	/pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*	销毁指定的 LUN 自动对象
DELETE	/pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*?convert=true	销毁指定的 LUN 自动快照调度对象。自动生成的、不符合新调度的现有快照将转换为手动快照。 排除 convert 属性会导致销毁自动生成的现有快照。

列出快照

列出设备上的可用快照。列表可包含项目快照、文件系统快照或 LUN 快照，具体取决于请求 URI。

表 66 列出快照命令表单

命令	附加到路径 /api/storage/v1/pools/*pool*/projects/*project*
列出项目快照	/snapshots
列出文件系统快照	/filesystems/*share*/snapshots
列出 LUN 快照	/lun/*share*/snapshots

请求示例：

```
GET /api/storage/v1/pools/p1/projects/default/snapshots
Accept: application/json
```

响应示例：

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

{
    "snapshots": [{
        "id": "3fbbcccf-d058-4502-8844-6feeffdf4cb5",
        "display_name": "snap-001",
        "display_description": "Daily backup",
        "volume_id": "521752a6-acf6-4b2d-bc7a-119f9148cd8c",
        "status": "available",
        "size": 30,
        "created_at": "2012-02-29T03:50:07Z"
    }, {
        "id": "e479997c-650b-40a4-9dfe-77655818b0d2",
        "display_name": "snap-002",
        "display_description": "Weekly backup",
        "volume_id": "76b8950a-8594-4e5b-8dce-0dfa9c696358",
        "status": "available",
        "size": 25,
        "created_at": "2012-03-19T01:52:47Z"
    }]
}
```

获取快照

查看有关单个快照的所有信息。成功后返回 HTTP 状态 200 (OK)。

请求示例：

```
GET /api/storage/v1/pools/p1/projects/default/snapshots/snap-001
Accept: application/json
```

响应示例：

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

{
    "snapshot": {
        "id": "3fbbcccf-d058-4502-8844-6feeffdf4cb5",
        "display_name": "snap-001",
        "display_description": "Daily backup",
        "volume_id": "521752a6-acf6-4b2d-bc7a-119f9148cd8c",
        "status": "available",
        "size": 30,
        "created_at": "2012-02-29T03:50:07Z"
    }
}
```

创建快照

创建快照命令为项目、文件系统或 LUN 创建快照。

- 创建项目快照－`POST /pools/`*pool*`/projects/`*project*`/snapshots`
- 创建文件系统快照
  －`POST /pools/`*pool*`/projects/`*project*`/filesystems/`*share*`/snapshots`
- 创建卷快照－`POST /pools/`*pool*`/projects/`*project*`/luns/`*lun*`/snapshots`

请求示例：

```
POST /api/storage/v1/pools/p1/projects/default/snapshots
Content-Type: application/json

{"name": "initial-backup"}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /pools/p1/projects/default/
snapshot/initial-backup

{
    "snapshot": {
        "name": "initial-backup",
```

```
        "numclones": 0,
        "creation": "20130610T21:00:49",
        "collection": "local",
        "project": "default",
        "canonical_name": "zfs-storage-1/local/default@initial-backup",
        "usage": {
            "unique": 0.0,
            "loading": false,
            "data": 145408.0
        },
        "type": "snapshot",
        "id": "a26abd24-e22b-62b2-0000-000000000000",
        "pool": "p1"
    }
}
```

重命名快照

重命名现有快照。

- **请求 URI**－"Snapshot"，当前快照名称
- **请求正文**－JSON 对象，其名称参数包含新快照名称

请求示例：

```
PUT /api/storage/v1/pools/p1/projects/default/snapshots/initial-snapshot
Content-Type: application/json
Accept: application/json

{"name": "old-snapshot"}
```

响应示例：

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Location: /pools/p1/projects/default/snapshot/initial-backup
```

克隆快照

根据现有快照创建新的文件系统或 LUN。

将使用以下 URI 参数：

pool 源池名称

project 源项目名称

filesystem 文件系统快照的源共享资源名称

lun LUN 快照的源共享资源名称

snapshot 源快照名称

克隆文件系统：

```
PUT /pools/pool/projects/project/filesystems/share/snapshots/snapshot/clone
```

克隆卷：

```
PUT /pools/pool/projects/project/luns/lun/snapshots/snapshot/clone
```

请求正文包含带以下属性的 JSON 对象。

表 67 克隆快照属性

属性	类型	说明
pool	string	目标克隆池名称
project	string	目标克隆项目名称
lun	string	LUN 快照的目标 LUN 名称

请求示例：

```
PUT /api/storage/v1/pools/p1/projects/default/filesystems/fs01/
    snapshots/snap01/clone

{"project":"rest", "share":"snap01clone01", "compression": "gzip-9"}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Length: 2035
X-Zfssa-Storage-Api: 1.0
Location: /api/storage/v1/pools/p1/projects/rest/filesystem/snap01clone01
Content-Type: application/json; charset=utf-8

{
    "filesystem": {
        "origin": {
            "project": "default",
            "share": "fs01",
            "snapshot": "snap01",
            "pool": "p1",
            "collection": "local"
        },
        "href": "/api/storage/v1/pools/p1/projects/rest/filesystems/snap01clone01",
        "mountpoint": "/export/snap01clone01",
        "compression": "gzip-9",
        "source": {
            "compression": "local",
            ...
        },
        ...
       "canonical_name": "zfs-storage-1/local/rest/snap01clone01"
   }
```

```
}
```

回滚快照

回滚快照会导致源文件系统或 LUN 被修改回拍摄快照时的状态。成功响应后会返回 HTTP 状态 202 (Accepted) 以及快照属性（使用 JSON 格式）。

将使用以下 URI 参数：

pool	源池名称
project	源项目名称
filesystem	文件系统快照的源文件系统名称
lun	LUN 快照的源 LUN 名称
snapshot	源快照名称

回滚文件系统快照：

```
PUT /pools/pool/projects/project/filesystems/filesystem/snapshots/snapshot/rollback
```

回滚 LUN 快照：

```
PUT /pools/pool/projects/project/luns/lun/snapshots/snapshot/rollback
```

请求示例：

```
PUT /api/storage/v1/pools/p1/projects/default/filesystems/fs-01
    /snapshots/initial-backup/rollback
```

响应示例：

```
HTTP/1.1 202 Accepted
Location: /pools/p1/projects/default/filesystems/fs-01/snapshot/fs-01-initial-clone
Content-Type: application/json

{
    "snapshot": {
        "name": "fs-01-initial-clone",
        "numclones": 0,
        "creation": "20130610T21:00:49",
        "filesystem": "fs-01",
        "collection": "local",
        "project": "default",
        "canonical_name": "zfs-storage-1/local/default/
               fs-01@fs-01-initial-clone",
        "usage": {
            "unique": 0.0,
            "loading": false,
            "data": 31744.0
        },
```

```
        "type": "snapshot",
        "id": "5c9bda07-21c1-2238-0000-000000000000",
        "pool": "p1"
    }
}
```

删除快照

DELETE 快照命令可用于从系统中删除项目快照、文件系统快照或 LUN 快照。

将使用以下 URI 参数：

pool	源池名称
project	源项目名称
filesystem	源文件系统名称
lun	LUN 名称
snapshot	源快照名称

删除项目快照：

```
DELETE /api/storage/v1/pools/pool/projects/project/snapshots/snapshot
```

删除文件系统快照：

```
DELETE /api/storage/v1/pools/pool/projects/project/filesystems/filesystem/snapshots/snapshot
```

删除文件系统 LUN：

```
DELETE /api/storage/v1/pools/pool/projects/projectsnapshot
```

如果快照中存在 NDMP，则必须将 `?confirm=true` 添加到 DELETE 命令中。但是，请注意，这会对 NDMP 运行产生不利影响。有关更多信息，请参见《Oracle ZFS Storage Appliance 管理指南，发行版 OS8.8.0》 中的 “NDMP 配置”。

请求示例：

```
DELETE /pools/p1/projects/default/filesystems/fs-01/snapshots/initial-backup?confirm=true
```

未添加 `?confirm=true` 时的结果示例：

当快照中存在 NDMP 时，如果未添加 `?confirm=true`，则该命令将失败并输出以下内容（为了提高可读性，已人为地进行了断行）：

```
HTTP/1.1 409 Conflict

{"fault": {"message": "request requires confirm=true to complete (confirmation
```

```
needed for scripted command (scripted commands must be prefixed with \"confirm\"
to automatically confirm or \"deny\" to automatically deny) (encountered while
attempting to run command \"confirm destroy snap\"))", "code": 409, "name":
"ERR_CONFIRM_REQUIRED"}}
```

列出快照相关项

列出文件系统或卷的相关项。将使用以下 URI 参数：

pool	系统存储池名称
project	项目名称
filesystem	文件系统名称
lun	LUN 名称
snapshot	快照名称

列出文件系统相关项：

GET /api/storage/v1/pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/dependents

列出卷相关项：

GET /api/storage/v1/pools/*pool*/projects/*project*/lun/*lun*/snapshots/*snapshot*/dependents

请求示例：

```
GET /api/storage/v1/pools/p1/projects/default/filesystems/fs01/snapshots/snap01/dependents
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Storage-Api: 1.0
Content-Type: application/json; charset=utf-8
X-Zfssa-Api-Version: 1.0

{
    "dependents": [
        {
            "project": "rest",
            "href": "/api/storage/v1/pools/p1/projects/rest/filesystems/snap01clone01",
            "share": "snap01clone01"
        },
        {
            "project": "rest",
            "href": "/api/storage/v1/pools/p1/projects/rest/filesystems/snap01clone02",
            "share": "snap01clone02"
        },
        {
            "project": "rest",
            "href": "/api/storage/v1/pools/p1/projects/rest/filesystems/snap01clone03",
```

```
            "share": "snap01clone03"
        }
    ]
}
```

模式

管理定制模式属性。

表 68 模式命令

请求	附加到路径 /api/storage/v1	说明
GET	/schema	列出所有 NAS 模式属性对象
GET	/schema/*property*	获取指定的 NAS 模式属性属性
POST	/schema	创建新的 NAS 模式属性
PUT	/schema/*property*	修改指定的 NAS 模式属性对象
DELETE	/schema/*property*	删除指定的 NAS 模式属性对象

通过在定制属性名称中添加前缀 `custom:`，可在项目、文件系统和 LUN 中设置各个定制模式属性。

例如，以下 PUT 正文修改了名为 `priority` 的定制 `int` 属性：

```
{"custom:priority": 5}
```

表 69 模式参数

参数	说明
`property`	属性名称（不可变）
`description`	属性描述（针对浏览器界面）
`type`	类型（"String"、"Integer"、"PositiveInteger"、"Boolean"、"EmailAddress"、"Host"）

列出属性

列出模式属性。

请求示例：

```
GET /api/storage/v1/schema
```

结果示例：

```
{
    "properties": [{
        "description": "bob",
        "href": "/api/storage/v1/schema/bob",
        "property": "bob",
        "type": "String"
    },{
        "description": "boo",
        "href": "/api/storage/v1/schema/boo",
        "property": "boo",
        "type": "String"
    }]
}
```

获取属性

获取模式属性。

请求示例：

```
GET /api/storage/v1/schema/priority
```

结果示例：

```
{
    "property": {
        "description": "priority",
        "href": "/api/storage/v1/schema/priority",
        "property": "bob",
        "type": "Integer"
    }
}
```

创建属性

创建新的模式属性。

请求示例：

```
POST /api/storage/v1/schema HTTP/1.1
Host: zfs-storage.example.com:215
Content-Type: application/json
Content-Length: 64

{"property":"priority", "type":"Integer", "description":"Oh my"}
```

结果示例：

```
HTTP/1.1 201 Created
Content-Length: 89
X-Zfssa-Nas-Api: 1.0
Content-Type: application/json
Location: /api/storage/v1/schema/priority
```

```
{
    "property": {
        "href": "/api/storage/v1/schema",
        "type": "Integer",
        "description": "Oh my"
    }
}
```

修改属性

修改模式属性。

请求示例：

```
PUT /api/storage/v1/schema/priority

{"description":"My custom priority level"}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Nas-Api: 1.0
Content-Type: application/json
Content-Length: 90

{
    "property": {
        "href": "//api/storage/v1/schema/priority",
        "type": "Integer",
        "description": "My custom priority level"
    }
}
```

删除属性

此命令删除模式属性。

请求示例：

```
DELETE /api/storage/v1/schema/me HTTP/1.1
```

结果示例：

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

复制

远程复制为设备之间项目和共享资源的复制提供了便利。

注 - 复制是某些型号的 Oracle ZFS Storage Appliance 的一项许可功能，复制 RESTful API 可管理此功能。此服务是通过以下 URI 提供的：`https://`*hostname*`:215/api/storage/v1/replication`。有关许可证详细信息，请参阅 Oracle Software License Agreement (SLA) and Entitlement for Hardware Systems with Integrated Software Options 和此软件发行版的《Licensing Information User Manual》。

复制 RESTful API 管理以下资源：

- **复制服务**－用于管理复制任务的服务。
- **复制目标**－将接收和存储从另一对等设备（源）中复制的数据的对等设备。此术语也指设备上使得设备可以向另一设备进行复制的一个配置对象。
- **复制操作**－源设备上的一个配置对象，它指定了项目或共享资源、目标设备和策略选项（包括发送更新的频率、是否对网络上的数据进行加密，等等）。
- **复制数据包**－某个操作的目标端对应体；目标设备上的一个配置对象，它管理作为某个特定操作的一部分从特定源复制的数据。源设备上的每个操作都恰好与目标设备上的一个数据包相关联，反之亦然。丢失了任何一个对象都将要求创建新的操作/数据包对（和完整的复制更新）。

API 为复制操作和复制数据包提供复制操作。此服务 API 用于管理复制服务以及复制源和复制目标。

表 70 复制服务命令

请求	附加到路径 /api/service/v1/services	说明
GET	/replication	获取复制服务状态属性
PUT	/replication/enable	启用复制服务
PUT	/replication/disable	禁用复制服务

获取复制服务

获取复制服务的状态。

请求示例：

```
GET /api/service/v1/services/replication HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 131
X-Zfssa-Replication-Api: 1.0

{
```

```
    "service": {
        "status": "online",
        "href": "/service/v1/services/replication",
        "sources": [],
        "targets": []
    }
}
```

修改复制服务状态

可像修改任何其他服务一样修改复制服务状态。有关详细信息，请参见服务 RESTful API。

复制目标

下表显示了可用的复制目标命令。

表 71 复制目标命令

请求	附加到路径 /api/storage/v1	说明
POST	/replication/targets	创建新复制目标
GET	/replication/targets/*target*	获取指定的复制目标属性
GET	/replication/targets	列出所有复制目标对象
PUT	/replication/targets/*target*	修改指定的复制目标对象
DELETE	/replication/targets/*target*	销毁指定的目标对象

列出复制目标

列出系统上所有可用的复制目标。

请求示例：

```
GET /api/storage/v1/replication/targets HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529

{
        "targets": [{
```

```
            "address": "ipaddr-1",
            "label": "zfs-storage-1",
            "hostname": "ipaddr-2",
            "asn": "9d7a7543-ca83-68f5-a8fc-f818f65e1cfc",
            "actions": 0,
            "target": "target-000",
            "href": "/api/storage/v1/replication/targets/zfs-storage-1"
        }, {
            "address": "ipaddr-3",
            "label": "zfs-storage-2",
            "hostname": "ipaddr-4",
            "asn": "16a4c82c-26c1-4a50-e317-ac53181f2e86",
            "actions": 0,
            "target": "target-001",
            "href": "/api/storage/v1/replication/targets/zfs-storage-2"
        }]
}
```

获取复制目标

此命令列出单个复制目标的详细信息。目标通过其主机名访问。

请求示例：

```
GET /api/storage/v1/replication/targets/zfs-storage-1 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 337

{
        "target": {
            "href": "/api/storage/v1/replication/targets/zfs-storage-1",
            "address": "ipaddr-1",
            "label": "zfs-storage-1",
            "hostname": "ipaddr-2",
            "asn": "9d7a7543-ca83-68f5-a8fc-f818f65e1cfc",
            "actions": 0
        }
}
```

创建复制目标

为远程复制创建新的复制目标。

请求示例：

```
POST /api/replication/v1/targets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
```

```
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 54

{"hostname":"example", "root_password":"root-password", "label":"zfs-storage-3"}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Length: 135
Content-Type: application/json
Location: /service/v1/services/replication/targets/target-000
X-Zfssa-Replication-Api: 1.0

{
    "target": {
        "actions": 0,
        "address": "123.45.78.9:216",
        "asn": "fa5bf303-0dcb-e20d-ac92-cd129ccd2c81",
        "hostname": "example",
        "href": "/service/v1/services/replication/targets/target-000",
        "label": "zfs-storage-3"
    }
}
```

删除复制目标

此命令可删除现有的复制目标。

请求示例：

```
DELETE /service/v1/services/replication/targets/target-000 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

成功删除后将返回 HTTP 状态 204 (No Content)。

响应示例：

```
HTTP/1.1 204 No-Content
X-Zfssa-Replication-Api: 1.0
```

复制操作

复制操作可定义将数据复制到复制目标的规则。以下命令可管理复制操作。

使用平面操作接口

可以直接对设备发出管理复制操作的请求，而不指定项目或共享资源。

下表列出了用于管理复制操作的基本命令。

表 72 基本操作接口

请求	附加到路径 /api/storage/v1	说明
GET	/replication/actions	列出所有复制操作对象
GET	/replication/actions/*ra_id*	获取指定的复制操作属性
PUT	/replication/actions/*ra_id*	修改指定的复制操作对象
DELETE	/replication/actions/*ra_id*	删除指定的复制操作对象
PUT	/replication/actions/*ra_id*/sendupdate	开始选定的复制操作
PUT	/replication/actions/*ra_id*/cancelupdate	停止选定的复制操作

下表列出了用于管理复制操作调度的命令。

表 73 访问操作调度

请求	附加到路径 /api/storage/v1	说明
GET	/replication/actions/*ra_id*/schedules	列出所有复制操作调度对象
GET	/replication/actions/*ra_id*/schedules/*ra_schedule*	获取指定的复制操作调度属性
POST	/replication/actions/*ra_id*/schedules	创建新的复制操作调度
PUT	/replication/actions/*ra_id*/schedules/*ra_schedule*	修改指定的复制操作调度对象
DELETE	/replication/actions/*ra_id*/schedules/*ra_schedule*	删除指定的复制操作调度对象

下表列出了用于管理复制自动快照的命令。

注 - 不能通过以下命令访问在项目级复制操作内配置的共享资源级自动快照调度。项目级操作可以在多个共享资源中具有多个自动快照调度，此接口未提供明确识别所有组合的方式。

表 74 访问复制自动快照配置

请求	附加到路径 /api/storage/v1	说明
GET	/replication/actions/*ra_id*/autosnaps	检索所选复制操作的自动快照配置
GET	/replication/actions/*ra_id*/autosnaps/*autosnaps_id*	获取指定的复制操作自动快照对象
PUT	/replication/actions/*ra_id*/autosnaps	修改指定的复制操作自动快照属性
PUT	/replication/actions/*ra_id*/autosnaps/*autosnaps_id*	修改指定的复制操作自动快照对象
DELETE	/replication/actions/*ra_id*/autosnaps/*autosnaps_id*	删除指定的复制操作自动快照对象

下表列出了用于复制统计信息的命令。

表 75 访问复制操作统计信息

请求	附加到路径 /api/storage/v1	说明
GET	/replication/actions/*ra_id*/stats	检索所选复制操作的只读复制统计信息

项目、文件系统或 LUN 上下文中的复制操作

还可以在特定项目、文件系统或 LUN 上下文中发出管理复制操作的请求。

下表列出了用于管理复制操作的基本命令。

- 基于项目的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*

- 基于文件系统的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*`/filesystems/`*filesystem*

- 基于 LUN 的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*`/luns/`*lun*

将以下基本命令附加到上面列出的所需上下文 URI 来管理复制操作。

表 76 项目、文件系统或 LUN 基本复制操作接口

请求	附加到上面列出的项目、文件系统或 LUN URI	说明
GET	/replication/actions	列出所有复制操作对象
GET	/replication/actions/*ra_id*	获取指定的复制操作属性
POST	/replication/actions	创建新的复制操作
PUT	/replication/actions/*ra_id*	修改指定的复制操作对象
DELETE	/replication/actions/*ra_id*	删除指定的复制操作对象
PUT	/replication/actions/*ra_id*/sendupdate	开始选定的复制操作
PUT	`/replication/actions/`*ra_id*`/cancelupdate`	停止选定的复制操作

下表列出了用于管理复制调度的基本命令。

- 基于项目的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*

- 基于文件系统的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*`/filesystems/`*filesystem*

- 基于 LUN 的操作 URI 以下面的内容开头：

```
/api/storage/v1/pools/pool/projects/project/luns/lun
```

将以下基本命令附加到上面列出的所需上下文 URI 来管理复制调度。

表 77 项目、文件系统或 LUN 复制操作调度

请求	附加到上面列出的项目、文件系统或 LUN URI	说明
GET	/replication/actions/*ra_id*/schedules	列出所有复制操作调度对象
GET	/replication/actions/*ra_id*/schedules/*ra_schedule*	获取指定的复制操作调度属性
POST	/replication/actions/*ra_id*/schedules	创建新的复制操作调度
PUT	/replication/actions/*ra_id*/schedules/*ra_schedule*	修改指定的复制操作调度对象
DELETE	/replication/actions/*ra_id*/schedules/*ra_schedule*	删除指定的复制操作调度对象

下表列出了用于管理复制自动快照配置的基本命令。

- 基于项目的操作 URI 以下面的内容开头：

  ```
  /api/storage/v1/pools/pool/projects/project
  ```

- 基于文件系统的操作 URI 以下面的内容开头：

  ```
  /api/storage/v1/pools/pool/projects/project/filesystems/filesystem
  ```

- 基于 LUN 的操作 URI 以下面的内容开头：

  ```
  /api/storage/v1/pools/pool/projects/project/luns/lun
  ```

将以下基本命令附加到上面列出的所需上下文 URI 来管理复制自动快照配置。

注 - 不能通过以下基于项目的操作访问在项目级复制操作内配置的共享资源级自动快照调度。项目级操作可以在多个共享资源中具有多个自动快照调度，此接口未提供明确识别所有组合的方式。

表 78 项目、文件系统或 LUN 复制自动快照配置

请求	附加到上面列出的项目、文件系统或 LUN URI	说明
GET	/replication/actions/*ra_id*/autosnaps	检索项目/共享资源的所选复制操作的自动快照配置
GET	/replication/actions/*ra_id*/autosnaps/*autosnaps_id*	获取项目/共享资源的指定复制操作自动快照配置
POST	/replication/actions/*ra_id*/autosnaps	创建新项目/共享资源级复制操作自动快照对象
PUT	/replication/actions/*ra_id*/autosnaps	修改项目/共享资源的指定复制操作的目标自动快照保留策略。
PUT	/replication/actions/*ra_id*/autosnaps/*autosnaps_id*	修改指定的复制操作自动快照对象

请求	附加到上面列出的项目、文件系统或 LUN URI	说明
DELETE	/replication/actions/*ra_id*/autosnaps/*autosnaps_id*	删除指定的复制操作自动快照对象

下表列出了用于访问复制操作统计信息的基本命令。

- 基于项目的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*

- 基于文件系统的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*`/filesystems/`*filesystem*

- 基于 LUN 的操作 URI 以下面的内容开头：

  `/api/storage/v1/pools/`*pool*`/projects/`*project*`/luns/`*lun*

将以下基本命令附加到上面列出的所需上下文 URI 来管理复制操作统计信息。

表 79 访问复制操作统计信息

请求	附加到上面列出的项目、文件系统或 LUN URI	说明
GET	/replication/actions/*ra_id*/stats	检索所选复制操作的只读复制统计信息

列出复制操作

获取所有可用复制操作的列表。

请求示例：

```
GET /api/storage/v1/replication/actions HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529

{
    "actions": [{
        "href": ""
        ...
    }, {
        "href": "",
        ...
    }]
}
```

获取复制操作

获取复制操作状态命令会返回单个复制操作 ID 所指定的单个复制操作的状态。

请求示例：

```
GET /api/storage/v1/replication/actions/1438ed7f-aad3-c631-d869-9e85cd7f15b4 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529

{
    "action": {
        "average_throughput": 0.0,
        "bytes_sent": 0.0,
        "collection": "local",
        "compression": true,
        "continuous": false,
        "enabled": true,
        "estimated_size": 0.0,
        "estimated_time_left": 0.0,
        "href": "/api/storage/v1/replication/actions",
        "id": "8373d331-de60-e590-90e8-9ad69fcb4aec",
        "include_clone_origin_as_data": false,
        "include_snaps": true,
        "last_sync": "20130916T21:36:50",
        "last_try": "20130916T21:36:50",
        "max_bandwidth": 0,
        "pool": "p1",
        "project": "proj-01",
        "retain_user_snaps_on_target": false,
        "share": "fs1",
        "state": "sending",
        "target": "38094753-6c90-49ed-aa92-995a296d432a",
        "use_ssl": true
    }
}
```

请求示例：

以下复制操作响应显示了示例恢复点目标 (recovery point objective, RPO) 和相关副本滞后警告和警报。

```
GET /api/storage/v1/replication/actions HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Content-Type:application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529
```

```
{
    "action": {"id": "12d981c3-b098-c65a-e1e9-a6b8263a0f6a",
        "target_id": "4fd305ac-4af5-c34a-87c3-88203207305b",
        ...
        "replica_lag": "42:25:31",
        "recovery_point_objective": 0,
        "replica_lag_warning_alert": 0,
        "replica_lag_error_alert": 0,
        "replica_lag_over_warning_limit": false,
        "replica_lag_over_error_limit": false,
        "project": "default"
    }
}
```

创建复制操作

创建新的复制操作。

创建属性：

```
Initial values:
                          target = cleo
                         enabled = true
                      continuous = false
                   include_snaps = true
     retain_user_snaps_on_target = false
                           dedup = true
    include_clone_origin_as_data = false
                  max_bandwidth = unlimited
                     bytes_sent = 0
                  estimated_size = 0
            estimated_time_left = 0
             average_throughput = 0
                        use_ssl = true
                    compression = on
```

请求示例：

```
POST /api/storage/v1/replication/actions HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
Content-Length: 121
Accept: application/json

{
    "pool": "p1",
    "project": "proj-01",
    "share": "fs1",
    "target_pool": "pool1",
    "target": "38094753-6c90-49ed-aa92-995a296d432a"
}
```

响应示例：

```
HTTP/1.1 201 Created
Content-Length: 506
Content-Type: application/json
```

```
Location: /api/storage/v1/replication/action/8373d331-de60-e590-90e8-9ad69fcb4aec
X-Zfssa-Replication-Api: 1.0

{
    "action": {
        "project": "blue1",
        "target": "38094753-6c90-49ed-aa92-995a296d432a",
        "bytes_sent": 0.0,
        "compression": true,
        "continuous": false,
        "enabled": true,
        "dedup": false,
        "max_bandwidth": 0,
        "collection": "local",
        "estimated_size": 0.0,
        "state": "idle",
        "href": "/api/storage/v1/replication/pools/p1/projects/blah1/shares/fs1/
                 actions/8373d331-de60-e590-90e8-9ad69fcb4aec",
        "average_throughput": 0.0,
        "use_ssl": true,
        "estimated_time_left": 0.0,
        "retain_user_snaps_on_target": false,
        "share": "fs1",
        "id": "8373d331-de60-e590-90e8-9ad69fcb4aec",
        "pool": "p1",
        "include_clone_origin_as_data": false,
        "include_snaps": true
    }
}
```

创建复制操作调度。

请求示例：

```
POST /api/storage/v1/replication/actions/b77bd8cd-17ed-69da-9e4b-aebe3cc63755/schedules
 HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic cm9vdDpsMWE=
Accept: */*
Content-Type:application/json
Content-Length: 65

{"frequency":"month","day":"5th", "hour":"auto", "minute":"auto"}
```

响应示例：

```
HTTP/1.1 201 Created
Date: Thu, 12 Jan 2017 17:35:48 GMT
Server: TwistedWeb/192.0.2
Content-Length: 0
X-Zfssa-Storage-Api: 1.1
Content-Type: application/json; charset=utf-8
X-Zfssa-Api-Version: 1.0
X-Zfssa-Version: user/generic@2016.12.08,1-0
```

修改复制操作

修改现有复制操作。

请求示例：

```
PUT /api/storage/v1/replication/actions/c141d88d-ffd2-6730-d489-b71905f340cc HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json

{"use_ssl": false}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 620

{
    "action": {
        "target_id": "407642ae-91b5-681c-de5e-afcd5cbf2974",
        "compression": true,
        "continuous": false,
        "enabled": true,
        "max_bandwidth": 0,
        "dedup": false,
        "retain_user_snaps_on_target": false,
        "use_ssl": false,
        "id": "c141d88d-ffd2-6730-d489-b71905f340cc",
        "include_clone_origin_as_data": false,
        "include_snaps": true
    }
}
```

请求示例：

```
PUT /api/storage/v1/replication/actions/action_id -d '{"recovery_point_objective":
 60}HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json
```

响应示例：

```
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 620

{
    "action": {
        "state_description": "Idle (no update in progress)",
        "recovery_point_objective": 60,
        "replica_lag_over_warning_limit": false,
        "bytes_sent": "0",
        "last_try": "Mon Nov 21 2016 23:25:59 GMT+0000 (UTC)",
        "max_bandwidth": 0,
        "estimated_size": "0",
        "href": "/api/storage/v1/replication/actions/12d981c3-b098-c65a-e1e9-a6b8263a0f6a",
        "estimated_time_left": 0,
        "use_ssl": true,
        "id": "12d981c3-b098-c65a-e1e9-a6b8263a0f6a",
        "stats": {"total_logical_bytes": 40656,
        "last_dd_table_build": 9169029,
        "total_after_dedup": 18476,
        "last_try": "Mon Nov 21 2016 23:25:59 GMT+0000 (UTC)",
        "dd_total_updates": 1,
```

```
        "href": "/api/storage/v1/replication/actions/12d981c3-b098-c65a-e1e9-a6b8263a0f6a/
stats",
        "dd_total_duration": 47149245470,
        "last_logical_bytes": 40656,
        "dd_total_table_mem": 2097152,
        "last_result": "success",
        "last_after_dedup": 18476,
        "last_duration": 47149245470,
        {"dd_total_logical_bytes": 40656,
        "total_updates": 1,
        "total_duration": 47149245470,
        "replica_data_timestamp": "Mon Nov 21 2016 23:25:12 GMT+0000 (UTC)",
        "total_to_network": 9623,
        "dd_total_table_build": 9169029,
        "dd_total_phys_bytes": 16800,
        "last_to_network": 9623,
        "total_phys_bytes": 16800,
        "last_phys_bytes": 16800,
        "last_sync": "Mon Nov 21 2016 23:25:59 GMT+0000 (UTC)",
        "last_dd_table_mem": 2097152,
        "dd_total_after_dedup": 18476,
        "dd_total_to_network": 9623},
        "compression": "on",
        "dedup": true,
        "replica_lag_warning_alert": 0,
        "last_result": "success",
        "include_clone_origin_as_data": false,
        "state": "idle",
        "offline": false,
        "export_path": "",
        "export_pending": false,
        "autosnaps": {"autosnaps_retention_policies":
        "synchronized",
        "href": "/api/storage/v1/replication/actions/12d981c3-b098-c65a-e1e9-a6b8263a0f6a/
autosnaps"},
        "replica_data_timestamp": "Mon Nov 21 2016 23:25:12 GMT+0000 (UTC)",
        "continuous": false,
        "target_id": "4fd305ac-4af5-c34a-87c3-88203207305b",
        {"average_throughput": "0B/s",
        "next_update": "Sync now",
        "pool": "p1",
        "replica_lag_over_error_limit": false,
        "target": "pool1",
        "replica_lag": "42:28:24",
        "retain_user_snaps_on_target": false,
        ...
    }
}
```

监视复制操作进度

获取复制操作状态命令会返回单个复制操作 ID 所指定的单个复制操作的状态。检查 `state` 和 `state_description` 来确定复制进度。

`state` 属性值：

- `sending`
- `idle`

state_description 属性值：

- Connecting to replication target
- Receiving checkpoint from target
- Estimating size of update
- Building deduplication tables

  此属性值仅适用于删除了重复数据的复制流。

请求示例：

```
GET /api/storage/v1/replication/actions/1438ed7f-aad3-c631-d869-9e85cd7f15b4 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529

{
    "action": {
       "id": "1438ed7f-aad3-c631-d869-9e85cd7f15b4",
       "target_id": "4fd3483e-b1f5-4bdc-9be3-b3a4becd0c42",
       "target": "cleo",
       "pool": "p0",
       "replication_of": "testproj",
       "enabled": true,
       "continuous": false,
       "include_snaps": true,
       "retain_user_snaps_on_target": false,
       "dedup": true,
       "include_clone_origin_as_data": false,
       "max_bandwidth": -1,
       "bytes_sent": 0,
       "estimated_size": 0,
       "estimated_time_left": 0,
       "average_throughput": 0,
       "use_ssl": true,
       "compression": "on",
       "export_path": "",
       "state": "sending",
        "state_description": "Receiving checkpoint from target",
       "export_pending": false,
       "offline": false,
       "next_update": "Sync now",
       "replica_data_timestamp": "Thu Apr 28 2016 22:38:03 GMT+0000 (UTC)",
       "last_sync": "<unknown>",
       "last_try": "<unknown>",
       "last_result": "<unknown>",
       "replica_lag": "00:00:18",
       "recovery_point_objective": 0,
       "replica_lag_warning_alert": 0,
       "replica_lag_error_alert": 0,
       "replica_lag_over_warning_limit": false,
       "replica_lag_over_error_limit": false,
       "project": "testproj"
    }
```

```
}
```

取消更新

取消正在进行的复制更新。

请求示例：

```
PUT /api/storage/v1/replication/actions/c141d88d-ffd2-6730-d489-b71905f340cc/cancelupdate
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

发送更新

调度复制更新以尽快开始更新。

请求示例：

```
PUT /api/storage/v1/replication/actions/c141d88d-ffd2-6730-d489-b71905f340cc/sendupdate
 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

删除复制操作

删除现有复制操作。

请求示例：

```
DELETE /api/storage/v1/replication/actions/e7e688b1-ff07-474f-d5cd-cac08293506e HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

成功删除后将返回 HTTP 状态 204 (No Content)。

响应示例：

```
HTTP/1.1 204 No-Content
X-Zfssa-Replication-Api: 1.0
```

复制数据包

本节详述了复制数据包和源命令。

表 80 复制数据包命令

请求	附加到路径 /api/storage/v1/replication	说明
GET	/packages	列出所有复制数据包
GET	/packages/*package*	获取指定的复制数据包
PUT	/packages/*package*	修改指定的复制数据包
DELETE	/packages/*package*	销毁指定的复制数据包
PUT	/packages/*package*/cancelupdate	对指定的数据包运行 cancelupdate
PUT	/packages/*package*/sever	对指定的数据包运行 sever
PUT	/packages/*package*/pkgreverse	对指定的数据包运行 reverse
PUT	/packages/*package*/clone	克隆指定的数据包
GET	/packages/*package*/clone/conflicts	列出共享资源属性冲突
GET	/packages/*package*/projects	列出数据包项目
GET	/packages/*package*/projects/*project*	获取数据包项目
PUT	/packages/*package*/projects/*project*	修改数据包项目
GET	/packages/*package*/projects/*project*/usage/groups	获取数据包项目组的使用情况
GET	/packages/*package*/projects/*project*/usage/users	获取数据包项目用户的使用情况
GET	/packages/*package*/projects/*project*/snapshots	列出所有快照对象
GET	/packages/*package*/projects/*project*/snapshots/*snapshot*	获取指定的快照属性
DELETE	/packages/*package*/projects/*project*/snapshots/*snapshot*	销毁指定的快照对象
PUT	/packages/*package*/projects/*project*/snapshots/*snapshot*	重命名数据包项目快照
GET	/packages/*package*/projects/*project*/automatic	列出所有数据包项目自动快照对象
GET	/packages/*package*/projects/*project*/automatic/*automatic*	获取指定的数据包项目自动快照属性
GET	/packages/*package*/projects/*project*/filesystems	列出数据包文件系统
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*	获取数据包文件系统
PUT	/packages/*package*/projects/*project*/filesystems/*filesystem*	修改数据包文件系统
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*/usage/groups	获取数据包文件系统组的使用情况
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*/usage/users	获取数据包文件系统用户的使用情况
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	获取指定的快照属性
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots	列出所有快照对象
DELETE	/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	销毁指定的快照对象
PUT	/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	重命名数据包文件系统快照
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*/automatic	列出所有数据包文件系统自动快照对象
GET	/packages/*package*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*	获取指定的数据包文件系统自动快照属性

请求	附加到路径 /api/storage/v1/replication	说明
GET	/packages/*package*/projects/*project*/luns	列出数据包 LUN
GET	/packages/*package*/projects/*project*/luns/*lun*	获取数据包 LUN
PUT	/packages/*package*/projects/*project*/luns/*lun*	修改数据包 LUN
GET	/packages/*package*/projects/*project*/luns/*lun*/usage/groups	获取数据包 LUN 组的使用情况
GET	/packages/*package*/projects/*project*/luns/*lun*/usage/users	获取数据包 LUN 用户的使用情况
GET	/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot*	获取指定的快照属性
GET	/packages/*package*/projects/*project*/luns/*lun*/snapshots	列出所有快照对象
DELETE	/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot*	销毁指定的快照对象
PUT	/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot*	重命名数据包 LUN 快照
GET	/packages/*package*/projects/*project*/luns/*lun*/automatic	列出所有数据包 LUN 自动快照对象
GET	/packages/*package*/projects/*project*/luns/*lun*/automatic/*automatic*	获取指定的数据包 LUN 自动快照属性

复制源及其相应的数据包还可以使用下面的命令来访问。

表 81 复制源命令

请求	附加到路径 /api/storage/v1/replication/sources	说明
GET	仅使用 /api/storage/v1/replication/sources	列出复制源
GET	/*source*	列出复制源详细信息
GET	/*source*/packages/*package*	获取指定的复制数据包
PUT	/*source*/packages/*package*	修改指定的复制数据包
DELETE	/*source*/packages/*package*	销毁指定的复制数据包
PUT	/*source*/packages/*package*/cancelupdate	对指定的数据包运行 cancelupdate
PUT	/*source*/packages/*package*/sever	对指定的数据包运行 sever
PUT	/*source*/packages/*package*/pkgreverse	对指定的数据包运行 reverse
PUT	/*source*/packages/*package*/clone	克隆指定的数据包
GET	/*source*/packages/*package*/clone/conflicts	列出共享资源属性冲突
GET	/*source*/packages/*package*/projects	列出数据包项目
GET	/*source*/packages/*package*/projects/*project*	获取数据包项目
PUT	/*source*/packages/*package*/projects/*project*	修改数据包项目
GET	/*source*/packages/*package*/projects/*project*/usage/groups	获取数据包项目组的使用情况
GET	/*source*/packages/*package*/projects/*project*/usage/users	获取数据包项目用户的使用情况
GET	/*source*/packages/*package*/projects/*project*/snapshots/*snapshot*	获取指定的快照属性
GET	/*source*/packages/*package*/projects/*project*/snapshots	列出所有快照对象
DELETE	/*source*/packages/*package*/projects/*project*/snapshots/*snapshot*	销毁指定的快照对象
PUT	/*source*/packages/*package*/projects/*project*/snapshots/*snapshot*	重命名数据包项目快照

请求	附加到路径 /api/storage/v1/replication/sources	说明
GET	/*source*/packages/*package*/projects/*project*/automatic	列出所有数据包项目自动快照对象
GET	/*source*/packages/*package*/projects/*project*/automatic/*automatic*	获取指定的数据包项目自动快照属性
GET	/*source*/packages/*package*/projects/*project*/filesystems	列出数据包文件系统
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*	获取数据包文件系统
PUT	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*	修改数据包文件系统
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/usage/groups	获取数据包文件系统组的使用情况
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/usage/users	获取数据包文件系统用户的使用情况
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	获取指定的快照属性
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots	列出所有快照对象
DELETE	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	销毁指定的快照对象
PUT	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*	重命名数据包文件系统快照
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/automatic	列出所有数据包文件系统自动快照对象
GET	/*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*	获取指定的数据包文件系统自动快照属性
GET	/*source*/packages/*package*/projects/*project*/luns	列出数据包 LUN
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*	获取数据包 LUN
PUT	/*source*/packages/*package*/projects/*project*/luns/*lun*	修改数据包 LUN
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*/usage/groups	获取数据包 LUN 组的使用情况
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*/usage/users	获取数据包 LUN 用户的使用情况
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot*	获取指定的快照属性
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots	列出所有快照对象
DELETE	/*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot*	销毁指定的快照对象
PUT	/*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot*	重命名数据包 LUN 快照
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*/automatic	列出所有数据包 LUN 自动快照对象
GET	/*source*/packages/*package*/projects/*project*/luns/*lun*/automatic/*automatic*	获取指定的数据包 LUN 自动快照属性

列出复制源

列出所有可用的复制源。

请求示例：

```
GET /api/storage/v1/replication/sources HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

输出示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529

{
    "sources": [{
        "asn": "314d252e-c42b-e844-dab1-a3bca680b563",
        "href": "/api/storage/v1/replication/sources/zfs-repl-host",
        "ip_address": "ipaddr-1",
        "name": "zfs-repl-host",
        "source": "source-000"
    }]
}
```

列出复制数据包

列出所有复制数据包。

请求示例：

```
GET /api/storage/v1/replication/packages HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json
```

结果示例：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 529

{
  "packages": [
    {
      "href": "/api/storage/v1/replication/packages/0efaab49-7b22-4d4a-def8-813c27780894",
      "id": "0efaab49-7b22-4d4a-def8-813c27780894",
      "source_name": "sourceA",
      "source_asn": "8a22f6e0-4ee4-4b85-f141-e152f5fac961",
      "source_ip": "ipaddr-1",
      "source_pool": "poolA",
      "target_pool": "poolA",
      "replica_of": "projTest",
      "enabled": true,
      "state": "idle",
      "state_description": "Idle (no update in progress)",
      "offline": false,
      "import_path": "",
      "data_timestamp": "2017-03-09T22:36:12Z",
      "last_sync": "2017-03-09T22:36:22Z",
      "last_try": "2017-03-09T22:36:22Z",
      "last_result": "success"
    }
  ]
```

```
}
```

修改数据包

修改数据包属性。

属性	类型	说明
enabled	boolean	复制更新的当前状态

请求示例：

```
PUT /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json

{"enabled": false}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

请求示例：

```
PUT /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/pkgreverse
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
Content-Type: application/json

{"new_project_name":"restrev", "enable_action_upon_reversal":"true"}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

删除数据包

销毁复制数据包。

请求示例：

```
DELETE /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

成功删除后将返回 HTTP 状态 204 (No Content)。

响应示例：

```
HTTP/1.1 204 No-Content
X-Zfssa-Replication-Api: 1.0
```

取消更新

取消此数据包正在进行的更新。

请求示例：

```
PUT /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/cancelupdate
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

如果未进行更新，将返回 HTTP 状态 409 (Conflict)。

响应示例：

```
HTTP/1.1 409 Conflict
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 137

{
    "cancelupdate": {
        "AKSH_ERROR": "EAK_NAS_REPL_BADSTATE",
        "message": "operation illegal for state"
    }
}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

克隆数据包

克隆数据包项目。

请求示例：

```
PUT /api/v1/storage/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/clone
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

克隆成功返回 HTTP 状态 202 (Accepted)。帮助命令可用于确定克隆操作是否出现冲突。

克隆冲突请求示例：

```
GET /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/clone/
conflicts HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

克隆/冲突返回冲突：

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 58

{
    "conflicts": "There are no conflicts."
}
```

属性：

```
Default settings:
            target_project = (unset)
       original_mountpoint = /export
       override_mountpoint = false
                mountpoint =
```

提供数据包

提供复制连接并将数据包内容移到新项目中。此操作永久性提供此数据包以及其在源系统中的复制的共享资源，使它们成为此系统上的本地项目。任何方向的后续复制更新都需要定义新操作和发送完整更新。

请求示例：

```
PUT /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/sever
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=

{"projname":"restsev"}
```

成功响应：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

反转数据包

反转复制方向。此操作禁用此数据包的复制，并将此数据包内容移到配置用于复制回源的新本地项目。当新项目首次复制回源后，自上次成功更新后对源所做的所有元数据或数据更改都将丢失。

请求示例：

```
PUT /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/reverse
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=

{"projname":"restrev"}
```

成功响应：

```
HTTP/1.1 202 Accepted
X-Zfssa-Replication-Api: 1.0
```

加密

注 - 加密是适用于某些型号的许可功能。有关详细信息，请参阅 "Oracle Software License Agreement ("SLA") and Entitlement for Hardware Systems with Integrated Software Options" 和此软件发行版的《Licensing Information User Manual》。

Oracle ZFS Storage Appliance 提供了项目级别以及单个共享资源（文件系统和 LUN）级别的透明数据加密。该设备包括一个内置的本地密钥库，并且还可连接到 Oracle Key Manager (OKM) 系统。每个加密的项目或共享资源都需要一个来自本地或 OKM 密钥库的包装密钥。数据加密密钥由存储设备管理，并使用本地或 OKM 密钥库提供的包装密钥永久加密存储。

下表介绍了可用于管理本地和 OKM 加密的 RESTful API 请求。

表 82 本地加密

请求	附加到路径 /api/storage/v1	说明
GET	/encryption/local	获取本地密钥库属性
PUT	/encryption/local	修改本地密钥库属性
GET	/encryption/local/keys	获取本地密钥
GET	/encryption/local/keys/*key*	获取本地密钥详细信息
POST	/encryption/local/keys	创建本地密钥
DELETE	/encryption/local/keys/*key*	销毁本地密钥
GET	/encryption/local/keys/*key*/dependents	列出依赖于此密钥的共享资源

表 83 OKM 加密

请求	附加到路径 /api/storage/v1	说明
GET	/encryption/okm	获取 OKM 密钥库属性
PUT	/encryption/okm	修改 OKM 密钥库属性
GET	/encryption/okm/keys	获取 OKM 密钥
GET	/encryption/okm/keys/*key*	获取 OKM 密钥详细信息

请求	附加到路径 /api/storage/v1	说明
POST	/encryption/okm/keys	创建 OKM 密钥
DELETE	/encryption/okm/keys/*key*	销毁 OKM 密钥
GET	/encryption/okm/keys/*key*/dependents	列出依赖于此密钥的共享资源

列出所有本地密钥

输出：

```
{
    "keys": [{
            "cipher": "AES",
            "keyname": "key-1",
            "href": "/api/storage/v1/encryption/local/keys/key-000"
        },{
            "cipher": "AES",
            "keyname": "key-2",
            "href": "/api/storage/v1/encryption/local/keys/key-001"
        },{
            "cipher": "AES",
            "keyname": "key-3",
            "href": "/api/storage/v1/encryption/local/keys/key-002"
        }]
}
```

列出一个本地密钥

输出：

```
{
    "key": {
        "href": "/api/storage/v1/encryption/local/keys/key-000",
        "cipher": "AES",
        "keyname": "key-1"
    }
}
```

列出所有 OKM 密钥

输出：

```
{
    "keys": [{
            "cipher": "AES",
            "keyname": "okm-key-1",
            "href": "/api/storage/v1/encryption/local/keys/key-000"
        },{
            "cipher": "AES",
            "keyname": "okm-key-2",
            "href": "/api/storage/v1/encryption/local/keys/key-001"
```

```
        },{
            "cipher": "AES",
            "keyname": "okm-key-3",
            "href": "/api/storage/v1/encryption/local/keys/key-002"
        }]
}
```

系统命令

系统命令用于获取系统标识信息和执行顶层系统管理命令。下表列出了可用的系统命令。

设备系统命令

以下系统命令可用。

表 84 设备系统命令

请求	附加到路径 /api/system/v1	说明
GET	/version	列出设备硬件和软件版本信息。
PUT	/reboot	重新引导设备。在该重新引导期间，将应用所有已排队的平台更新。
PUT	/reboot?skip_update=true	重新引导设备而不应用任何已排队的平台更新。
PUT	/reboot?diag=true	诊断重新引导：重新引导设备，并在此过程中收集其他诊断信息。
PUT	/poweroff	关闭设备。
PUT	/restart	重新启动管理接口并收集诊断信息。
PUT	/factoryreset	将设备配置重置为出厂设置。
GET	/disks	列出所有系统磁盘。
GET	/disks/*disk*	列出指定的系统磁盘属性。
GET	/memory	系统内存状态报告。

获取版本

此命令返回包含系统标识信息的系统结构。命令成功执行后，将返回 HTTP 状态 200 (OK)。

请求示例：

```
GET /api/system/v1/version HTTP/1.1
```

```
Host: zfs-storage.example.com
Accept: application/json
```

响应示例：

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

{
    "version": {
        "hw_csn": "1211FM2009",
        "updated": "20130528T16:21:17",
        "fw_vendor": "American Megatrends Inc.",
        "os_isa": "i386",
        "os_boot": "20130528T16:25:44",
        "hw_product": "Sun Netra X4270 M3",
        "http_version": "Apache/2.2.24 (Unix)",
        "hw_asn": "2f4aeeb3-b670-ee53-e0a7-d8e0ae410749",
        "ssl_version": "OpenSSL 1.0.0k 5 Feb 2013",
        "os_machine": "i86pc",
        "os_nodename": "admin1",
        "os_version": "nas/generic@2013.05.16,1-0",
        "ak_product": "SUNW,iwashiG2",
        "fw_version": "21000208",
        "os_release": "5.11",
        "installed": "20130411T19:50:16",
        "sp_version": "3.1.2.0",
        "os_platform": "i86pc",
        "fw_release": "10/22/2012"
    }
}
```

关闭系统电源

此命令对设备执行完全关闭。所有数据服务都将永久性不可用，除非设备属于群集的一部分。要重新打开系统电源，需要访问服务处理器或物理访问电源开关。此命令异步运行并返回 HTTP 状态 202 (Accepted)。必须监控设备以关注实际命令的状态。

请求示例：

```
PUT /api/system/v1/poweroff HTTP/1.1
Host: zfs-storage.example.com
```

重新引导系统

此命令会对设备执行完全开关机循环。所有服务暂时不可用。此命令异步运行并返回 HTTP 状态 202 (Accepted)。必须监控设备以关注实际命令的状态。

注 - 如果有暂挂的平台更新可供设备使用，则在该重新引导期间将应用此更新。要执行重新引导而不应用暂挂的平台更新，请改用 `/reboot?skip_update=true` 命令。

请求示例：

```
PUT /api/system/v1/reboot HTTP/1.1
Host: zfs-storage.example.com
```

请求示例：

```
PUT /api/system/v1/reboot?skip_update=true HTTP/1.1
Host: zfs-storage.example.com
```

重新启动系统管理

此命令重新启动管理接口并收集诊断信息。此命令异步运行并返回 HTTP 状态 202 (Accepted)。必须监控设备以关注实际命令的状态。

请求示例：

```
PUT /api/system/v1/restart HTTP/1.1
Host: zfs-storage.example.com
```

诊断重新引导

此命令重新引导设备，并在此过程中收集其他诊断信息。此命令异步运行并返回 HTTP 状态 202 (Accepted)。必须监控设备以关注实际命令的状态。

注 - 如果存在可供设备使用的暂挂平台更新，则在该诊断重新引导期间将不应用此更新。

请求示例：

```
PUT /api/system/v1/reboot?diag=true HTTP/1.1
Host: zfs-storage.example.com
```

恢复出厂设置

此命令将设备配置恢复为初始出厂设置。所有配置更改都将丢失，并且设备必须执行首次安装时所进行的初始设置。此命令异步运行并返回 HTTP 状态 202 (Accepted)。必须监控设备以关注实际命令的状态。由于此命令会导致所有配置数据丢失，因此必须添加查询参数 `?confirm=true`，此命令才会成功。

请求示例：

```
PUT /api/system/v1/factoryreset?confirm=true HTTP/1.1
Host: zfs-storage.example.com
```

系统支持包

以下支持包命令可用。

表 85 支持包命令

请求	附加到路径 /api/system/v1	说明
GET	/bundles	列出所有支持包
GET	/bundles/*bundle*	获取指定的包数据或属性
POST	/bundles	创建一个支持包并将其上载到 Oracle 支持。
PUT	/bundles/*bundle*/retry	重试上载指定的包
PUT	/bundles/*bundle*/cancel	取消上载指定的包
PUT	/bundles/*bundle*/send	将指定的包上载到 Oracle 技术支持部门并提供可选的 SR 编号。
DELETE	/bundles/*bundle*	销毁指定的包

创建支持包

创建新的支持包以帮助解决服务请求。必须提供服务请求 (Service Request, SR) 编号将支持包与未解决的服务请求相关联，并将该编号发送给 Oracl 技术支持部门。SR 编号必须使用 `3-`*nnnnnnnnn* 格式。要支持包自动上载到 Oracle 支持，必须使用具有上载权限的有效 MOS 凭证注册 "Phone Home Settings"（电话主页设置）。

请求示例：

```
POST /api/system/v1/bundles HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 23

{"srn": "3-0123456789"}
```

响应示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
```

如果未提供服务请求编号 (Service Request Number, SRN)，系统将改为创建一个本地包。

请求示例：

```
POST /api/system/v1/bundles HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 23
```

响应示例：

```
{
  "bundle": {
    "status": "",
    "uuid": "d4431d57-ba4f-4f37-fa1e-a09fcbf3e56b",
    "associated_bundle": [
      {
        "href": "/api/system/v1/bundles/4050963a-4082-663f-99c0-fee915f2839c"
      }
    ],
    "srn": null,
    "filename": "ak.d4431d57-ba4f-4f37-fa1e-a09fcbf3e56b.tar.gz",
    "href": "/api/system/v1/bundles/d4431d57-ba4f-4f37-fa1e-a09fcbf3e56b",
    "date": "Thu Mar 10 2016 19:38:58 GMT+0000 (UTC)",
    "type": "User initiated"
  }
}
```

列出支持包

此命令列出系统正在处理或收集的所有支持包。当支持包上载到 Oracle 支持后，此支持包将从系统中删除。

请求示例：

```
GET /api/system/v1/bundles HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

结果示例：

```
{
    "bundles": [{
        "status": "building",
        "step_progress": 6.25,
        "srn": "3-0123456789",
        "filename": "/upload/issue/3-0123456789/3-0123456789_ak.ba8ebd55-2349-c31c-cde3-
acf3fb0c3389.tar.gz",
        "href": "/api/system/v1/bundles/ba8ebd55-2349-c31c-cde3-acf3fb0c3389",
        "date": "Wed Apr 30 2014 19:31:06 GMT+0000 (UTC)",
        "type": "User initiated",
        "uuid": "ba8ebd55-2349-c31c-cde3-acf3fb0c3389"
    }],
}
```

获取支持包

获取单个包的属性。

请求示例：

```
GET /api/system/v1/bundles/9604155c-928b-cf97-c826-cda9fc17ac57 HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

结果示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 165

{
    "bundle": {
        "status": "building",
        "step_progress": 62.5,
        "srn": "3-0123456789",
        "filename": "/upload/issue/3-0123456789/3-0123456789_ak.ba8ebd55-2349-c31c-cde3-
acf3fb0c3389.tar.gz",
        "href": "/api/system/v1/bundles/ba8ebd55-2349-c31c-cde3-acf3fb0c3389",
        "date": "Wed Apr 30 2014 19:31:06 GMT+0000 (UTC)",
        "type": "User initiated",
        "uuid": "ba8ebd55-2349-c31c-cde3-acf3fb0c3389"
    }
}
```

取消支持包

此命令取消支持包的自动上载。

请求示例：

```
PUT /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3/cancel HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

重试支持包上载

此命令可创建尝试将包上载到 Oracle 支持的新的包上载作业。获取包命令可用于监视支持包上载的状态。

请求示例：

```
PUT /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3/retry HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

要使用不同的服务请求 (Service Request, SR) 编号重试包上载，请使用 send 命令。如果未提供 SR 编号，系统将使用原始 SR 编号重试此上载。

注 - 对本地生成的包运行 send 命令时，需要提供 SR 编号，否则会抛出一个错误。

请求示例：

```
PUT /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3/send HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215

{"srn": "3-0123456789"}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

上载支持包

可手动上载不会自动上载到 Oracle 技术支持部门的支持包。

注 - 对本地生成的包运行 send 命令时，需要提供 SR 编号，否则会抛出一个错误。

请求示例：

```
PUT /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3/send HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215

{"srn": "3-0123456789"}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

删除支持包

此命令将支持包从设备中删除。

请求示例：

```
DELETE /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3 HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

系统更新

这些命令用于管理系统更新映射。

表 86 更新命令

请求	附加到路径 /api/system/v1	说明
GET	/updates	列出所有系统更新
GET	/updates/*update*	获取指定的系统更新属性
GET	/update/platform	显示平台固件的更新状态（请参阅控制器上的服务处理器和系统板固件）。
GET	/update/firmware	显示组件固件的更新状态（请参阅磁盘和 SSD 固件，另请参阅磁盘盒 IOM 固件）
PUT	/updates/*update*	修改更新设置
PUT	/updates/*update*/upgrade	升级到指定的更新映射
PUT	/updates/*update*/check	对指定的更新映射执行升级运行状况检查
PUT	/updates/*update*/rollback	回滚到指定的更新映射
PUT	/updates-apply	应用不兼容的延迟更新
DELETE	/updates/*update*	销毁指定的系统更新
POST	/updates	将更新映射加载到设备上

表 87 系统更新属性

属性	类型	说明
`version`	string	更新介质版本
`release_date`	DateTime	更改发行日期
`install_date`	DateTime	更新最新安装日期；如果未安装，则为下载到设备的日期
`status`	string	更新介质状态（不可变）
`update_deferred`	ChooseOne	延迟设置：`onreboot` 或 `onrequest`

延迟更新通知：

```
The following updates enable features that are incompatible with earlier
software versions. As these updates cannot be reverted once committed, and
peer system resources are updated across a cluster, verifying first that the
system upgrade is functioning properly before applying deferred updates is
advised.
```

列出系统更新

用于获取系统更新的示例请求：

```
GET /api/system/v1/updates HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Length: 541
Content-Type: application/json

{
    "updates": [
    {
        "release_date": "Tue Aug 13 2013 17:47:32 GMT+0000 (UTC)",
        "install_date": "Wed Aug 14 2013 12:33:08 GMT+0000 (UTC)"
        "href": "/api/system/v1/updates/nas@2013.08.13,1-0",
        "status": "previous",
        "version": "2013.08.13,1-0"
    },
    {
        "release_date": "Sat Aug 24 2013 17:54:23 GMT+0000 (UTC)",
        "install_date": "Sun Aug 25 2013 11:30:14 GMT+0000 (UTC)"
        "href": "/api/system/v1/updates/nas@2013.08.24,1-0",
        "status": "current",
        "version": "2013.08.24,1-0"
    },
    {
        "release_date": "Sun Aug 25 2013 12:56:57 GMT+0000 (UTC)",
        "install_date": "Mon Aug 26 2013 18:50:33 GMT+0000 (UTC)"
        "href": "/api/system/v1/updates/nas@2013.08.25,1-0",
        "status": "waiting",
        "version": "2013.08.25,1-0"
    }]
}
```

获取系统更新

获取单个更新映像的属性。

请求示例：

```
GET /api/system/v1/updates/nas@2013.08.25,1-0 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
```

```
Content-Length: 541
Content-Type: application/json

{
    "update": {
        "release_date": "Sat Aug 24 2013 17:54:23 GMT+0000 (UTC)",
        "install_date": "Sun Aug 25 2013 11:30:14 GMT+0000 (UTC)"
        "href": "/api/system/v1/updates/nas@2013.08.24,1-0",
        "status": "current",
        "version": "2013.08.24,1-0",
        "update_deferred", "on_request"
    }
}
```

获取平台固件更新状态

获取暂挂平台固件更新的更新状态。平台固件是控制器上的服务处理器和系统板固件的统称。

请求示例：

```
GET /api/system/v1/update/platform HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Length: 541
Content-Type: application/json

{
    "platform": {
       "href": "/api/system/v1/update/platform",
       "power_down_needed": true,
       "update_needed": "true"
    }
}
```

获取组件固件更新状态

获取暂挂、已失败和正在进行的组件固件更新的数量。组件固件是磁盘、SSD 固件和磁盘盒 IOM 固件的统称。

请求示例：

```
GET /api/system/v1/update/firmware HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Length: 541
Content-Type: application/json

{
    "firmware": {
        "href": "/api/system/v1/update/firmware",
        "upgrades_pending": 0,
        "upgrades_failed": 0,
        "upgrades_in_progress": 0
    }
}
```

上载系统更新

此命令可上载新的系统更新映像。

使用 `curl` 的上载命令示例：

```
$ curl --user root:root-password -k --data-binary @nas@2013.08.24,1-0.pkg.gz \
  --header "Content-Type: application/octet-stream" \
  https://zfs-storage.example.com/api/system/v1/updates
```

在上载并解压缩映像后，将返回更新映像的属性。成功后，HTTP 状态将设置为 "201 (Created)"，并在位置头中返回新映像的相对位置。

结果示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Length: 541
Content-Type: application/json
Location: /api/system/v1/updates/nas@2013.08.24,1-0

{
    "update": {
        "release_date": "Sat Aug 24 2013 17:54:23 GMT+0000 (UTC)",
        "install_date": "Sun Aug 25 2013 11:30:14 GMT+0000 (UTC)"
        "href": "/api/system/v1/updates/nas@2013.08.24,1-0",
        "status": "current",
        "version": "2013.08.24,1-0",
        "update_deferred", "on_request"
    }
}
```

升级

此命令可加载更新映像并将设备重新引导到指定的更新映像。指定的映像状态应等于 "previous"，否则命令将失败。

请求示例：

```
PUT /api/system/v1/updates/nas@2013.08.25,1-0/upgrade
Host: zfs-storage.example.com:215
Authorization: Basic abcefgMWE=
Content-Length: 0
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

回滚

回滚会将设备重新引导到上一个更新映像。

请求示例：

```
PUT /api/system/v1/updates/nas@2013.08.24,1-0/rollback
Host: zfs-storage.example.com:215
Authorization: Basic abcefgMWE=
Content-Length: 0
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
```

删除更新映像

将未使用的更新映像从设备中删除。

请求示例：

```
DELETE /api/system/v1/updates/nas@2013.08.13,1-0 HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcefgMWE=
```

响应示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

RESTful API 用户服务

RESTful API 用户服务用于在设备上配置本地管理用户和用户首选项。

用户服务命令

以下用户服务命令可用。

表 88 用户服务命令

请求	附加到路径 /api/user/v1	说明
GET	仅使用 /api/user/v1	列出用户服务命令
GET	/users	列出所有用户的汇总信息
GET	/users/*user*	获取特定用户的详细信息
DELETE	/users/*user*	从系统中删除本地用户
POST	/users	创建新的本地用户，克隆现有用户以作为新用户，或从网络目录添加管理员
PUT	/users/*user*	修改用户属性
PUT	/users/*user*/preferences	修改用户首选项
GET	/users/*user*/preferences	获取用户首选项
POST	/users/*user*/exceptions	创建新的用户授权例外
GET	/users/*user*/exceptions/*auth*	获取指定的用户授权例外属性
GET	/users/*user*/exceptions	列出所有的用户授权例外对象
PUT	/users/*user*/exceptions/*auth*	修改指定的用户授权例外对象
DELETE	/users/*user*/exceptions/*auth*	销毁指定的授权对象
POST	/users/*user*/preferences/keys	创建新的用户 ssh 密钥
GET	/users/*user*/preferences/keys/*key*	获取指定的用户 ssh 密钥属性
GET	/users/*user*/preferences/keys	列出所有用户 ssh 密钥对象
PUT	/users/*user*/preferences/keys/*key*	修改给定用户的指定的 ssh 密钥
DELETE	/users/*user*/preferences/keys/*key*	销毁指定的密钥对象

列出用户

每个用户都具有以下汇总属性。

表 89 用户属性

属性	类型	说明
logname	string	用户名（创建后不可变）
uid	number	用户 ID，未为目录用户启用
fullname	string	全名
initial_password	string	密码
require_annotation	boolean	需要会话注释的标志
roles	string	此用户的角色
kiosk_mode	boolean	Kiosk 用户
kiosk_screen	string	Kiosk 屏幕

请求示例：

```
GET /api/user/v1/users HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
{
    "user":
    {
        "href": "/api/user/v1/users/admin3",
        "logname": "admin3",
        "type": "local",
        "uid": 2000000000,
        "fullname": "Administrator",
        "initial_password": "password",
        "require_annotation": false,
        "roles": [
            "basic"
        ],
        "kiosk_mode": false,
        "kiosk_screen": "status/dashboard",
        "exceptions": [
        ],
        "preferences": {
            "href": "/api/user/v1/users/admin3/preferences",
            "locale": "C",
            "login_screen": "status/dashboard",
            "session_timeout": 15,
            "advanced_analytics": false,
            "keys": [
            ]
        }
    }
```

```
}
```

获取用户

获取用户的详细信息并包括用户首选项和授权例外。每个授权例外类型都定义了其自己的属性。下面显示了用户首选项属性。

表 90 用户首选项

属性	类型	说明
locale	string	语言环境
login_screen	string	初始登录屏幕
session_timeout	string	会话超时（以分钟为单位）
advanced_analytics	string	创建可用的高级 Analytics 统计

每个用户都可以将 ssh 密钥指定为所定义的首选项的一部分。

表 91 SSH 密钥属性

属性	类型	说明
type	string	SSH 密钥类型：RSA 或 DSA
key	string	SSH 密钥内容
comment	string	与此 SSH 密钥关联的注释

请求示例：

```
GET /api/user/v1/users/admin1 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 390

{
    "user": {
        "fullname": "Administrator",
        "href": "/api/user/v1/users/admin3",
        "initial_password": "password",
        "kiosk_mode": false,
        "kiosk_screen": "status/dashboard",
        "logname": "admin3",
        "require_annotation": false,
```

```
        "roles": ["basic"]
    }
}
```

创建用户

要了解用户和用户类型的更多信息，请参见《Oracle ZFS Storage Appliance 管理指南，发行版 OS8.8.0》 中的 “了解用户和角色”。

此命令使用三种形式：

- **创建新用户** – 创建新用户
- **克隆现有用户** – 通过现有用户克隆新用户
- **添加管理员** – 允许企业目录中定义的用户管理设备。

在上述三种情况中，都会向用户发送 POST 请求，此请求的正文中包括 JSON 格式的属性。

表 92 创建新用户属性

属性	类型	说明
logname	string	新用户的登录名（必需）
uid	number	可选用户 ID
fullname	string	新用户的全名（必需）
type	string	"local"、"data"、"nologin"（默认为 "local"）
initial_password	string	初始用户密码（仅 "local" 和 "data"）
require_annotation	boolean	需要会话注释的可选标志（仅 "local"）

表 93 克隆用户属性

属性	类型	说明
user	string	源用户名
uid	number	用户 ID，未为目录用户启用
clonename	string	新的克隆登录名
fullname	string	新克隆用户的全名（不适用于目录用户）
password	string	新克隆用户的密码（不适用于目录用户或禁止登录用户）

表 94 添加管理员属性

属性	类型	说明
type	string	目录用户

属性	类型	说明
logname	string	目录用户登录名

例 1 创建本地用户

请求示例：

```
POST /api/user/v1/users HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcdefghijklmnop
Content-type: application/json

{
    "type": "local",
    "logname": "admin3",
    "initial_password": "password",
    "fullname": "Administrator"
}
```

结果示例：

```
{
    "user":
    {
        "href": "/api/user/v1/users/admin3",
        "logname": "admin3",
        "type": "local",
        "uid": 2000000002,
        "fullname": "Administrator",
        "initial_password": "password",
        "require_annotation": false,
        "roles": [
            "basic"
        ],
        "kiosk_mode": false,
        "kiosk_screen": "status/dashboard",
        "exceptions": [
        ],
        "preferences": {
            "href": "/api/user/v1/users/admin3/preferences",
            "locale": "C",
            "login_screen": "status/dashboard",
            "session_timeout": 15,
            "advanced_analytics": false,
            "keys": [
            ]
        }
    }
}
```

例 2 创建目录用户

请求示例：

```
POST /api/user/v1/users
{
    "type": "directory",
    "logname": "admin3"
}
```

结果示例：

```
{
    "user":
    {
        "href": "/api/user/v1/users/admin3",
        "logname": "admin3",
        "type": "directory",
        "uid": 26718,
        "fullname": "Administrator",
        "require_annotation": false,
        "roles": [
            "basic"
        ],
        "kiosk_mode": false,
        "kiosk_screen": "status/dashboard",
        "exceptions": [
        ],
        "preferences": {
            "href": "/api/user/v1/users/admin3/preferences",
            "locale": "C",
            "login_screen": "status/dashboard",
            "session_timeout": 15,
            "advanced_analytics": false,
            "keys": [
            ]
        }
    }
}
```

例 3 创建仅数据用户

请求示例：

```
POST /api/user/v1/users
{
    "type": "data",
    "logname": "admin3",
    "initial_password": "password",
    "fullname": "Administrator",
    "uid": 5000000
}
```

结果示例：

```
{
    "user":
    {
        "href": "/api/user/v1/users/data",
        "logname": "admin3",
        "type": "data",
        "uid": 5000000,
        "fullname": "Administrator",
        "initial_password": "password"
    }
}
```

例 4 创建禁止登录用户

请求示例：

```
POST /api/user/v1/users
{
    "type": "nologin",
    "logname": "admin3",
    "fullname": "Administrator",
    "uid": 5000001
}
```

结果示例：

```
{
    "user":
    {
        "href": "/api/user/v1/users/admin3",
        "logname": "admin3",
        "type": "nologin",
        "uid": 5000001,
        "fullname": "Administrator"
    }
}
```

修改用户

直接修改用户属性。用户资源：可添加、修改或删除例外、首选项和 ssh 密钥。"UID" 和 "Type" 创建后不可变。

请求示例：

```
PUT /api/user/v1/users/admin1 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 24

{"require_annotation": true}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 236

{
    "user": {
        "href": "/api/user/v1/users/admin3",
        "logname": "admin3",
        "type": "local",
        "uid": 2000000000,
        "fullname": "Administrator",
        "initial_password": "password",
        "require_annotation": true,
        "kiosk_mode": false,
        "kiosk_screen": "status/dashboard",
        "roles": ["basic"]
    }
}
```

删除用户

从系统中删除用户。

请求示例：

```
DELETE /api/user/v1/users/admin1 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

结果示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

工作流和脚本命令

使用此服务可以管理工作流。工作流是上载到设备并由设备管理的脚本。工作流可以在浏览器界面或命令行界面中通过先进的方式参数化和执行。还可以作为警报操作或在指定时间执行工作流；因此，工作流允许设备以捕获特定策略和过程的方式进行扩展，并可用于正式对特定组织或应用程序的最佳做法进行编码。

工作流和脚本服务命令

下表显示了工作流服务命令。

表 95 工作流服务命令

请求	附加到路径 /api/workflow/v1	说明
GET	仅使用 /api/workflow/v1	列出工作流服务命令
POST	/workflows	将新工作流上载到设备
GET	/workflows	列出所有工作流
GET	/workflows/*workflow*	列出指定的工作流属性
PUT	`/workflows/`*workflow*	修改指定的工作流属性
PUT	/workflows/*workflow*/execute	执行指定的工作流
DELETE	/workflows/*workflow*	销毁指定的工作流
POST	/scripts	上载和运行脚本
GET	/scripts	列出所有正在运行的脚本
GET	/scripts/*script*	重新连接到正在运行的脚本
DELETE	/scripts/*script*	停止正在运行的脚本

上载工作流

将工作流上载到设备。

请求示例：

```
POST /api/workflow/v1/workflows HTTP/1.1
```

```
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/javascript
Content-Length: 290

var workflow = {
    name: 'Echo',
    description: 'Echo bird repeats a song.',
    parameters: {
        song: {
            label: 'Words of a song to sing',
            type: 'String',
        }
    },
    execute: function (params) { return (params.song) }
};
```

结果示例：

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 268
X-Zfssa-Version: user/generic@2013.09.14,1-0
Location: /api/workflow/v1/workflows/f4fe892f-cf46-4d6a-9026-cd0c0cce9971

{
    "workflow": {
        "href": "/api/workflow/v1/workflows/f4fe892f-cf46-4d6a-9026-cd0c0cce9971",
        "name": "Echo",
        "description": "Echo bird repeats a song.",
        "uuid": "f4fe892f-cf46-4d6a-9026-cd0c0cce9971",
        "owner": "root",
        "origin": "<local>",
        "setid": false,
        "alert": false,
        "version": "",
        "scheduled": false
    }
}
```

列出工作流

列出设备上安装的所有工作流。如果设置了查询参数 `showhidden=true`，此列表包括通常处于隐藏状态的工作流。

请求示例：

```
GET /api/workflow/v1/workflows HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
```

```
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json; charset=utf-8
Content-Length: 1908

{
    "workflows": [{
        "description": "Clear locks held on behalf of an NFS client",
        "href": "/api/workflow/v1/workflows/10f25f2c-3a56-e733-d9c7-d4c6fd84e073",
        ...
    },
    {
        "description": "Sets up environment for Oracle Solaris Cluster NFS",
        "href": "/api/workflow/v1/workflows/2793f2dc-72de-eac4-c58b-cfbe527df92d",
        ...
    },
    {
        "description": "Removes the artifacts from the appliance used by Oracle Solaris
 Cluster NFS",
        "href": "/api/workflow/v1/workflows/9e2d5eed-cc72-67b0-e913-bf5ffad1d9e1",
        ...
    },
    {
        "description": "Sets up environment to be monitored by Oracle Enterprise Manager",
        "href": "/api/workflow/v1/workflows/bb5de1b8-b950-6da6-a650-f6fb19f1172c",
        ...
    },
    {
        "description": "Removes the artifacts from the appliance used by Oracle Enterprise
 Manager",
        "href": "/api/workflow/v1/workflows/bd7214fc-6bba-c7ad-ed1f-942c0189e757",
        ...
    }]
}
```

获取工作流

获取单个工作流的属性。在标头中，如果 Accept 指定为 application/javascript，则会返回工作流内容，否则会返回工作流属性。

将 Accept 指定为 application/javascript 的请求示例：

```
GET /api/workflow/v1/workflows/cc574599-4763-4523-9e72-b74e1246d448 HTTP/1.1
Authorization: Basic cm9vdDpsMWE=
Host: zfs-storage.example.com:215
Accept: application/javascript
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/javascript; charset=utf-8
Content-Length: 916

    var workflow = {
     name: 'Clear locks',
     description: 'Clear locks held on behalf of an NFS client',
     origin: 'Oracle Corporation',
     version: '1.0.0',
```

```
     parameters: {
      hostname: {
       label: 'Client hostname',
       type: 'String'
      },
      ipaddrs: {
       label: 'Client IP address',
       type: 'String'
      }
     },
     validate: function (params) {
      if (params.hostname == '') {
       return ({ hostname: 'Hostname cannot be empty.' });
      }

      if (params.ipaddrs == '') {
       return ({ ipaddrs: 'IP address cannot be empty.' });
      }
     },
     execute: function (params) {
      try {
       nas.clearLocks(params.hostname, params.ipaddrs);
      } catch (err) {
       return ('Failed to clear NFS locks: ' + err.message);
      }

      return ('Clear of locks held for ' + params.hostname +
          ' returned success.' );
     }
    };
```

将 Accept 指定为 application/json 的请求示例：

```
GET /api/workflow/v1/workflows/cc574599-4763-4523-9e72-b74e1246d448 HTTP/1.1
Authorization: Basic cm9vdDpsMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

响应示例：

```
HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json; charset=utf-8
Content-Length: 649

{
    "workflow": {
        "href": "/api/workflow/v1/workflows/cc574599-4763-4523-9e72-b74e1246d448",
        "name": "Clear locks",
        "description": "Clear locks held on behalf of an NFS client",
        "uuid": "cc574599-4763-4523-9e72-b74e1246d448",
        "checksum": "695d029224f614258e626fe0b3c449c1233dee119571f23b678f245f7748d13c",
        "installdate": "Wed Apr 01 2015 17:59:44 GMT+0000 (UTC)",
        "owner": "root",
        "origin": "Oracle Corporation",
        "setid": false,
        "alert": false,
        "version": "1.0.0",
        "scheduled": false
    }
}
```

修改工作流

可通过向工作流资源发送 PUT 请求来修改单个工作流的属性。

请求示例：

```
PUT /api/workflow/v1/workflows/6c2b6545-fa78-cc7b-8cc1-ff88bd628e7d HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 28

{"setid": false}
```

响应示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 234

{
    "workflow": {
        "alert": false,
        "description": "Echo bird repeats a song.",
        "href": "/api/workflow/v1/workflows/448b78e1-f219-e8f4-abb5-e01e09e1fac8",
        "name": "Echo",
        "origin": "<local>",
        "owner": "root",
        "scheduled": false,
        "setid": true,
        "uuid": "448b78e1-f219-e8f4-abb5-e01e09e1fac8",
        "version": ""
    }
}
```

执行工作流

执行工作流脚本并返回结果。所有工作流参数都必须传递到正文中的 JSON 对象。成功后，将返回 HTTP 状态 202 (Accepted) 以及包含单个结果属性的 JSON 对象，此属性包含工作流输出。

请求示例：

```
PUT /api/workflow/v1/workflows/6c2b6545-fa78-cc7b-8cc1-ff88bd628e7d/run HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 28

{"song": "tweet tweet tweet"}
```

结果示例：

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 34

{
    "result": "tweet tweet tweet\n"
}
```

删除工作流

从设备中删除工作流脚本。

请求示例：

```
DELETE /api/workflow/v1/workflows/f4fe892f-cf46-4d6a-9026-cd0c0cce9971 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

结果示例：

```
HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
```

上载和运行脚本

将脚本上载到设备并在设备上运行脚本。

根用户可以查看和访问所有已上载到设备的脚本。非根用户只能查看和访问自己的脚本。

有关脚本的更多信息，请参见《Oracle ZFS Storage Appliance 管理指南，发行版 OS8.8.0》 中的 “使用 CLI 脚本编写工具”。

此脚本列出设备上的所有共享资源。

请求示例：

```
$ curl -kv --user root:pw --data-binary @listShares.aksh \
  https://hostname:215/api/workflow/v1/scripts

POST /api/workflow/v1/scripts HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic dDpscm9vMWE=
User-Agent: curl/7.45.0
Accept: */*
Content-Length: 12
Content-Type: application/x-www-form-urlencoded
```

结果示例：

```
HTTP/1.1 201 Created
Date: Mon, 27 Mar 2017 22:16:38 GMT
X-Zfssa-Workflow-Api: 1.1
X-Zfssa-Version: user/generic@2017.02.27,1-0
X-Zfssa-Api-Version: 1.0
Content-Type: plain/text; charset=utf-8
Transfer-Encoding: chunked

default
share1
share2
fs1
lun1
```

列出所有正在运行的脚本

可使用以下命令列出所有正在运行的脚本。

根用户可以查看和访问所有已上载到设备的脚本。非根用户只能查看和访问自己的脚本。

有关脚本的更多信息，请参见《Oracle ZFS Storage Appliance 管理指南，发行版 OS8.8.0》 中的 “使用 CLI 脚本编写工具”。

请求示例：

```
$ curl -kv --user root:pw https://hostname:215/api/workflow/v1/scripts

GET /api/workflow/v1/scripts HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic cm9vdDpsMWE=
User-Agent: curl/7.45.0
Accept: */*
```

结果示例：

```
HTTP/1.1 200 OK
Date: Mon, 27 Mar 2017 22:41:06 GMT
Content-Length: 96
X-Zfssa-Workflow-Api: 1.1
X-Zfssa-Api-Version: 1.0
Content-Type: application/json; charset=utf-8

{
    "scripts": [
        {
            "time": 4,
            "href": "/api/workflow/v1/scripts/1",
            "user": "root",
            "script": "1"
        },
        {
              "time": 39,
              "href": "/api/workflow/v1/scripts/9",
              "user": "root",
              "script": "9"
        }
```

```
    ]
}
```

重新连接到正在运行的脚本

根用户可以重新连接到所有正在运行的、已上载到设备的脚本。非根用户只能重新连接到其自己的正在运行的脚本。

有关脚本的更多信息，请参见《Oracle ZFS Storage Appliance 管理指南，发行版 OS8.8.0》 中的 “使用 CLI 脚本编写工具”。

请求示例：

```
$ curl -kv -H "Accept: text/plain" --user root:pw \
  https://hostname:215/api/workflow/v1/scripts/9
```

```
GET /api/workflow/v1/scripts/9 HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic cm9vdDpsMWE=
User-Agent: curl/7.45.0
Accept: text/plain
```

结果示例：

```
{
    "test2": {,
        "str": "this is a string",
        "bool": "True",
        "posint": 994,
        "int": 1123,
        "address": "",
        "host": "192.0.2.0",
        "hostname": "example.com",
        "color": "red",
        "languages": "latin",
        "size": "red",
        "onoff": "off",
        "number": 0,
        "stringlist": "this is another string",
        "emptystringlist": "this is another string",
        "yetanotherstr": "You can't change me",
        "emptystr": "Any string",
        "password": "password",
        "longpassword": "longpassword",
        "permissions": "022",
        "nonnegativeint": 42,
        "port": 21,
        "time": "Thu Jan 01 1970 15:22:30 GMT+0000 (UTC)",
        "date": "Sun Jun 17 2007 00:00:00 GMT+0000 (UTC)",
        "datetime": "Sun Jun 17 2007 15:22:00 GMT+0000 (UTC)",
        "hostport": "ipaddr-1",
        "dn": "uid=root,ou=people,dc=fishpong,dc=com",
        "commalist": "foo,bar"
    }
},
    "utask": []
}
```

停止正在运行的脚本

根用户可以删除所有正在运行的、已上载到设备的脚本。非根用户只能访问和删除其自己的正在运行的脚本。

有关脚本的更多信息，请参见《Oracle ZFS Storage Appliance 管理指南，发行版 OS8.8.0》 中的 “使用 CLI 脚本编写工具”。

请求示例：

```
$ curl -kv -X DELETE --user root:l1a \
  https://hostname:215/api/workflow/v1/scripts/9

DELETE /api/workflow/v1/scripts/9 HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic cm9vdDpsMWE=
User-Agent: curl/7.45.0
Accept: */*
```

结果示例：

```
HTTP/1.1 204 No Content
Date: Mon, 27 Mar 2017 22:59:12 GMT
Content-Length: 0
X-Zfssa-Workflow-Api: 1.1
X-Zfssa-Version: build/generic@2017.02.27,1-0
X-Zfssa-Api-Version: 1.0
Content-Type: application/json; charset=utf-8
```

RESTful 客户机

任何 HTTP 客户机均可用作 RESTful 客户机。通过在资源 URL 中键入，即使 BUI 也可返回 RESTful API `GET` 结果。Mozilla Firefox 配置了可安装用于提出 RESTful 请求的 RESTful 客户机模块 (`https://addons.mozilla.org/en-us/firefox/addon/restclient/`)。此模块允许 `PUT`、`POST` 和 `DELETE` 请求以及正常的 HTTP `GET` 请求。

RESTful 客户机必须使用 TLS 协议，因为不再支持 SSLv2/3 协议。Curl 客户机必须使用 `curl` 版本 7.34.0 或更高版本。

本节包含有关各种 RESTful 客户机的更多详细信息。

Curl Rest 客户机

Curl 客户机必须使用 `curl` 版本 7.34.0 或更高版本。两个常用的基于 CLI 的 HTTP 客户机为 `wget` 和 `curl`。本节介绍了多个使用 `curl` 执行 RESTful API 调用的示例，以及可使用 `wget` 实现的类似功能。

获取资源数据

此示例显示如何使用简单的 HTTP `GET` 请求来获取某些 JSON 数据：

```
$ curl --user ${USER}:${PASSWORD} -k -i https://hostname:215/api/nas/v1/pools/p1

 HTTP/1.1 200 OK
 Date: Tue, 23 Jul 2018 12:57:02 GMT
 Server: WSGIServer/0.1 Python/2.6.4
 Content-Length: 284
 Content-Type: application/json
 X-Zfs-Sa-Nas-Api: 1.0

{
   "pool": {
      "profile": "mirror",
      "name": "p1",
      "usage": {
         "available": 895468984832.0,
         "total": 895500681216.0,
         "dedupratio": 100,
```

```
         "used": 31696384.0
      },
      "peer": "00000000-0000-0000-0000-000000000000",
      "state": "online",
      "owner": "admin1",
      "asn": "314d252e-c42b-e844-dab1-a3bca680b563"
    }
}
```

创建新资源

此示例显示如何在请求中发送 JSON 数据以创建新资源：

```
$ curl --user ${USER}:${PASSWORD} -s -k -i -X POST -d @-  \
  -H "Content-Type: application/json" \
  https://zfs-storage.example.com:215/api/user/v1/users <<JSON
> {"logname": "rest_user",
> "fullname": "REST User",
> "initial_password": "password"}
> JSON

HTTP/1.1 201 Created
Date: Tue, 23 Jul 2018 13:07:37 GMT
Server: WSGIServer/0.1 Python/2.6.4
X-Zfs-Sa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 357

{
   "user": {
      "logname": "rest_user",
      "fullname": "REST User",
      "initial_password": "password",
      "require_annotation": false,
      "kiosk_mode": false,
      "kiosk_screen": "status/dashboard",
      "roles": ["basic"],
      "exceptions": {},
      "preferences": {
         "href": "/api/user/v1/users/admin1/preferences",
         "locale": "C",
         "login_screen": "status/dashboard",
         "session_timeout": 15,
         "advanced_analytics": false,
         "keys": {}
      }
   }
}
```

修改现有资源

此示例修改用户的会话超时：

```
$ curl --user admin1:password -3 -s -k -i -X PUT \
  -H "Content-Type: application/json" -d @-  \
```

```
  https://zfs-storage.example.com:215/api/appliance/v1/users/admin1/preferences
 <<JSON
> {"session_timeout":60}
> JSON

HTTP/1.1 202 Accepted
Date: Wed, 24 Jul 2018 05:43:17 GMT
X-Zfs-Sa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 0

{
    "preferences": {
        "href": "appliance/v1/users/admin1/preferences",
        "locale": "C",
        "login_screen": "status/dashboard",
        "session_timeout": 60,
        "advanced_analytics": false,
        "keys": {}
    }
}
```

删除现有资源

此命令将用户从系统中删除：

```
$ curl --user ${USER}:${PASSWORD} -s -k -i -X DELETE \
  https://zfs-storage.example.com:215/api/appliance/v1/users/admin1
HTTP/1.1 204 No Content
Date: Tue, 23 Jul 2018 13:21:11 GMT
Server: WSGIServer/0.1 Python/2.6.4
X-Zfs-Sa-Appliance-Api: 1.0
Content-Length: 0
```

Python RESTful 客户机

Python RESTful API 客户机随附了一个 REST 测试库，可协助 RESTful 服务的测试开发。

RESTful 客户机程序示例：

```
>>> import urllib2
>>> import json

>>> request = urllib2.Request("https://zfsssa.example:215/api/access/v1", "")
>>> request.add_header("X-Auth-User", "rest_user")
>>> request.add_header("X-Auth-Key", "password")
>>> response = urllib2.urlopen(request)
>>> response.getcode()
201

>>> info = response.info()
>>>
        opener = urllib2.build_opener(urllib2.HTTPHandler)
```

```
>>> opener.addheaders = [("X-Auth-Session", info.getheader("X-Auth-Session")),
... ('Content-Type', 'application/json'), ('Accept', 'application/json')]
```

然后，可以使用 opener 打开那些已经预先验证并准备好发送/接收 JSON 数据的请求。

获取资源

使用以下 Python 代码可以从任何 RESTful API 资源获取数据。

GET 示例：

```
>>> request = urllib2.Request("https://zfs-storage.example.com:215/api/network/v1/routes")
>>> response = opener.open(request)
>>> response.getcode()
200
>>> body = json.loads(response.read())
>>> print json.dumps(body, sort_keys=True, indent=4)
{
    "routes": [

            {
            "destination": "ipaddr-0",
            "family": "IPv4",

            "gateway": "ipaddr-1",
            "href":
            "/api/network/v1/routes/ixgbe0,ipaddr-0,ipaddr-1",

            "interface": "ixgbe0",
            "mask": 0,
            "type": "static"

            }
    ]
}
```

创建资源

用于创建新资源的 Python 示例代码：

```
>>> action = {'category': 'network'}
>>> post_data = json.dumps(action)
>>> request = urllib2.Request("https://zfs-storage.example.com:215/api/alert/v1/actions",
 post_data)
>>> request.add_header('Content-Type', 'application/json')

>>> response = opener.open(request)
>>> response.getcode()
201
>>> response.info().getheader('Location')
'/api/alert/v1/actions/actions-001'
>>> body = json.loads(response.read())
>>> print json.dumps(body, sort_keys=True, indent=4)
{
```

```
        "actions": {
        "category": "network",
        "datalink_failed": true,

        "datalink_ok": true,
        "href":
        "/api/alert/v1/actions/actions-001",

        "ip_address_conflict": true,

        "ip_address_conflict_resolved": true,

        "ip_interface_degraded": true,
        "ip_interface_failed":
        true,
        "ip_interface_ok": true,

        "network_port_down": true,
        "network_port_up":
        true
    }
}
```

修改资源

用于修改现有资源的 Python 示例代码：

```
>>> put_data = '{"ip_address_conflict_resolved": false}'
>>>
        request = urllib2.Request("https://zfs-storage.example.com:215/api/alert/v1/
actions/actions-001", put_data)
>>> request.add_header('Content-Type', 'application/json')
>>> request.get_method = lambda: 'PUT'

>>> response = opener.open(request)
>>> response.getcode()
202
>>> body = json.loads(response.read())
>>> print json.dumps(body, sort_keys=True, indent=4)
{

        "actions": {
        "category": "network",
        "datalink_failed": true,

        "datalink_ok": true,
        "href":
        "/api/alert/v1/actions/actions-001",

        "ip_address_conflict": true,

        "ip_address_conflict_resolved": false,

        "ip_interface_degraded": true,
        "ip_interface_failed":
        true,
        "ip_interface_ok": true,

        "network_port_down": true,
        "network_port_up":
        true
```

```
    }
}
```

删除现有资源

用于删除现有资源的 Python 示例代码：

```
>>> request = urllib2.Request("https://zfs-storage.example.com:215/api/alert/v1/actions/
actions-001")
>>> request.get_method = lambda: 'DELETE'
>>> response = opener.open(request)
>>> response.getcode()
204
```