# Oracle® ZFS Storage Appliance RESTful API Guide, Release OS8.8.0

**Part No: E91294-01**
November 2018

ORACLE

Oracle ZFS Storage Appliance RESTful API Guide, Release OS8.8.0

**Part No: E91294-01**



**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

**Référence: E91294-01**



**Accès aux services de support Oracle**

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` ou le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` si vous êtes malentendant.

# Contents

# Getting Started with the Oracle ZFS Storage Appliance RESTful API

Oracle ZFS Storage Appliance provides efficient file and block data services over the network. This guide describes the Oracle ZFS Storage Appliance RESTful Application Programming Interface (API), which can be used to manage the appliance. The RESTful architecture is based on a layered client-server model that lets services be transparently redirected through standard hubs, routers, and other network systems without client configuration.

## RESTful API Authentication

The Oracle ZFS Storage Appliance RESTful API uses the same authentication credentials as the browser user interface (BUI) and the command-line interface (CLI). All requests from external clients are individually authenticated using the appliance credentials and are conducted over an HTTPS connection on port 215. The RESTful API supports HTTPS sessions that have a user-definable timeout of 15 minutes.

Authentication can take one of the following forms:

- **Basic authentication** – Each request must contain the user login.

  Example HTTP Header:

  ```
  Authorization:  Basic abcefgMWE
  ```

- **User authentication** – BUI or CLI login credentials are used for authentication. In this case, the X-Auth-User header must contain the login name, and the X-Auth-Key header must contain the login password.

  Example HTTP Headers:

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

- **Session authentication** – When a session has been authenticated, a session header can be used to continue to run commands until the session expires. After a session expires, authentication must be done again before commands are accepted.

Session Header Example:

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

# RESTful API Versions

The RESTful API version for a given release of the appliance has a global version number matching the appliance software version. This version number is returned in the response header of all requests:

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

# Service Versions

Each service has a version number as part of the Uniform Resource Identifier (URI) to access the service. The version has a major and minor number. Requests must supply the major version number, but the minor version number is optional and defaults to a value of `0` if not supplied. The major number must match the major number of the service. The minor number must be less than or equal to the minor number of the service.

For example, the following table shows whether the specified versions could be used in a client request if the client is requesting a service that is running version `2.1`.

| Request Version | Allowed |
|---|---|
| `1` | No: Major version does not match the version the service is running. |
| `2` | Yes: Major version matches, and the minor version (default `0`) is backward compatible. |
| `2.1` | Yes: Major and minor version values match the version the service is running. |
| `2.2` | No: Major version matches, but the minor version is newer than the version the service is running. |

No service API version changes are required for the following property changes. The appliance version number and model must be used to determine which properties are available. These property changes are also reflected in the CLI and BUI and are an indication of the capabilities of that appliance instance.

- New output properties (without removing old properties).

- New input properties added to an existing command, that have default values that make the command behave is it did in an earlier version.

Since a newer version of a backwards-compatible command can return additional properties, clients should be coded to ignore new properties. The minor number is incremented for backwards-compatible changes to the service API.

- Add a new command to an existing service.
- Add new query parameters to service commands.

The major number is incremented with incompatible changes to the service API.

- Removing command query parameters.
- Removing a command from an existing service.

Major releases of appliance software may include incompatible version changes. There may or may not be older versions of a given service during a major update. Each command response must contain an HTTP header with the current version of the appliance API for a given module:

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

# Common RESTful Operations

The following table shows the common RESTful operations for a given resource.

**TABLE 1** Common RESTful Operations

| Request | Path | Description |
|---|---|---|
| GET | resources | List all resources |
| GET | resources/*name* | Get a JSON object describing the selected resource |
| POST | resources | Create a new resource |
| PUT | resources/*name* | Modify the selected resource |
| DELETE | resources/*name* | Delete the selected resource |

# HTTP Response Body

All response data is encoded in JSON format as defined by RFC 4627. Unless otherwise specified, commands against a single resource return a single JSON results object with the

resource name as a property. Each command section documents which property names are returned in this JSON result object.

Unless otherwise stated, the create (`POST`) and modify (`PUT`) commands return the properties of the created or modified resource. The contents should match the values returned by the `GET` request.

Example Body:

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

Some GET commands return a list of resources.

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

**Note -** Throughout this document, commands show JSON return results that have been formatted by adding returns and spaces to make it more readable. The actual output does not contain this formatting.

# HTTP Response Headers

All appliance service commands that send data use the JSON data format and require the following header values:

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

Response Headers include the following information:

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

For list results, the content length may not be known before data is sent back. If the content length is not supplied, the client must read the response body until EOF to read all the returned data.

# Query Parameters

Some requests take optional query parameters that modify or enhance the data returned. See the documentation for each resource for details. Not every resource supports every query parameter. This section documents only the common query parameters that will be used when a resource does implement the specified query parameter.

**TABLE 2** Common Query Parameters

| Parameter | Description |
|---|---|
| `props=true` | List property metadata for a resource. The default value is `false`. |
| `limit=`*n* | Limit the number of list elements returned. |
| `start=`*n* | Index number (or time) used to begin element data returned. |
| `depth=`*n* | Index number that specifies the level of detail for the returned data. |
| `match_`*property-name*`=`*value* | List the data that matches the specified property name and value. |

# Query Parameter: props

The `props` query parameter can be used on `GET`, `POST`, and `PUT` commands to allow the end users to access the metadata. The end user will request this capability by setting query parameter `props` to `true`. For `GET` and `PUT` operations, the returned JSON object will contain the required data with list of properties' metadata. For `POST`, only metadata will be returned to help the user to create a resource properly.

**TABLE 3** Property Metadata Values

| Property | Description |
|---|---|
| `name` | Property name |

| Property | Description |
|---|---|
| `label` | Description of property |
| `immutable` | Flag indicating that property cannot be modified |
| `type` | Property type such as String, Integer, or Boolean |
| `choices` | For enumerated properties, an array of available values |

# Query Parameter: limit

The `limit` query can be used on many `GET` commands that can return a large number of elements in order to limit the maximum number of elements returned.

# Query Parameter: start

For resources that support time values, the `index` can be a time value, such as `20170531T01:13:58`, and must be expressed in UTC time.

# Query Parameter: depth

The `depth` query parameter can be used with the `GET` command to retrieve a list of resources. It is used to specify the level of detail for the returned list. The greater depth number, more detail will be returned. For example:

- `/api/...?depth=0` – Return properties of node and only children names.
- `/api/...?depth=1` – Return properties of node, names, and properties of children, only names of grandchildren.
- `/api/...?depth=2` – Return properties of node, names, and properties of children, and `depth=0` output of grandchildren.

**Note -** The `depth` query parameter is not supported for listing logs using `/api/log/v1`, and listing pools, projects, filesystems, and LUNs using `/api/storage/v1`.

Example request for query parameter `depth`:

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

In this example, a list of users will be returned with details up to `depth=2`.

Example response:

```
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"
    }]
}
```

## Query Parameter: match

The `match_`*property-name*`=`*value* query parameter can be used with the `GET` command to retrieve a list of resources. It returns a list of data that matches the specified property name and value. For example:

- `/api/...?depth=0&match_kiosk_mode=true` – Return a filtered list for `kiosk_mode` is `true` with names of children.
- `/api/...?depth=1&match_kiosk_mode=true` – Return a filtered list for `kiosk_mode` is `true` with details up to `depth=1`.
- `/api/...?depth=2&match_Fullname='Super*'&kiosk_mode=true` – Return a filtered list for `fullname` containing `Super` and `kiosk_mode` is `true` with details up to `depth=2`.

**Note -** The `match_`*property-name*`=`*value* query parameter is not supported for listing logs using `/api/log/v1`, and listing pools, projects, filesystems, and LUNs using `/api/storage/v1`.

# Appliance Errors

Errors return an HTTP status code that indicates the error along with the following fault response payload.

JSON Fault Response:

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

**TABLE 4** Common Error Codes

| Error | Code | Description |
|---|---|---|
| ERR_INVALID_ARG | 400 | Invalid input argument |
| ERR_UNKNOWN_ARG | 400 | Extra unhandled input argument |
| ERR_MISSING_ARG | 400 | Required input argument missing |
| ERR_UNAUTHORIZED | 401 | This user is not authorized to execute command |
| ERR_DENIED | 403 | Operation denied |
| ERR_STATE_CHANGED | | Conflict in system state |

| Error | Code | Description |
|---|---|---|
| ERR_NOT_FOUND | 404 | The requested item was not found |
| ERR_OBJECT_EXISTS | 409 | Request creates an object that already exists |
| ERR_CONFIRM_REQUIRED | 409 | Request requires the `?confirm=true` query parameter to complete |
| ERR_OVER_LIMIT | 413 | Input request too large to handle |
| ERR_UNSUPPORTED_MEDIA | 415 | Requested media type is not supported by request |
| ERR_NOT_IMPLEMENTED | 501 | Operation not implemented |
| ERR_BUSY | 503 | Service not available due to limited resources |

# Access Settings

The protocol version and associated cipher commands manage the SSL/TLS protocol versions and ciphers for accessing the appliance.

## Security Protocols and Ciphers Settings

By default, SSL/TLS protocol versions TLSv1.1, TLSv1.2 and their associated ciphers are enabled. You can enable TLSv1.0 by sending a PUT request to the HTTPS service to set the tls_version property.

Example Request:

```
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"] }
```

Example Result (lines are artificially broken for readability):

```
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"
}
}
```

To enable TLSv1.0 only, set the ciphers property to the list of ciphers available for TLSv1.0 only.

Example Request (lines are artificially broken for readability):

```
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"]
}
```

Example Result (lines are artificially broken for readability):

```
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"
}
}
```

**Note -** To avoid being blocked from using the RESTful API or the BUI, keep the default settings for the `tls_version` and `ciphers` properties unless otherwise needed or as instructed by Oracle Support.

# Working with the RESTful API

The access service is the entry point for all RESTful API services on the Oracle ZFS Storage Appliance. The service is used to authenticate user credentials and to list the available RESTful API services, including their versions and access points.

## Accessing the Service

To access the service, use this URL: `http://`*hostname*`:215/api/access/v1`.

To access other services, log in using the access service to get the location and versions of the available services and then use the returned URI to access those services. Service locations can change based on the current appliance configuration or release level.

**TABLE 5** Access Service Commands

| Request | Path | Description |
|---|---|---|
| GET | /api/access/v1 | Lists RESTful API service access points |
| POST | /api/access/v1 | Creates a login session |
| DELETE | /api/access/v1 | Logs out of a session |

## List Services

The list services command lists the available service access URIs. If a login session is not desired, list services can be used with appropriate credentials to list the available service access URIs. This command lists all the RESTful API services and versions available on the appliance.

Example Request:

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

Example Result:

```
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"
        } ... ]
    }
}
```

# Get Service Commands

The `get service` command returns information about that service, including a list of all the available commands.

Example Request:

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

Example Response:

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

# Authentication Sessions

An authentication session ID is obtained from the access service by sending a `POST` request. This authentication session ID can be used by all other services as an identity credential. The authentication ID is invalidated after a timeout period set by the user's session timeout property. The default is usually 15 minutes. A `DELETE` request can be used to logout and invalidate the session ID.

An authentication session is not required as clients can re-send authentication information with each request. Since the RESTful API operations are stateless, only the authentication ID is stored.

## Login Session

An empty `POST` request requests a new login session. On success, an HTTP status of 201 is returned along with a JSON object that has a single property "access" that contains a list of available RESTful API services.

Example Login Request:

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

A successful login returns HTTP status 201 (Created), as well as a session ID through the X-Auth-Session HTTP header. The response body contains a list of services accessible via this login.

Response Header:

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

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

# Logout Session

An empty `DELETE` sends a request to log out and invalidate the session.

Example Logout Request:

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

Example Response:

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

# RESTful API Alert Service

The alert RESTful API service lets you configure alert thresholds and responses to posted alerts.

## Alert Service Commands

The following table shows the alert service commands.

**TABLE 6** Alert Service Commands

| Request | Append to Path /api/alert/v1 | Description |
|---|---|---|
| GET | Use only /api/alert/v1 | List the alert service commands |
| POST | /thresholds | Create a new alert threshold watch |
| GET | /thresholds/*threshold* | Get the specified alert threshold watch properties |
| GET | /thresholds | List all alert threshold watch objects |
| PUT | /thresholds/*threshold* | Modify the specified alert threshold watch object |
| DELETE | /thresholds/*threshold* | Destroy the specified threshold object |
| POST | /actions | Create a new alert actions |
| GET | /actions/*actions* | Get the specified alert actions properties |
| GET | /actions | List all alert actions objects |
| PUT | /actions/*actions* | Modify the specified alert actions object |
| DELETE | /actions/*actions* | Destroy the specified actions object |
| POST | /actions/*actions* | Create a new alert actions action |
| GET | /actions/*actions*/*action* | Get the specified alert actions action properties |
| PUT | /actions/*actions*/*action* | Modify the specified alert actions action object |
| DELETE | /actions/*actions*/*action* | Destroy the specified action object |
| GET | /events | Listen for new alert events |

# Alert Thresholds

Thresholds can be set to create custom alert watches. The following table lists typical properties for managing an alert threshold. For a complete reference, see the CLI help.

**TABLE 7** Alert Thresholds

| Property | Type | Description |
|---|---|---|
| `uuid` | Default | Unique identifier for the watch (“immutable”) |
| `statname` | AnalyticsStatistics | Statistic to watch [“cpu.utilization”, “arc.accesses”, “arc.size”, “arc.l2_bytes”, “arc.l2_accesses”, “arc.l2_size”, “syscap.bytesused”, “syscap.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 | Whether to post alert when the stat exceeds the limit (normal) or falls below the limit (inverted) [“normal”, “inverted”] |
| `limit` | PositiveInteger | Limit value for the statistic |
| `minpost` | Duration | Minimum time condition must hold before posting alert |
| `days` | ChooseOne | Only post alert on particular days [“all”, “weekdays”, “weekends”] |
| `window_start` | TimeOfDay | Only post alerts between window_start and window_end [“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 | Only post alerts between window_start and window_end [“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 | Minimum time before reposting an alert |
| `minclear` | Duration | Minimum time of normality before reposting “all clear” alert |

# List Alert Thresholds

Lists all of the configured alert thresholds.

Example Request:

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

Example Response:

```
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
        }
    ]
}
```

# Get Alert Threshold

Lists the properties for a single alert threshold.

Example Request:

```
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
```

Example Result:

```
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
    }
}
```

# Create Alert Threshold

Creates an alert threshold.

Example Request:

```
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}
```

Example Response:

```
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",
        ...
    }
}
```

# Modify Alert Threshold

Modifies any of the properties for the specified alert threshold.

Example Request:

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

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

Example Response:

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

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

## Delete Alert Threshold

Delete the specified alert threshold.

Example Request:

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

Example Response:

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

# Alert Actions

The `category` property determines the type of alert action being defined. Each category has its own property set defined.

Supported categories are:

- ad
- all
- appliance_software
- backup
- cluster
- custom
- hardware
- hardware_faults
- ndmp
- network

- replication
- replication_source
- replication_target
- restore
- scrk
- shadow
- smf
- thresholds
- zfs_pool

**TABLE 8** Alert Actions "ad"

| Property | Type | Description |
|---|---|---|
| active_directory_degraded | Boolean | Filter should match active_directory_degraded events: `true` or `false` |
| smb_kerberos_client_authentication_degraded | Boolean | Filter should match mb_kerberos_client_authentication_degraded events: `true` or `false` |

**TABLE 9** Alert Actions "all"

| Property | Type | Description |
|---|---|---|
| all_defects | Boolean | Filter should match all_defects events |
| service_alerts | Boolean | Filter should match service_alerts events: `true` or `false` |
| all_hardware_faults | Boolean | Filter should match all_hardware_faults events: `true` or `false` |

**TABLE 10** Alert Actions "appliance_software"

| Property | Type | Description |
|---|---|---|
| obstacles_to_system_software_update | Boolean | Filter should match obstacles_to_system_software_update events: `true` or `false` |
| operating_system_kernel_panic | Boolean | Filter should match operating_system_kernel_panic events: `true` or `false` |

**TABLE 11** Alert Actions "backup"

| Property | Type | Description |
|---|---|---|
| backup_finished | Boolean | Filter should match backup_finished events: `true` or `false` |

| Property | Type | Description |
|---|---|---|
| backup_started | Boolean | Filter should match backup_started events: `true` or `false` |

**TABLE 12** Alert Actions "cluster"

| Property | Type | Description |
|---|---|---|
| cluster_i/o_link_down | Boolean | Filter should match cluster_i/o_link_down events: `true` or `false` |
| cluster_i/o_link_failed | Boolean | Filter should match cluster_i/o_link_failed events: `true` or `false` |
| cluster_i/o_link_up | Boolean | Filter should match cluster_i/o_link_up events: `true` or `false` |
| unexpected_peer_error_occurred | Boolean | Filter should match unexpected_peer_error_occurred events: `true` or `false` |
| communication_to_peer_lost | Boolean | Filter should match communication_to_peer_lost events: `true` or `false` |
| cluster_peer_panicked | Boolean | Filter should match cluster_peer_panicked events: `true` or `false` |
| failed_to_set_sp_root_password_on_cluster_peer | Boolean | Filter should match failed_to_set_sp_root_password_on_cluster_peer events: `true` or `false` |
| cluster_rejoin_failed_on_peer | Boolean | Filter should match cluster_rejoin_failed_on_peer events: `true` or `false` |
| cluster_rejoin_mismatch_on_peer | Boolean | Filter should match cluster_rejoin_mismatch_on_peer events: `true` or `false` |
| cluster_rejoin_completed_on_peer | Boolean | Filter should match cluster_rejoin_completed_on_peer events: `true` or `false` |
| cluster_peer_lost_communication_token | Boolean | Filter should match cluster_peer_lost_communication_token events: `true` or `false` |
| cluster_rejoin _failed | Boolean | Filter should match cluster_rejoin_failed events: `true` or `false` |
| cluster_rejoin_mismatch | Boolean | Filter should match cluster_rejoin_mismatch events: `true` or `false` |
| cluster_rejoin_completed | Boolean | Filter should match cluster_rejoin_completed events: `true` or `false` |
| cluster_takeover_complete | Boolean | Filter should match cluster_takeover_complete events: `true` or `false` |

| Property | Type | Description |
|---|---|---|
| resource_import_failed_during_cluster_takeover | Boolean | Filter should match resource_import_failed_during_cluster_takeover events: `true` or `false` |
| local_cluster_communication_token_lost | Boolean | Filter should match local_cluster_communication_token_lost events: `true` or `false` |

**TABLE 13** Alert Actions "custom"

| Property | Type | Description |
|---|---|---|
| patterns | Default | FMA event patterns |

**TABLE 14** Alert Actions "hardware"

| Property | Type | Description |
|---|---|---|
| fibre_channel_port_down | Boolean | Filter should match fibre_channel_port_down events: `true` or `false` |
| multiple_transient_fibre_channel_port_status_changes | Boolean | Filter should match multiple_transient_fibre_channel_port_status_changes events: `true` or `false` |
| transient_fibre_channel_port_status_change | Boolean | Filter should match transient_fibre_channel_port_status_change events: `true` or `false` |
| fibre_channel_port_up | Boolean | Filter should match fibre_channel_port_up events: `true` or `false` |
| network_port_down | Boolean | Filter should match network_port_down events: `true` or `false` |
| network_port_up | Boolean | Filter should match network_port_up events: `true` or `false` |
| chassis_connected_to_system | Boolean | Filter should match chassis_connected_to_system events: `true` or `false` |
| chassis_removed | Boolean | Filter should match chassis_removed events: `true` or `false` |
| hardware_component_inserted | Boolean | Filter should match hardware_component_inserted events: `true` or `false` |
| hardware_component_removed | Boolean | Filter should match hardware_component_removed events: `true` or `false` |
| disk_inserted | Boolean | Filter should match disk_inserted events: `true` or `false` |
| disk_removed | Boolean | Filter should match disk_removed events: `true` or `false` |
| i/o_path_added | Boolean | Filter should match i/o_path_added events: `true` or `false` |
| i/o_path_removed | Boolean | Filter should match i/o_path_removed events: `true` or `false` |
| service_processor_offline_or_unavailable | Boolean | Filter should match service_processor_offline_or_unavailable events: `true` or `false` |

| Property | Type | Description |
|---|---|---|
| service_processor_online_after_outage | Boolean | Filter should match service_processor_online_after_outage events: `true` or `false` |
| failed_to_set_root_password_on_service_processor | Boolean | Filter should match failed_to_set_root_password_on_service_processor events: `true` or `false` |

**TABLE 15** Alert Actions "hardware_faults"

| Property | Type | Description |
|---|---|---|
| all_hardware_faults | Boolean | Filter should match all_hardware_faults events: `true` or `false` |

**TABLE 16** Alert Actions "ndmp"

| Property | Type | Description |
|---|---|---|
| invalid_ndmp_restore | Boolean | Filter should match invalid_ndmp_restore events: `true` or `false` |
| backup_finished | Boolean | Filter should match backup_finished events: `true` or `false` |
| backup_started | Boolean | Filter should match backup_started events: `true` or `false` |
| restore_finished | Boolean | Filter should match restore_finished events: `true` or `false` |
| restore_started | Boolean | Filter should match restore_started events: `true` or `false` |

**TABLE 17** Alert Actions "network"

| Property | Type | Description |
|---|---|---|
| datalink_failed | Boolean | Filter should match datalink_failed events: `true` or `false` |
| datalink_ok | Boolean | Filter should match datalink_ok events: `true` or `false` |
| network_port_down | Boolean | Filter should match network_port_down events: `true` or `false` |
| network_port_up | Boolean | Filter should match network_port_up events: `true` or `false` |
| ip_address_conflict | Boolean | Filter should match ip_address_conflict events: `true` or `false` |
| ip_address_conflict_resolved | Boolean | Filter should match ip_address_conflict_resolved events: `true` or `false` |

| Property | Type | Description |
|---|---|---|
| ip_interface_degraded | Boolean | Filter should match ip_interface_degraded events: `true` or `false` |
| ip_interface_failed | Boolean | Filter should match ip_interface_failed events: `true` or `false` |
| ip_interface_ok | Boolean | Filter should match ip_interface_ok events: `true` or `false` |

**TABLE 18** Alert Actions "replication"

| Property | Type | Description |
|---|---|---|
| receive_failed_(unsupported _version) | Boolean | Filter should match receive_failed_ (unsupported_version) events: `true` or `false` |
| receive_failed_(cancelled) | Boolean | Filter should match receive_failed_(cancelled) events: `true` or `false` |
| receive_failed_(all_others) | Boolean | Filter should match receive_failed_(all_others) events: `true` or `false` |
| receive_failed_(out_of_space) | Boolean | Filter should match receive_failed_(out_of_space) events: `true` or `false` |
| receive_failed_(package_not _upgraded) | Boolean | Filter should match receive_failed_ (package_not_upgraded) events: `true` or `false` |
| receive_finished | Boolean | Filter should match receive_finished events: `true` or `false` |
| receive_started | Boolean | Filter should match receive_started events: `true` or `false` |
| send_failed_(unsupported_version) | Boolean | Filter should match send_failed_ (unsupported_version) events: `true` or `false` |
| send_failed_(cancelled) | Boolean | Filter should match send_failed_(cancelled) events: `true` or `false` |
| send_failed_(all_others) | Boolean | Filter should match send_failed_(all_others) events: `true` or `false` |
| send_failed_(connectivity) | Boolean | Filter should match send_failed_(connectivity) events: `true` or `false` |
| send_failed_(out_of_space) | Boolean | Filter should match send_failed_(out_of_space) events: `true` or `false` |
| send_failed_(remote _verification) | Boolean | Filter should match send_failed_ (remote_verification) events: `true` or `false` |
| send_finished | Boolean | Filter should match send_finished events: `true` or `false` |
| send_skipped_(already_running) | Boolean | Filter should match send_skipped_ (already_running) events: `true` or `false` |
| send_started | Boolean | Filter should match send_started events: `true` or `false` |

**TABLE 19** Alert Actions "replication_source"

| Property | Type | Description |
|---|---|---|
| send_failed_(unsupported_version) | Boolean | Filter should match send_failed_ (unsupported_version) events: `true` or `false` |
| send_failed_(cancelled) | Boolean | Filter should match send_failed_(cancelled) events: `true` or `false` |
| send_failed_(all_others) | Boolean | Filter should match send_failed_(all_others) events: `true` or `false` |
| send_failed_(connectivity) | Boolean | Filter should match send_failed_(connectivity) events: `true` or `false` |
| send_failed_(out_of_space) | Boolean | Filter should match send_failed_(out_of_space) events: `true` or `false` |
| send_failed_(remote_verification) | Boolean | Filter should match send_failed_ (remote_verification) events: `true` or `false` |
| send_finished | Boolean | Filter should match send_finished events: `true` or `false` |
| send_skipped_(already_running) | Boolean | Filter should match send_skipped_ (already_running) events: `true` or `false` |
| send_started | Boolean | Filter should match send_started events: `true` or `false` |

**TABLE 20** Alert Actions "replication_target"

| Property | Type | Description |
|---|---|---|
| receive_failed_(unsupported_verrsion) | Boolean | Filter should match receive_failed_ (unsupported_version) events: `true` or `false` |
| receive_failed_(cancelled) | Boolean | Filter should match receive_failed_(cancelled) events: `true` or `false` |
| receive_failed _(all_others) | Boolean | Filter should match receive_failed_(all_others) events: `true` or `false` |
| receive_failed _(out_of_space) | Boolean | Filter should match receive_failed_(out_of_space) events: `true` or `false` |
| receive_failed_(package_not_upgraded) | Boolean | Filter should match receive_failed_ (package_not_upgraded) events: `true` or `false` |
| receive_finished | Boolean | Filter should match receive_finished events: `true` or `false` |
| receive_started | Boolean | Filter should match receive_started events: `true` or `false` |

**TABLE 21** Alert Actions "restore"

| Property | Type | Description |
|---|---|---|
| restore_finished | Boolean | Filter should match restore_finished events: `true` or `false` |
| restore_started | Boolean | Filter should match restore_started events: `true` or `false` |

**TABLE 22** Alert Actions "scrk"

| Property | Type | Description |
|---|---|---|
| support_bundle_build_failed | Boolean | Filter should match support_bundle_build_failed events: `true` or `false` |
| support_bundle_sent | Boolean | Filter should match support_bundle_sent events: `true` or `false` |
| support_bundle _upload_failed | Boolean | Filter should match support_bundle_upload_failed events: `true` or `false` |
| an_update_is_available_on_my_oracle_support. | Boolean | Filter should match an_update_is_available_on_my_oracle_support. events: `true` or `false` |
| no_updates_available. | Boolean | Filter should match no_updates_available. events: `true` or `false` |
| the_appliance_failed_to_verify_if_an_update_is_available. | Boolean | Filter should match the_appliance_failed_to_verify_if_an_update_is_available. events: `true` or `false` |

**TABLE 23** Alert Actions "shadow"

| Property | Type | Description |
|---|---|---|
| shadow_migration_complete | Boolean | Filter should match shadow_migration_complete events: `true` or `false` |

**TABLE 24** Alert Actions "smf"

| Property | Type | Description |
|---|---|---|
| service_failures | Boolean | Filter should match service_failures events: `true` or `false` |

**TABLE 25** Alert Actions "thresholds"

| Property | Type | Description |
|---|---|---|
| thresholdid | Default | UUID of watch whose alerts should match |

**TABLE 26** Alert Actions "zfs_pool"

| Property | Type | Description |
|---|---|---|
| resilver_finished | Boolean | Filter should match resilver_finished events: `true` or `false` |
| resilver_started | Boolean | Filter should match resilver_started events: `true` or `false` |
| scrub_finished | Boolean | Filter should match scrub_finished events: `true` or `false` |
| scrub_started | Boolean | Filter should match scrub_started events: `true` or `false` |
| hot_spare_activated | Boolean | Filter should match hot_spare_activated events: `true` or `false` |

# List Alert Actions

The list alert actions command lists all of the alert actions. To get data for a single resource, send an HTTP `GET` request to the href property of the given alert actions resource.

Example Request to Get Alert Actions:

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

Example Response:

```
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"
        }
    ]
}
```

# Get Alert Action

This command is similar to List Alert Action, but it returns only the specified alert action.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Create Alert Action

When you create an alert action `POST` request containing a JSON object, the action properties must be sent to /api/alert/v1/alerts/actions. The `category` property must be set to select the type of action to create. See the CLI documentation for all of the available `category` values on a given system.

The following are typical `category` values:

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

Example Request:

```
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"}
```

Example Response:

```
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
    }
}
```

# Modify Alert Action

Some of the properties returned by the `list` command can be modified by sending an HTTP `PUT` request.

Example Request:

```
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}
```

Example Response:

```
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
    }
}
```

## Delete Alert Action

Sending an HTTP `DELETE` request to any alert actions href or action href deletes the specified resource. A successful delete response is HTTP status 204 (No Content).

Example Request:

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

Example Response:

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

# Alert Action Items

Individual action items are added to each alert actions list.

## Create Alert Item

This adds an alert action to an existing alert actions group.

Example Request:

```
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"}
```

Example Response:

```
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"
    }
}
```

# Modify Alert Action

This modifies an existing alert action.

Example Request:

```
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"}
```

Example Response:

```
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"
    }
}
```

# Delete Alert Action Item

For a given alert action, a single action can be deleted. To delete an action, you send a `DELETE` request to the action href property.

Example Request to Delete an Action:

```
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 Services

Analytics enables you to graph a variety of statistics in real time and record data for later retrieval. You can perform both long-term monitoring and short-term analysis. Analytics uses DTrace to dynamically create custom statistics that allow different layers of the operating system stack be analyzed in detail.

## Analytics Commands

The following Analytics services are available at: `http://`*hostname*`/api/analytics/v1.0/`.

**TABLE 27** Analytics Commands

| Request | Append to Path /analytics/v1 | Description |
|---|---|---|
| GET | Use only /analytics/v1 | List analytics service information |
| POST | /worksheets | Create a new analytics dataset |
| GET | /worksheets/*worksheet* | Get the specified analytics dataset properties |
| GET | /worksheets | List all analytics dataset objects |
| PUT | /worksheets/*worksheet* | Modify the specified analytics dataset object |
| DELETE | /worksheets/*worksheet* | Destroy the specified worksheet object |
| PUT | /worksheets/*worksheet*/suspend | Suspend all worksheet datasets |
| PUT | /worksheets/*worksheet*/resume | Resume all worksheet datasets |
| POST | /worksheets/*worksheet*/datasets | Create a new worksheet dataset |
| GET | /worksheets/*worksheet*/datasets/*dataset* | Get the specified worksheet dataset properties |
| GET | /worksheets/*worksheet*/datasets | List all worksheet dataset objects |
| PUT | /worksheets/*worksheet*/datasets/*dataset* | Modify the specified worksheet dataset object |
| DELETE | /worksheets/*worksheet*/datasets/*dataset* | Destroy the specified dataset object |
| POST | /datasets | Create a new analytics dataset |
| GET | /datasets/*dataset* | Get the specified analytics dataset properties |
| GET | /datasets | List all analytics dataset objects |
| PUT | /datasets/*dataset* | Modify the specified analytics dataset object |

| Request | Append to Path /analytics/v1 | Description |
|---|---|---|
| DELETE | /datasets/*dataset* | Destroy the specified dataset object |
| PUT | /datasets | Suspend or resume all datasets |
| PUT | /datasets/*dataset*/data | Save this dataset (if unsaved) |
| DELETE | /datasets/*dataset*/data | Remove data at the given [granularity] from this dataset |
| GET | /settings | List analytics settings |
| PUT | /settings | Modify analytics settings |

# Analytics Settings

The properties described in the following table enable you to collect all analytic data, set the number of hours of data to retain, and set a hostname lookup policy.

| Property | Description |
|---|---|
| `retain_second_data` | Retention interval in hours for per-second data |
| `retain_minute_data` | Retention interval in hours for per-minute data |
| `retain_hour_data` | Retention interval in hours for per-hour data |
| `hostname_lookup` | Hostname lookup policy |

# Get Settings

This command gets the current values of analytics properties.

Example Request:

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

Example Results:

```
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
    }
}
```

# Modify Settings

The modify settings command is used to modify analytics settings, such as data retention values and the hostname lookup policy.

Example Request:

```
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}
```

Example Results:

```
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 Worksheets

A worksheet is the BUI screen on which statistics are graphed. Multiple statistics can be plotted at the same time, and worksheets can be assigned a title and saved for future viewing. The act of saving a worksheet automatically executes the archive action on all open statistics, meaning that whatever statistics were open continue to be read and archived forever. The worksheet commands can be used to manage the worksheets available from the BUI.

The following table shows properties that are used in analytics worksheets.

| Property | Description |
|---|---|
| `ctime` | Time and date when this worksheet was created |
| `mtime` | Time and date when this worksheet was last modified |
| `name` | Name of this worksheet |
| `owner` | Owner of this worksheet |
| `uuid` | Universal unique identifier for this worksheet |

# List Worksheets

Lists all currently configured analytics worksheets.

Example Request:

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

Example Response:

```
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"
    }]
```

```
}
```

# Get Analytics Worksheet

Gets a single analytics worksheet.

Example Request:

```
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
```

Example Response:

```
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"
    }
}
```

# Create Worksheets

Creates a new analytics worksheet.

Example Request:

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

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

Example Results:

```
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"
    }
}
```

# Rename Worksheets

Renames a saved worksheet.

Example Request:

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

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

Example Response:

```
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)"
    }
}
```

# Destroy Worksheets

Destroys an analytics worksheet. In this example, the worksheet name is used as the worksheet identifier but the uuid identified in the href can also be used. The behavior of this command matches the behavior of the CLI command that destroys worksheets.

Example Request:

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

Example Response:

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

# List Worksheet Datasets

Lists all datasets in the specified worksheet.

The following table shows properties that are used in dataset configuration.

| Property | Description |
|---|---|
| `name` | Name of the underlying statistic for this dataset |
| `drilldowns` | Drilldowns currently highlighted, if any |
| `seconds` | Number of seconds being displayed for this dataset |

Example Request:

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

# Add Worksheet Dataset

Creates a worksheet dataset.

Example Request:

```
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}
```

Example Results:

```
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": ""
    }
}
```

# Modify Worksheet Dataset

Modifies an existing worksheet dataset.

Example Request:

```
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}
```

Example Results:

```
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 Datasets

Analytics datasets use the following properties. All properties except for `suspended` are immutable.

| Property | Description |
|---|---|
| `name` | Name of the underlying statistic for this dataset |
| `grouping` | Group to which this statistic belongs |
| `explanation` | Explanation of underlying statistic |
| `incore` | Bytes of dataset data in-core |
| `size` | Bytes of dataset data on-disk |
| `suspended` | Boolean indicating whether dataset is currently suspended |
| `activity` | Pending dataset activity flag |

Available datasets:

- 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]

# List Datasets

Lists all configured analytic datasets.

Example Request:

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

Example Results:

```
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]",
    }, {
        ...
    }]
}
```

# Get Dataset

Gets properties from the specified dataset.

Example Request:

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

```
Accept: application/json
```

Example Results:

```
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
    }
}
```

# Create Datasets

Creates a new dataset.

Example Request:

```
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"}
```

Example Results:

```
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"
    }
}
```

# Modify Dataset

The modify dataset command is used to suspend or resume data collection of a single dataset.

Example Suspend Request:

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

{"suspended":true}
```

Example Resume Request:

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

Example Response:

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

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

# Destroy Datasets

Destroys a dataset.

Example Request:

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

Example Response:

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

## Save Dataset

Saves a dataset.

Example Request:

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

Example Response:

```
HTTP/1.1 202 Accepted
```

## Prune Dataset Data

The following table shows query parameters that are used in pruning datasets.

| Parameter | Description |
|---|---|
| `granularity` | Prune granularity. The data within a dataset can be pruned at a granularity value of `second`, `minute`, or `hour`. |
| `endtime` | Prune data collected prior to the given `endtime`. The `endtime` is an ISO 8601 time/date string such as 20130531T01:13:58. |

Example Request:

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

Example Response:

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

## Get Dataset Data

Gets data from an analytic dataset. Both per-second and granular data retrieval are supported.

The following table shows time-based query parameters for getting dataset data.

| Parameter | Description |
|---|---|
| `start` | The time to start collecting sample data. The default value is the current time. |
| `seconds` | Number of seconds to collect sample data. The default value is 1. The `seconds` parameter is ignored if the `span` and `interval` parameters are specified. |
| `span` | Duration of time to collect sample data: `minute`, `hour`, `day`, `week`, `month`, or `year`. |
| `granularity` | The granularity within a given span from which the average of data points is given: `minute`, `hour`, `day`, `week`, `month`, or `year`. |

If the `start` parameter is not supplied, the start time is set to the current time minus the number of `seconds` of sample data specified. The start time cannot be in the future. If the number of seconds to collect data goes beyond the current time, the server waits for each sample before returning the data.

To retrieve granular data, use a combination of parameters `span` and `granularity`. When `span` and `granularity` are used, the `seconds` parameter is ignored. If either `span` or `granularity` is entered incorrectly, the request is ignored and the `seconds` parameter is used instead. An incorrect or unsupported request displays the error message "Input span and granularity are not supported."

The following list describes how the `span` and `granularity` parameters can be combined:

- If `span` is `minute`, `granularity` can only be `minute`.
- If `span` is `hour`, `granularity` can be `minute` or `hour`.
- If `span` is `day`, `granularity` can be `minute`, `hour`, or `day`.
- If `span` is `week`, `granularity` can be `hour`, `day`, or `week`.
- If `span` is `month`, `granularity` can be `day`, `week`, or `month`.
- If `span` is `year`, `granularity` can be `week`, `month`, or `year`.

The following table shows the dataset data properties that are returned.

| Property | Description |
|---|---|
| `startTime` | The time of the first sample returned |
| `sample` | The sample index of the first sample returned |
| `data` | Array of sample data |
| `min` | The minimum value per second within the specified granularity |
| `max` | The maximum value per second within the specified granularity |

The `startTime` property can be in one of the following formats:

- An ISO 8601 time/date string (for example, 20130531T01:13:58)
- Sample index number
- The string literal `now`

Example request to collect three seconds of live data:

```
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
```

Example Results:

```
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
---
```

Example request to collect seven days of live data within the span of one week:

```
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
```

Example Results:

```
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
```

# Hardware Services

This section describes management of the hardware cluster, chassis, and components.

## Cluster

The Cluster command is used to set up clustering and manage clustered resources.

**TABLE 28** Cluster Commands

| Request | Append to Path /hardware/v1 | Description |
|---|---|---|
| GET | /cluster | Get cluster properties and cluster resource list |
| GET | /cluster/resources/*resource:path* | Get properties for the specified cluster resource |
| PUT | /cluster/resources/*resource:path* | Modify the specified cluster resource |
| PUT | /cluster/failback | Fail back all resources assigned to the cluster peer |
| PUT | /cluster/takeover | Take over all resources assigned to the cluster peer |
| PUT | /cluster/unconfigure | Unconfigure a clustered appliance to standalone mode |
| GET | /cluster/links | Get cluster card link status |
| PUT | /cluster/setup | Run through initial cluster setup |

## Get Cluster Properties

Gets the current cluster configuration state and resource properties.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Get Cluster Resource

By following the href property from cluster resources, it is possible to get the data for that single cluster resource. In the previous example, two resources are available: /hardware/v1/cluster/resources/resources/zfs/zfs-storage-1 and /hardware/v1/cluster/resources/resources/net/ixgbe0.

# Modify Cluster Resource

When a system is clustered, it is possible to modify the properties for each cluster resource with this command. For more information, see CLI "configuration cluster resources."

# Cluster Commands

The commands supported by cluster are `failback`, `takeover`, and `unconfigure`. All commands take a `PUT` request to the cluster resource with the name of the command appended. On success, the commands return HTTP status 202 (Accepted).

The failback operation is asynchronous. When the REST client sends a failback request using the PUT command, HTTP status 202 (Accepted) is returned after successfully receiving the request. The client will need to monitor failback progress by listening for alerts or polling the cluster state.

Example Request:

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

Example Result:

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

If the cluster is not in the correct state to accept the command, an HTTP status 409 (Conflict) is returned.

# Cluster Links

This command returns the current link status of the cluster card. The output is the same as the `aksh` command "configuration cluster links." It is recommended to run this command before running cluster setup to ensure that there is no issue with the cluster cabling. All links should be in the AKCIOS_ACTIVE state before running setup.

Example Request:

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

Example Response:

```
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"
    }
}
```

## Setup Cluster

The `setup cluster` command sets up initial clustering for the system. All cluster links should be in the AKCIOS_ACTIVE state and the peer system should powered on but not configured or this command fails.

Example Request:

```
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"}
```

Example Result:

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

# Chassis

The hardware commands are used to get a list of appliance hardware chassis and components.

**TABLE 29** Hardware Commands

| Request | Append to Path /hardware/v1.0 | Description |
|---|---|---|
| GET | /chassis | List hardware chassis |
| GET | /chassis/*chassis* | Get the specified hardware chassis properties |

| Request | Append to Path /hardware/v1.0 | Description |
|---|---|---|
| PUT | /chassis/*chassis* | Modify the specified hardware chassis properties |
| GET | /chassis/*chassis*/*fru_type* | List hardware chassis components |
| GET | /chassis/*chassis*/*fru_type*/*fru* | Get the specified chassis component properties |
| PUT | /chassis/*chassis*/*fru_type*/*fru* | Modify hardware chassis component properties |

# List Chassis

The get chassis command does not take any arguments and returns a list of system chassis objects. An HTTP status 200 (OK) is returned for a successful command.

| Property | Type | Description |
|---|---|---|
| `name` | string | Chassis name |
| `model` | string | Chassis model number |
| `manufacturer` | string | Chassis manufacturer |
| `serial` | string | Chassis serial number |
| `revision` | string | Chassis revision level |
| `part` | string | Chassis replacement part number |
| `type` | string | Chassis storage type |
| `faulted` | boolean | Fault indicator |
| `fru` | string | FMRI representation of the chassis |
| `uuid` | string | Chassis uuid identifier |

Example Request:

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

Example Response:

```
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"
    }]
}
```


# Get Chassis Components

This command returns all the hardware components within the specified chassis. An HTTP status 200 (OK) is returned for a successful command.

Example Request:

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

```
Accept: application/json
```

Example Response:

```
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",
        }],
    }
}
```

# Get Hardware Component

This command returns the properties from a single hardware component. An HTTP status 200 (OK) is returned for a successful command. The response object contains the component properties contained in the following table.

| Property | Type | Description |
|---|---|---|
| `device` | string | The FRU device ID |
| `faulted` | boolean | Flag indicating whether FRU is faulted |
| `fru` | string | FMRI representation of a FRU |
| `interface` | string | FRU interface type |
| `label` | string | FRU location label |
| `locate` | boolean | Locate indicator on flag |
| `manufacturer` | string | FRU manufacturer |
| `model` | string | FRU model |
| `part` | string | FRU part number |

| Property | Type | Description |
|---|---|---|
| present | boolean | FRU presence indicator |
| rpm | number | Platter RPM (disk only) |
| serial | string | FRU serial number |
| size | number | FRU size (capacity) |
| type | string | Component type |
| use | string | Component usage enumeration |

Example Request:

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

Example Response:

```
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"
    }
}
```

# Modify Component Property

A PUT request can be used to set properties on a selected hardware component. A successful request returns HTTP status 201 (Accepted) as well as the component properties in JSON format.

Example Request:

```
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}
```

Example JSON Response:

```
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
    }
}
```

# Log Commands

The log commands manage the logs available under the CLI "maintenance logs" menu. For individual service log information, see the service API.

## Manage Logs Commands

The following table show how to call the manage logs commands.

**TABLE 30** Manage Logs Commands

| Request | Append to Path /api/log/v1 | Description |
|---|---|---|
| GET | Use only /api/log/v1 | List the log service commands |
| GET | /logs | List all log types |
| GET | /logs/?start=*index/time*&limit=*entry limit* | Get log entries for the selected range |
| GET | /logs/alert | List all alert logs |
| GET | /logs/alert?start=*index/time*&limit=*entry limit* | Get log entries for the selected range |
| GET | /collect | Download a collection of all log entries |
| GET | /collect?start=*index/time*&limit=*entry limit* | Download a collection of log entries from a selected range |

## List Logs

This command lists all of the logs available on an appliance. Each log returns the number of entries in the log and a time stamp of the last entry.

---

**Note -** The `depth` query parameter and the `match_`*property-name=value* query parameter are not supported.

---

Example Request:

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

Example Results:

```
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"
        }
    ]
}
```

# Get Log Entries

Log entries can be returned from the specified appliance log. Each log entry returns the date/time of the entry along with log specific content properties.

**Note -** Depending on the number of logs, older log entries might not be available due to memory constraints. This same limit occurs in the BUI and CLI. To obtain all system logs, use the `collect` function described in "Manage Logs Commands" on page 79.

| Parameter | Description |
|---|---|
| `start=`*index* | Start returning logs from the given index/time |
| `limit=`*number* | Limit number of log entries returned |

The start index defaults to the value of 0, which returns the first log that was generated. Negative values and values greater than or equal to the log size are not allowed. The start index can also be a time string; for example, 20130724T03:21:55.

**Note -** REST only accepts UTC time. Time values that are older than one month from the current time are not accepted. Retrieval of older logs must use an index number for the start value. The limit value limits the number of logs returned for a given request. No more than the given limit value is returned.

Example Request:

```
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
```

Example Result:

```
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"
        }
    ]
}
```


# Download Logs

The download logs command returns a gzipped tar file containing all of the system logs. The file disposition name is set to logs.tar.gz. Since the data is created and streamed in real time, it is not possible to resume a download.

# Download Log

If only one log type is desired to be downloaded, its name can be appended to the `collect` resource as shown in the table. The text of the log is streamed back to the client. If `gzip` compression is requested, the text stream is compressed with `gzip`. Other compression types are not supported and are ignored.

# Network Commands

The network commands described in this section are used to view network addresses and devices as well as configure network datalinks, interfaces, and routes.

## Networking Configuration

The network configuration features let you create a variety of advanced networking setups out of your physical network ports, including link aggregations, virtual NICs (VNICs), virtual LANs (VLANs), and multipathing groups. You can then define any number of IPv4 and IPv6 addresses for these abstractions, for use in connecting to the various data services on the system.

There are four components to a system's network configuration:

- **Devices** – Physical network ports that correspond to your physical network connections or IP on InfiniBand (IPoIB) partitions.
- **Datalinks** – The basic construct for sending and receiving packets. Datalinks may correspond 1:1 with a device (that is, with a physical network port) or IB Partition, or you can define Aggregation, VLAN and VNIC datalinks composed of other devices and datalinks.
- **Interface** – The basic construct for IP configuration and addressing. Each IP interface is associated with a single datalink, or is defined as an IP MultiPathing (IPMP) group, which is comprised of other interfaces.
- **Routing** – IP routing configuration, which controls how the system directs IP packets.

In this model, network devices represent the available hardware; they have no configurable settings. Datalinks are a layer 2 entity and must be created to apply settings such as LACP to these network devices. Interfaces are a layer 3 entity containing the IP settings, which they make available via a datalink. This model has separated network interface settings into two parts: datalinks for layer 2 settings and interfaces for layer 3 settings.

# Network Datalinks

The network datalinks command provides datalink management on the appliance. You can list, modify, create, and delete datalink resources.

**TABLE 31** Network Datalink Commands

| Request | Append to Path /network/v1 | Description |
|---|---|---|
| POST | /datalinks | Create a new network datalink |
| GET | /datalinks/*datalink* | Get the specified network datalink properties |
| GET | /datalinks | List all network datalink objects |
| PUT | /datalinks/*datalink* | Modify the specified network datalink object |
| DELETE | /datalinks/*datalink* | Destroy the specified datalink object |

**TABLE 32** Physical Device Datalink Properties

| Property | Type | Description |
|---|---|---|
| `class` | String | “device” (“immutable”) |
| `label` | NetworkLabel | Label |
| `links` | ChooseOne | Links [“igb1”, “igb0”, “ixgbe2”, “ixgbe3”, “igb4”, “igb3”, “ixgbe1”, “igb2”, “igb5”] |
| `jumbo` | Boolean | Use Jumbo Frames [“true”, “false”] (“deprecated”) |
| `mtu` | PositiveInteger | Max transmission unit (MTU) |
| `speed` | ChooseOne | Link Speed [“auto”, “10”, “100”, “1000”, “10000”] |
| `duplex` | ChooseOne | Link Duplex [“auto”, “half”, “full”] |

**TABLE 33** VNIC Device Datalink Properties

| Property | Type | Description |
|---|---|---|
| `class` | String | “vnic” (“immutable”) |
| `label` | NetworkLabel | Label |
| `links` | ChooseOne | Links [“ixgbe0”] |
| `mtu` | PositiveInteger | Max transmission unit (MTU) |
| `id` | VLAN | VLAN ID |

**TABLE 34** VLAN Device Datalink Properties

| Property | Type | Description |
|---|---|---|
| `class` | String | “vlan” (“immutable”) |

| Property | Type | Description |
|---|---|---|
| label | NetworkLabel | Label |
| links | ChooseOne | Links [“ixgbe0”] |
| mtu | PositiveInteger | Max transmission unit (MTU) |
| id | VLAN | VLAN ID |

**TABLE 35** Aggregation Based Device Datalink Properties

| Property | Type | Description |
|---|---|---|
| class | String | “aggregation” (“immutable”) |
| label | NetworkLabel | Label |
| links | ChooseN | Links [“igb1”, “igb0”, “ixgbe2”, “ixgbe3”, “igb4”, “igb3”, “ixgbe1”, “igb2”, “igb5”] |
| jumbo | Boolean | Use Jumbo Frames [“true”, “false”] (“deprecated”) |
| mtu | PositiveInteger | Max transmission unit (MTU) |
| policy | ChooseOne | Policy [“L2”, “L3”, “L4”, “L2+L3”, “L2+L4”, “L3+L4”] |
| mode | ChooseOne | Mode [“active”, “passive”, “off”] |
| timer | ChooseOne | Timer [“short”, “long”] |
| key | Integer | Aggregation Key (“immutable”) |

**TABLE 36** IP-Partition-Based Device Datalink Properties

| Property | Type | Description |
|---|---|---|
| class | String | “partition” (“immutable”) |
| label | NetworkLabel | Label |
| links | ChooseOne | Links |
| pkey | Pkey | Partition Key |
| linkmode | ChooseOne | Link Mode [“cm”, “ud”] |

# List Network Datalinks

Lists all configured datalinks on the appliance. Each object in the datalinks list contains an href to get the operation on a single datalink resource along with datalink properties.

Example Request:

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

```
Accept: application/json
```

Example JSON Data:

```
{
    "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",
        ...
    }]
}
```

# Get Network Datalink

The `GET` method returns a JSON object that contains a datalink property with a list of datalink objects.

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

Accept: application/json
```

Example JSON Data:

```
{
    "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"
    }
```

```
}
```

# Create Network Datalink

The `POST` command creates a new datalink. One additional property that is needed when creating a new datalink is the `class` property, which defines the class of datalink to create. The datalinks class is defined during datalink creation and can be one of the following class types:

- `device` – Create a device-based datalink
- `vnic` – Create a VNIC-based datalink
- `vlan` – Create a VLAN-based datalink
- `aggregation` – Create an aggregation-based datalink
- `partition` – Create an IB partition datalink

The properties map to the same CLI properties available in the "configuration net datalinks" menu.

Example Request:

```
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"
}
```

Example Response:

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

# Modify Network Datalink

The `PUT` method is used to modify datalink properties. For details on setting up datalinks, see the CLI documentation.

Example Request:

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

{"jumbo": true}
```

Example Response:

```
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"
    }
}
```

## Delete Network Datalink

This command removes the datalink from the system. Use the href path to delete the specified datalink.

Example Request:

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

Example Responses:

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

# Network Devices

These commands list the physical network devices on the system. There are no modifiable properties on physical network devices.

**TABLE 37** Network Devices Commands

| Request | Append to Path /network/v1 | Description |
|---|---|---|
| GET | /devices/*device* | Get the specified network device properties |
| GET | /devices | List all network device objects |

**TABLE 38** Network Device Properties

| Property | Description |
|---|---|
| `active` | Boolean flag indicating whether the device is active |
| `duplex` | Duplex of device |
| `factory_mac` | Factory MAC address |
| `media` | Device media |
| `speed` | Device speed, in megabits/second |
| `up` | Boolean flag indicating whether the device is operational |

# List Network Devices

This command lists all network devices.

Example Request:

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

Example Result:

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

## Get Network Device

This command gets the properties from a single network device.

Example Request:

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

Example Result:

```
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
    }
}
```

# Network Interfaces

**TABLE 39** Network Interface Commands

| Request | Append to Path /api/network/v1 | Description |
|---|---|---|
| POST | /interfaces | Create a new network interface |

| Request | Append to Path /api/network/v1 | Description |
|---|---|---|
| GET | /interfaces/*interface* | Get the specified network interface properties |
| GET | /interfaces | List all network interface objects |
| PUT | /interfaces/*interface* | Modify the specified network interface object |
| DELETE | /interfaces/*interface* | Destroy the specified interface object |

**TABLE 40** Network Interface Properties

| Property | Description |
|---|---|
| `admin` | Flag indicating whether administration is allowed on this interface |
| `class` | Class type ("ip", "ipmp") (immutable after create) |
| `curaddrs` | Current IP Addresses (immutable) |
| `enable` | Flag indicating whether this interface is enabled |
| `label` | User label for interface |
| `links` | Chose a network link for this interface |
| `state` | State of Interface (immutable) |
| `v4addrs` | IPv4 Addresses |
| `v6dhcp` | IPv4 DHCP flag |
| `v6addrs` | IPv6 Addresses |
| `v6dhcp` | IPv6 DHCP flag |

# List Network Interfaces

This command lists all of the configured network interfaces.

Example Request:

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

Example Response:

```
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"]
        ...
    }]
}
```

# Get Network Interface

This command gets the full list of properties for a specified network interface.

Example Request:

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

Example Response:

```
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
    }
}
```

# Create Network Interface

This command creates a new network interface.

Example Request:

```
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"
}
```

Example Response:

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

# Modify Network Interface

This command modifies an existing network interface.

Example Request:

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

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

Example Response:

```
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
}
```

## Delete Network Interface

This command deletes an existing network interface.

**Note -** When an interface is deleted, all routes associated with the interface are also removed.

Example Request:

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

Example Result:

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

# Network Routes

These commands manage network routes.

**TABLE 41** Manage Network Routes

| Request | Append to Path /api/network/v1 | Description |
|---|---|---|
| POST | /routes | Create a new network route |
| GET | /routes/*route* | Get the specified network route properties |
| GET | /routes | List all network route objects |
| DELETE | /routes/*route* | Destroy the specified route object |
| GET | /routing | Get net routing properties |
| PUT | /routing | Modify net routing properties |

**TABLE 42** Manage Network Route Properties

| Property | Description |
|---|---|
| `type` | Type of route such as “system” or “static” (immutable) |
| `family` | Address family (either IPv4 or IPv6) |
| `destination` | Route destination address |
| `gateway` | Gateway address |
| `interface` | Network datalink interface |

The href path to each route uses the route IDs set in the CLI, but these values can change as routes are modified. The API supports selecting single routes using unique properties within the route. The syntax is `routes/name=`*value* compared to `routes/route-###`.

# List Routes

Lists all of the network routes created on an appliance.

Example Request:

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

Example Result:

```
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"
    }]
}
```

# Get Route

Gets the properties for a single route.

Example Request:

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

Example Result:

```
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"
    }
}
```

# Add Route

Creates a new network route. The route href values can change if other routes are added to the system. No route information is returned on a create since the returned properties would be identical to the input properties. A successful create returns HTTP status 204 (Created).

Example Request to Create a Static Route:

```
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"
}
```

Example Result:

```
HTTP/1.1 201 Created
```

# Delete Route

Deletes an existing network route.

Example Request:

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

Example Result:

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

# RESTful API Problem Service

The RESTful API Problem service is used to view and manage problems discovered by the appliance fault manager.

## Problem Service Commands

**TABLE 43** Problem Service Commands

| Request | Append to Path /problem/v1 | Description |
|---|---|---|
| GET | Use only /problem/v1 | List the problem service commands |
| GET | /problems | List all current problems |
| GET | /problems/*problem* | Get detail properties for a problem with the specified uuid |
| PUT | /problems/*problem*/markrepaired | Mark the specified problem uuid as repaired |

## List Problems

This command lists all problems that are currently active on an appliance. HTTP status of 200 (OK) is returned for a successful command.

Example Request:

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

Example Response:

```
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."
        }]
    }]
}
```

# List Problem

The list problem command lists a single problem. HTTP status of 200 (OK) is returned for a successful command.

The list problem command uses the `uuid` input parameter, which is the UUID of a single problem.

Example Request:

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

Example Response:

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

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

# Repair Problem

The repair problem command marks a problem as repaired.

The repair problem command uses the `uuid` input parameter, which is the UUID of the problem to be marked repaired.

Example Request:

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

Successful Response Returns HTTP status 202 (Accepted):

```
HTTP/1.1 202 Accepted
```

# RESTful API Role Service

A role is a collection of privileges that can be assigned to users. It may be desirable to create administrator and operator roles with different authorization levels. Staff members can be assigned any role that is suitable for their needs, without assigning unnecessary privileges. The use of roles is more secure than the use of shared administrator passwords, for example, that gives everyone the root password. Roles restrict users to necessary authorizations only, and attribute their actions to their individual username in the Audit log. By default, a role called "Basic administration" exists, which contains very basic authorizations.

Use the RESTful API Role service to manage system roles and authorizations.

## Role Service Command Overview

The following list shows the role commands.

**TABLE 44** Role Service Commands

| Request | Append to Path /role/v1 | Description |
|---|---|---|
| GET | Use only /role/v1 | Lists the role service commands |
| GET | /roles/*role* | Gets the specified administrative role properties |
| GET | /roles | Lists all administrative role objects |
| PUT | /roles/*role* | Modifies the specified administrative role object |
| DELETE | /roles/*role* | Destroys the specified role object |
| POST | /roles | Creates a new role or clone an existing role |
| PUT | /roles/*role*/revoke | Removes the specified role from all users |
| POST | /roles/*role*/authorizations | Creates a new role authorization |
| GET | /roles/*role*/authorizations/*auth* | Gets the specified role authorization properties |
| GET | /roles/*role*/authorizations | Lists all role authorization objects |
| PUT | /roles/*role*/authorizations/*auth* | Modifies the specified role authorization object |
| DELETE | /roles/*role*/authorizations/*auth* | Destroys the specified auth object |

# List Roles

Each role has the following summary properties. For full descriptions of the role properties, see the CLI Help.

**TABLE 45** Role Properties

| Property | Type | Description |
|---|---|---|
| `name` | string | Role name (immutable after creation) |
| `description` | string | Description of role |

Example Request:

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

Example Response:

```
{
    "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"
    }]
}
```

# Get Role

Retrieves the properties for a single role. To return the property metadata, set the `props` query parameter to `true`.

Example Request:

```
GET /api/role/v1/roles/basic?props=true HTTP/1.1
Authorization: Basic abcefgMWE=
```

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

Example Response:

```
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"
    }
}
```

# Create Role

This command creates a new role.

**TABLE 46** Create New Role Properties

| Property | Type | Description |
|---|---|---|
| `name` | string | New roles's name (required) |
| `clone` | string | Name of role to clone original properties (optional) |
| `description` | string | Role description (required) |

Example Request:

```
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"}
```

Example Result:

```
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"
    }
}
```

# Modify Role

The role properties can be modified after a role is created.

Example Request:

```
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!"}
```

Example Result:

```
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"
    }
}
```

# Revoke Role

Revokes a role from all users.

Example Request:

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

Example Result:

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

# Delete Role

Deletes a role from the system. If the role is still assigned to one or more users, add `?confirm=true` to the `DELETE` command.

Example Request:

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

Example Result:

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

```
X-Zfssa-Appliance-Api: 1.0
```

# List Role Authorizations

Lists the authorizations for the selected role.

Example Request:

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

Example Response:

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

# Create Role Authorization

Creates a new role authorization. The input properties are the same as defined in the CLI. Each authorization has a defined `scope` property. Other properties can be set based on the input scope. Scope values include:

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

Example Request:

```
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}
```

Example Result:

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

# Modify Role Authorization

The role authorization properties can be modified.

Example Request:

```
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}
```

Example Result:

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

# Delete Role Authorization

Deletes a role authorization.

Example Request:

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

Example Result:

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

# RESTful API SAN Service

The RESTful API SAN service lets you connect your appliance to your Storage Area Network (SAN).

## SAN Overview

A SAN has the following basic components:

- A client that accesses network storage
- A storage appliance that provides network storage
- A network that links the client to the storage

These three components remain the same regardless of which protocol is used on the network. In some cases, the network might even be a cable between the initiator and the target, but in most cases some type of switching is involved. The RESTful API SAN service manages four types of SAN resources for each supported protocol:

- **Initiators** – An application or production system end-point that is capable of initiating a SCSI session and sending SCSI commands and I/O requests. Initiators are also identified by unique addressing methods.
- **Initiator groups** – A set of initiators. When an initiator group is associated with a Logical Unit Numbers (LUNs), only initiators from that group can access the LUN.
- **Targets** – A storage system end-point that provides a service of processing SCSI commands and I/O requests from an initiator. A target is created by the storage system administrator, and is identified by unique addressing methods. A target, once configured, consists of zero or more logical units.
- **Target groups** – A set of targets. LUNs are exported over all the targets in one specific target group.

## SAN Initiators

The following commands are used to manage SAN initiators.

These commands use the following URI parameters:

*protocol* The NAS protocol for the initiator: `fc`, `iscsi`, or `srp`

*initiator* The IQN, WWN, or EUI of the initiator

**TABLE 47** Initiator Commands

| Request | Append to Path /san/v1.0 | Description |
|---|---|---|
| GET | /*protocol*/initiators | List all SAN initiators for the given protocol: `fc`, `iscsi`, `srp` objects |
| GET | /*protocol*/initiators/*initiator* | Get the specified SAN initiator for the given protocol: `fc`, `iscsi`, `srp` properties |
| POST | /*protocol*/initiators | Create a new SAN initiator for the given protocol: `fc`, `iscsi`, `srp` |
| PUT | /*protocol*/initiators/*initiator* | Modify the specified SAN initiator for the given protocol: `fc`, `iscsi`, `srp` object |
| DELETE | /*protocol*/initiators/*initiator* | Destroy the specified initiator object |

Many of the initiator commands use the properties listed in the following table as return values. The create and modify commands also use the properties as input values.

**TABLE 48** Initiator Properties

| Property | Protocol | Description |
|---|---|---|
| `alias` | all | Alias for this initiator |
| `initiator` | `fc` | Port world wide name for this initiator (WWN) |
| `iqn` | `iscsi` | iSCSI qualified name for this initiator |
| `chapuser` | `iscsi` | Challenge handshake `auth` protocol (CHAP) user name |
| `chapsecret` | `iscsi` | Challenge handshake `auth` protocol (CHAP) secret |
| `initiator` | `srp` | Extended Unique Identifier (EUI) |

# List Initiators

Lists all of the initiators configured on the appliance of a specified protocol type. The response body contains an array of initiator properties named "initiators" in JSON format.

Example Request to List iSCSI Initiators:

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

```
Accept: application/json
```

Example Response:

```
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": ""
    }]
}
```

# Get Initiator Details

Lists the details of a single iSCSI initiator. The response body contains iSCSI initiator properties as an object named "initiator" in JSON format.

Example Request:

```
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
```

Example Response:

```
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": ""
    }
```

```
}
```

# Create an Initiator

Creates a new iSCSI initiator. You must supply the iSCSI Qualified Name (IQN). The request body contains the iSCSI initiator properties in JSON format. The response includes the location URI of the new iSCSI initiator in the HTTP header and status Code 201 (Created) on success. The response body contains iSCSI initiator properties as an object named "initiator" in JSON format.

Example Request:

```
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"
}
```

Example Response:

```
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": ""
    }
}
```

# Modify an Initiator

This command modifies an existing initiator. The request body contains the initiator properties that should be modified in JSON format. The IQN for the initiator is supplied in the URI. HTTP

status 202 (Accepted) is returned on success. The response body contains new iSCSI initiator properties as an object named `initiator` in JSON format.

Example Request:

```
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"
}
```

Example Response:

```
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"
    }
}
```

# Delete an Initiator

Removes an initiator from the appliance.

Example Request:

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

Successful Delete returns HTTP Code 204 (No Content):

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

# Initiator Groups

The iSCSI initiator commands are used to manage iSCSI initiators and iSCSI initiator groups on an appliance. The available commands are listed in the table below.

These commands use the following URI parameters:

*protocol* The NAS protocol for the initiator: `fc`, `iscsi`, or `srp`

*name* The name of the initiator group

Each initiator group has a `name` property and an `initiators` property that contains a list of initiators in the initiator group.

**TABLE 49** Initiator Group Commands

| Request | Append to Path /san/v1.0 | Description |
|---|---|---|
| GET | /*protocol*/initiator-groups | List all SAN initiator groups for the given protocol: `fc`, `iscsi`, `srp` objects |
| GET | /*protocol*/initiator-groups/*name* | Get the specified SAN initiator group for the given protocol: `fc`, `iscsi`, `srp` properties |
| POST | /*protocol*/initiator-groups | Create a new SAN initiator group for the given protocol: `fc`, `iscsi`, `srp` |
| PUT | /*protocol*/initiator-groups/*name* | Modify the specified SAN initiator group for the given protocol: `fc`, `iscsi`, `srp` object |
| DELETE | /*protocol*/initiator-groups/*name* | Destroy the specified name object |

# List Initiator Groups

Lists all available iSCSI initiator groups. On success HTTP status 200 (OK) is returned and the body contains a JSON object with a property named "groups" that contains an array of initiator group objects.

Example Request:

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

Example Response:

```
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"
    }]
}
```

# Get Initiator Group Details

Gets detailed information from a single iSCSI initiator group. The group can be accessed by following the href property returned in the list initiator group command.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Create an Initiator Group

Creates an iSCSI initiator group with no members. The request body contains a JSON object with a single `name` parameter containing the group name.

**TABLE 50** Initiator Group Create Properties

| Property | Type | Description |
|---|---|---|
| `name` | string | The name of the initiator group |
| `initiators` | array | An array of existing initiator IQN properties |

Example Request:

```
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"]
}
```

Example Response:

```
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"
    }
}
```

# Delete an Initiator Group

Removes an initiator group from the appliance.

Example Request:

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

Successful delete returns HTTP status 204 (No Content):

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

# Targets

The iSCSI target commands are used to manage iSCSI targets and iSCSI target groups. The available commands are listed in the table below.

The targets commands take the following URI parameters:

*protocol* The SAN protocol: `fc`, `iscsi`, or `srp`

*target* The target ID: IQN, WWN, or EUI

**TABLE 51** Target Commands

| Request | Append to Path /san/v1.0 | Description |
|---|---|---|
| GET | /*protocol*/targets | List all SAN target for the given protocol: `fc`, `iscsi`, `srp` objects |
| GET | /*protocol*/targets/*target* | Get the specified SAN target for the given protocol: `fc`, `iscsi`, `srp` properties |
| POST | /*protocol*/targets | Create a new SAN target for the given protocol: `fc`, `iscsi`, `srp` |
| PUT | /*protocol*/targets/*target* | Modify the specified SAN target for the given protocol: `fc`, `iscsi`, `srp` object |
| DELETE | /*protocol*/targets/*target* | Destroy the specified target object |

The get target commands return target properties. The create and modify target commands use the properties listed in the following table as input.

**TABLE 52** Target Input Properties

| Property | Protocol | Description |
|---|---|---|
| `alias` | iscsi | Simple human readable name |
| `iqn` | iscsi | The iSCSI qualified name |
| `state` | iscsi | State of the iSCSI target ("online", "offline") |
| `auth` | iscsi | Optional authentication type ("none", "chap") |
| `targetchapuser` | iscsi | Optional CHAP user authentication |
| `targetchapsecret` | iscsi | Optional CHAP secret authentication |
| `interfaces` | iscsi | List of network interfaces that target is available |
| `wwn` | fc | Worldwide name for this target |
| `port` | fc | Physical location of the port |
| `mode` | fc | Mode of this port (initiator or target) |

| Property | Protocol | Description |
|---|---|---|
| speed | fc | Negotiated speed of this port |
| discovered_ports | fc | Number of discovered remote initiator ports |
| alias | srp | Alias for the SRP target |
| eui | srp | Extended unique identifier for this target |

The following properties are used for getting iSCSI target group information.

**TABLE 53** Target Group Properties

| Property | Type | Description |
|---|---|---|
| protocol | string | The target group protocol: FC, iSCSI, or SRP |
| name | string | The iSCSI target group name |
| targets | array | A list of iSCSI target IQN group members |

# List Targets

Lists all of the SAN targets of the specified protocol available on the appliance.

Example Request:

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

Example Response:

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

# Get Target Details

Gets properties from a single target. The target can be selected by using the "iqn" property or by using "alias=*alias*".

Example Request:

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

Example Response:

```
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": ""
    }
}
```

# Create a Target

Creates a new target. The request body has a JSON object with a single `name` property that is the name of the new iSCSI target group.

Example Request:

```
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"}
```

Example Response:

```
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"]
    }
}
```

# Modify a Target

Modifies an existing iSCSI target. The request body contains a JSON object that contains the iSCSI target properties that are modified. HTTP status 202 (Accepted) is returned on success. The response body contains the resulting iSCSI target properties for the target encoded in a JSON object.

Example Request:

```
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"}
```

Example Response:

```
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"]
    }
}
```

## Delete a Target

Removes a SAN target from the system.

Example Request:

```
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=
```

Successful Delete returns HTTP code 204 (No Content):

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

# Target Groups

Target groups are collections of targets. The target groups commands are listed in the table below.

The target groups commands take the following URI parameters:

*protocol* The NAS protocol for the initiator: `fc`, `iscsi`, or `srp`

*target-group* The name of the target group

**TABLE 54** Target Groups Commands

| Request | Append to Path /san/v1.0 | Description |
|---|---|---|
| GET | /*protocol*/target-groups | List all SAN target group for the given protocol: `fc`, `iscsi`, or `srp` objects |
| GET | /*protocol*/target-groups/*target-group* | Get the specified SAN target group for the given protocol: `fc`, `iscsi`, or `srp` properties |
| POST | /*protocol*/target-groups | Create a new SAN target group for the given protocol: `fc`, `iscsi`, or `srp` |
| PUT | /*protocol*/target-groups/*target-group* | Modify the specified SAN target group for the given protocol: `fc`, `iscsi`, or `srp` object |
| DELETE | /*protocol*/target-groups/*target-group* | Destroy the specified target-group object |

# List Target Groups

Lists all of the target groups available for an appliance. On success, HTTP status 200 (OK) is returned and the body contains a JSON object with a property named `groups` that contains an array of target group objects.

Example Request:

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

Example Response:

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

# Get Target Group

Gets a single target group. The request takes a single URI parameter, which is the target group name. The response body contains a JSON object property named `group` that contains the target group properties.

Example Request:

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

Example Response:

```
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"]
    }
}
```

# Create a Target Group

Creates a new iSCSI target group. The request body is a JSON object with a single `name` property that is the name of the new group.

Example Request:

```
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"]}
```

Example Response:

```
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"]
    }
}
```

# Delete a Target Group

Deletes an existing target group.

Example Request:

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

Successful delete returns HTTP status 204 (No Content):

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

# Service Commands

The Service RESTful API is used to list and manage software services running on the appliance.

## Service Commands

The following service commands are available.

**TABLE 55** Service Commands

| Request | Append to Path /service/v1 | Description |
|---|---|---|
| GET | Use only /service/v1 | List service commands |
| GET | /services | List all services |
| GET | /services/*service* | Get configuration and status for the specified service |
| PUT | /services/*service* | Modify the configuration of the specified service |
| PUT | /services/*service*/enable | Enable the specified service |
| PUT | /services/*service*/disable | Disable the specified service |

## List Services

This command returns the list of configurable services available on the storage appliance along with their enabled status. HTTP status 200 (OK) is returned for a successful command.

Example Request:

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

Example Response:

```
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"
    }]
}
```


# Get Service

This command gets the details from a single service including its state and its configuration.

Example Request:

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

Example Response:

```
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
    }
}
```

# Change Service State

This command changes the state of a given service. The following URI parameters are used:

*service* Name of the service

*state* New service state: `enable` or `disable`

Example Request:

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

Successful response returns HTTP status 202 (Accepted). The service can also be enabled or disabled by sending a JSON request to the service.

Example request using JSON:

```
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"}
```

To disable the service send the following JSON:

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

# Modify Service Configuration

Configuration properties on a specified service can be modified by sending a `PUT` request with the new property values defined in the header. Some services may have sub-resources, and they can also be modified by following the href defined in the sub-resource.

Example Request:

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

{"port": 218}
```

Successful response returns HTTP status of 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
    }
}
```

# Service Resources

Some services have sub-resources. See the data returned for each service or the list of service commands to see what sub-resources are available.

**TABLE 56** Service Sub Resource Commands

| Request | Path | Description |
|---|---|---|
| GET | /services/*service*/*resource* | List service sub-resource |
| PUT | /services/*service*/*resource*/*href* | Modify sub-resource |
| POST | /services/*service*/*resource* | Create a new sub-resource |
| DELETE | /services/*service*/*resource*/*href* | Destroy an sub-resource |

Each of these commands follow the same pattern as other RESTful API commands where `GET` is used to list or get a specified sub-resource type, `POST` is used to create a new sub-resource type, `PUT` is used to modify the sub-resource and `DELETE` is used to destroy the specified sub-resource.

For a list of sub-resources and properties and commands available for each sub-resource, see the CLI "configuration services" documentation.

# RESTful API Storage Service

The RESTful API Storage service is used to view configuration and manage aspects of storage pools, projects, filesystems and LUNs. It also manages snapshots and replication.

## Storage Pool Operations

In the Oracle ZFS Storage Appliance, NAS is configured in pools that characterize the same data redundancy characteristics across all LUNs and filesystem shares. In this version of the NAS API, pool operations are used to obtain the appliance storage configuration.

**TABLE 57** Storage Pool Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /pools | List all storage pools |
| GET | /pools/*pool* | Get storage pool details |
| POST | /pools | Configure a new storage pool |
| PUT | /pools/*pool* | Add or remove storage from a pool |
| PUT | /pools/*pool*/scrub | Start a data scrub on the specified pool |
| DELETE | /pools/*pool*/scrub | Stop any data scrub job on the specified pool |
| DELETE | /pools/*pool* | Unconfigure the specified storage pool |

## List Pools

This command lists the properties of all storage pools on the system. HTTP status 200 (OK) is returned for a successful command. The HTTP body contains a list of JSON objects describing each pool. The names of the properties are shown in the following table.

**Note -** The `depth` query parameter and the `match_`*property-name*=*value* query parameter are not supported.

**TABLE 58** Storage Pool Properties

| Property | Type | Description |
|---|---|---|
| `pool` | string | The target pool name |
| `profile` | string | Data device profile |
| `state` | string | Pool state: `online`, `offline`, `exported`) |
| `asn` | string | Serial number of the appliance that owns the pool |
| `peer` | string | In a clustered system, the ASN of the peer node |
| `owner` | string | Hostname of the system that owns the pool |

Example Request:

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

Example Response:

```
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"
    }]
}
```

# Get Pool

This command returns the properties from a single storage pool along with storage usage information for the pool. HTTP status 200 (OK) is returned for a successful command.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Configure Pool

Configures a pool. For the parameters needed to create a pool, see the CLI configuration storage command. A dry run request to create a pool can be done that returns the available property names and values. This is done by setting the `props` query parameter properties to `true`.

Example Request:

```
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",
}
```

Example Response:

```
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"
    }]
}
```

Example Request (to create a pool that uses 8 disks from chassis [1]):

```
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
}
```

Example Response:

```
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
        }
    }
}
```

# Add Storage to a Pool

This command is similar to create or configure a pool. Add storage adds additional storage devices to an existing pool. Send href *pool*/add with the body containing the desired number of storage devices to add to the pool.

Example Request:

```
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
}
```

Example Response:

```
HTTP/1.1 202 Accepted
```

# Remove Storage from a Pool

This command is similar to add storage to a pool. Remove storage removes cache and log storage devices from an existing pool. Send href *pool*/remove with the body containing the desired type, chassis number, and number of storage devices to remove from the pool.

Example Request:

```
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
}
```

Example Response:

```
HTTP/1.1 202 Accepted
```

To display the number of devices that can be removed, set the `props` query parameter to `true`.

Example Request:

```
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
```

Example Response:

```
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"
        }
    ]
}
```

# Pool Scrub

Sending a *pool*/scrub `PUT` or `DELETE` requests starts a pool scrub or stops a running scrub job respectively. For details, see the CLI command "configuration storage scrub".

# Unconfigure Pool

This command removes a pool from the system.

Request to Delete a Pool:

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

Example Response:

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

# Project Operations

All project operations can be scoped to a given pool. Commands that operate across all projects append /projects to the URI, and commands that operate on a single project append /projects/*project*.

**TABLE 59** Project Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /projects | List all projects |
| GET | /pools/*pool*/projects | List projects |
| GET | /pools/*pool*/projects?snaps=true | List all projects, including snapshots |
| GET | /pools/*pool*/projects/*project* | Get project details |
| POST | /pools/*pool*/projects | Create a project |
| PUT | /pools/*pool*/projects/*project* | Modify a project |
| DELETE | /pools/*pool*/projects/*project* | Destroy a project |
| GET | /pools/*pool*/projects/*project*/usage/groups | Get project group usage |
| GET | /pools/*pool*/projects/*project*/usage/groups/*group* | Get project usage for the specified group |
| GET | /pools/*pool*/projects/*project*/usage/users | Get project user usage |
| GET | /pools/*pool*/projects/*project*/usage/users/*user* | Get project usage for the specified user |

The following table shows the list of editable properties within a project resource.

**TABLE 60** Project Properties

| Property | Type | Description |
|---|---|---|
| `aclinherit` | string | ACL inheritance behavior ("discard", "noallow", "restricted", "passthrough", "passthrough-x", "passthrough-mode-preserve") |
| `aclmode` | string | ACL behavior on mode change ("discard", "mask", "passthrough") |
| `atime` | boolean | Update access time on read flag |
| `canonical_name` | string | Canonical name |
| `checksum` | string | Block checksum ("fletcher2", "fletcher4", "sha256") |
| `compression` | string | Data compression setting ("off", "lzjb", "gzip-2", "gzip", "gzip-9") |
| `copies` | number | Number of additional replication copies |
| `creation` | datetime | Date and time of project (or LUN, filesystem) creation |
| `dedup` | boolean | Data deduplication flag |
| `default_group` | string | Project default filesystem group: "other" |

| Property | Type | Description |
|---|---|---|
| default_permissions | string | Project default filesystem permissions "700" |
| default_sparse | boolean | Project default LUN sparse data flag |
| default_user | string | Project default filesystem user: "nobody" |
| default_volblocksize | number | Project default LUN blocksize: 8192 |
| default_volsize | number | Project default LUN Size |
| exported | boolean | Exported flag |
| logbias | string | Synchronous write bias ("latency", "throughput") |
| mountpoint | string | Share mountpoint default "/export/proj-01" |
| name | string | Project name |
| nbmand | boolean | Non-blocking mandatory locking flag |
| nodestroy | boolean | Prevent destruction flag |
| quota | number | Project quota size in bytes |
| origin | string | Clone origin |
| pool | string | Pool names |
| readonly | boolean | Data is read only if set to true |
| recordsize | string | Database record size "128k" |
| reservation | number | Data reservation size |
| rstchown | boolean | Restrict ownership change flag |
| secondarycache | string | Secondary cache usage ("all", "metadata", "none") |
| sharedav | string | HTTP share ("off", "rw", "ro") |
| shareftp | string | FTP share ("off", "rw", "ro") |
| sharenfs | string | NFS share ("off", "on", "ro", "rw") |
| sharesftp | string | SFTP share ("off", "rw", "ro") |
| sharesmb | string | SMB/CIFS share ("off", "rw", "ro") |
| sharetftp | string | TFTP share ("off", "rw", "ro") |
| snapdir | string | .zfs/snaphsot visibility ("hidden", "visible") |
| snaplabel | string | Scheduled snapshot label |
| vscan | boolean | Virus scan flag |

# List Projects

This command lists all of the projects in a given pool. The request takes a single URI parameter, which is the storage pool name. Each returned project contains the list of modifiable properties listed above as well as the pool name, creation time, loading state, replication actions, and data usage.

**Note -** The `depth` query parameter and the `match_`*property-name*`=`*value* query parameter are not supported.

Request parameters: `filter` – A simple string match filter that requires a property within the project to contain the same filter string within its value.

Example Request:

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

On a successful `get`, an HTTP code 200 (OK) is returned along with an array of project properties in JSON format.

Example Result:

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

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

A list of all projects across all pools is also supported; the URI would contain only the /projects path.

Example Request to get all projects with `backup` as part of its properties:

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

# Get Project Properties

This command lists the properties for a single project in a given pool. A successful `get` returns HTTP Code 200 (OK) along with the project properties in JSON format.

Example Request to list the project named `proj-01` in the `zfs-storage-1` pool:

```
GET /api/storage/v1/pools/p1/projects/proj-01 HTTP/1.1
```

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

Example Response:

```
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
    }
}
```

# Create Project

The create project command creates a project with a given name residing in the given storage pool. The request takes a single URI parameter, which is the storage pool name. The new project with default properties is returned.

JSON body request parameters:

- `name` – The project name must be supplied to create a project.
- Project properties – Any of the project properties can be set as the new project's initial values.

Example request to create a project named `proj-01`:

```
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"
}
```

Successful creation returns HTTP status 201 (Created) with the location header containing the URI of the new project. The body contains all of the project properties in JSON format.

Example Results:

```
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",
        ...
    }
}
```

# Modify Project

The modify project command changes the attributes of an existing project. The following URI parameters are used:

*pool* Storage pool name

*project* Project name

Request parameters: project properties – Any of the project properties can be set as the new project's initial values.

Example request to change a project's name from `proj-01` to `new-name`:

```
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"
}
```

Successful response returns HTTP status 202 (Accepted) and lists all project properties.

Example Response:

```
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",
         ...
    }
}
```

## Delete Project

The delete project command removes a single project in a given pool. The following URI parameters are used:

*pool* Storage pool name

*project* Project name

To monitor the amount of space to be reclaimed in the storage pool if deferred update Asynchronous Dataset Deletion (OS8.7.0) has been accepted, enter the GET command for pools/*pool*. Note the amount of space for property `async_destroy_reclaim_space`. When the operation has completed, 0 (zero) is displayed.

Example Request:

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

## Project Usage

Get requests project usage resources can be used to get usage data per user or per group for the project.

# Filesystem Operations

Filesystem operations list and manage filesystem shares. All commands are scoped to a given storage pool or project.

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

**TABLE 61** Filesystem Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /filesystems | List all filesystems |

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /pools/*pool*/projects/*project*/filesystems | List specified filesystems |
| GET | /pools/*pool*/projects/*project*/filesystems?snaps=true | List all filesystems, including snapshots |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem* | Get filesystem details |
| POST | /pools/*pool*/projects/*project*/filesystems | Create a filesystem |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem* | Modify a filesystem |
| DELETE | /pools/*pool*/projects/*project*/filesystems/*filesystem* | Destroy a filesystem |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups | Get filesystem group usage |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups/*group* | Get filesystem usage for the specified group |
| POST | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups | Create a filesystem group quota |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/groups/*name* | Modify a filesystem group quota |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users | Get filesystem user usage |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users/*user* | Get filesystem usage for the specified user |
| POST | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users | Create a filesystem user quota |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/usage/users/*name* | Modify a filesystem user quota |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/shadow/errors | List Shadow Migration Errors |

Each filesystem contains properties from the project and has the following filesystem-specific properties.

**TABLE 62** Filesystem Properties

| Property | Type | Description |
|---|---|---|
| `casesensitivity` | string | Case Sensitivity setting: `mixed`, `sensitive`, or `insensitive` |
| `group` | string | The group name |
| `normalization` | string | Normalization |
| `permissions` | string | The filesystem permissions |
| `project` | string | The project name |
| `quota_snap` | boolean | Flag to include snapshots in the quota |
| `reservation_snap` | boolean | Flag to include snapshots in the reservation |
| `shadow` | string | Data migration source |
| `errors` | string | Data migration errors |
| `sharesmb_name` | string | Name of SMB share |
| `source` | object | Project inheritance properties |
| `usage` | object | File system usage information |
| `user` | string | The user name that owns the share |

| Property | Type | Description |
|---|---|---|
| `utf8only` | boolean | Flag to reject non-UTF-8 |

# List Filesystems

The list filesystems command shows all filesystems in a given pool or project.

**Note -** The `depth` query parameter and the `match_`*property-name=value* query parameter are not supported.

Request parameters: `filter` – A simple string match filter that requires a property within the project to contain the same filter string within its value.

The list filesystems command uses the following URI parameters:

*pool* Storage pool name

*project* Project name

Example Request:

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

Successful request returns HTTP status 200 (OK) along with an array of filesystem properties in JSON format.

Example Result:

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

A list of all filesystems across all pools and projects is also supported. In that case, the URI would be /api/storage/v1/filesystems.

Example Request to get all filesystems with the "abcd" string as part of its properties:

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

# Get Filesystem

The get filesystem command returns a single filesystem's properties in a given pool or project. The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *filesystem* | Filesystem name |

Example request to list project named `proj-01`:

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

Successful `get` returns HTTP status 200 (OK) along with the filesystem properties in JSON format.

Example Response:

```
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",
        }
}
```

# Create Filesystem

The create filesystem command creates a filesystem with a given name residing in the given storage pool or project. The new filesystem with default properties is returned.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *filesystem* | Filesystem name |

Request parameters:

- `name` – The filesystem name must be supplied to create a new filesystem.
- Filesystem properties – Any of the properties listed in filesystem properties or project properties can be set as initial values.

Example Request (to create a filesystem named `share-01` and owned by the user `admin1`):

```
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"
}
```

Successful creation returns HTTP status 201 (Created) with the Location header containing the URI of the new filesystem. The body contains all filesystem properties in JSON format.

Example Response:

```
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"
        ...
    }
}
```

# Modify Filesystem

The modify filesystem command changes the attributes of an existing filesystem. Successful response returns HTTP status 202 (Accepted) and lists all filesystem properties.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *filesystem* | Filesystem name |

Request parameters: filesystem properties – Any of the filesystem or project properties can be modified.

Example Request (to change a filesystem name from `share-01` to `new-name` and change the owner to `nobody`):

```
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",
}
```

Example Response:

```
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"
        ...
    }
}
```

# Delete Filesystem

The delete filesystem command removes a single filesystem in a given pool or project.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *filesystem* | Filesystem name |

To monitor the amount of space to be reclaimed in the storage pool, enter the GET command for pools/*pool*. Note the amount of space for property `async_destroy_reclaim_space`. When the operation has completed, 0 (zero) is displayed.

Example Request:

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

Successful delete returns HTTP status 204 (No Content).

Example Response:

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

## Filesystem Quota and Usage

User or group quotas can be created or modified with `POST` or `PUT` requests, respectively. `GET` requests to filesystem use resources are used to get usage data per user or per group for the project.

# LUN Operations

All LUN or volume operations are scoped to a given pool or project. The following LUN commands are available.

**TABLE 63** Volume Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /luns | List all LUNs |
| GET | /pools/*pool*/projects/*project*/luns | List LUNs |
| GET | /pools/*pool*/projects/*project*/luns?snaps=true | List all LUNs, including snapshots |
| GET | /pools/*pool*/projects/*project*/luns/*lun* | Get LUN details |
| POST | /pools/*pool*/projects/*project*/luns | Create a LUN |
| PUT | /pools/*pool*/projects/*project*/luns/*lun* | Modify a LUN |
| DELETE | /pools/*pool*/projects/*project*/luns/*lun* | Destroy a LUN |

The following table lists the LUN properties. Volumes can also inherit or override project properties.

**TABLE 64** Volume Properties

| Property | Type | Description |
|---|---|---|
| `assignednumber` | number or list of numbers | The assigned LU number. If presented to multiple initiator groups, the type is a list of numbers.<br><br>If presented to multiple initiator groups, the ordering of `assignednumber` and `initiatorgroups` are aligned. For example, the first item in the `assignednumber` list pertains to the first item in the `initiatorgroups` list. |
| `fixednumber` | boolean | Flag to fix LU number at current value. |
| `initiatorgroups` | list of strings | The initiator group.<br><br>If the LUN is presented to multiple initiator groups, the ordering of `assignednumber` and `initiatorgroups` are aligned. For example, the first item in the `assignednumber` list pertains to the first item in the `initiatorgroups` list. |
| `lunguid` | string | STMF GUID. |
| `lunumber` | number or string | The LU number. Either a number or `auto`. |
| `project` | string | The project name (immutable). |
| `source` | object | Lists source of properties: `local` or `inherited`. |
| `sparse` | boolean | Flag to enable thin provisioning. |
| `status` | string | Logical unit status: `online` or `offline`. |
| `targetgroup` | string | The target group |
| `usage` | object | Lists LUN usage statistics |
| `volblocksize` | number | Volume block size |
| `volsize` | number | Volume size |
| `writecache` | boolean | Flag to enable write cache |

Some properties can be inherited from the project. The source object lists each of these properties and identifies whether the property is local to the LUN or is inherited from the project. By default these properties are inherited by the project. Once set, they are local to the LUN. The source object is immutable. To change the source back to inherited, the properties can be unset.

Example JSON Request to Unset Compression:

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

# List LUNs

The list LUNs command returns a list of LUNs available in a given pool or project.

**Note -** The `depth` query parameter and the `match_`*property-name*`=`*value* query parameter are not supported.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *filesystem* | Filesystem name |

Example request to list LUNs within project `proj-01`:

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

Successful `get` returns HTTP status 200 (OK) along with the LUN properties in JSON format.

Example Response:

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

# Get LUN

The get LUN command returns a single LUN's properties in a given pool or project.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |

*lun* LUN name

Example Request (to get a LUN named "vol-01"):

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

Successful `get` returns HTTP status 200 (OK) along with the LUN properties in JSON format.

Example Response:

```
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
    }
}
```

# Create a New LUN

This command creates a new LUN. You must supply a size or a cloning source for the new LUN.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |

Request Parameters:

- `name` – The LUN name must be supplied to create a new LUN.
- Volume properties – Any of the properties listed in LUN properties or project properties can be set as initial values.

Example Request:

```
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"
        }
}
```

Successful creation returns HTTP status 201 (Created) with the Location header containing the URI of the new LUN. The body contains all of the LUN properties in JSON format.

Example Results:

```
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",
        ...
    }
}
```

# Modify LUN

The modify LUN command changes the attributes of an existing LUN.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *lun* | LUN name |

Request parameters: volume properties – Any of the LUN or project properties can be modified.

Example request to change a LUN name from `vol-01` to `new-name`:

```
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",
```

```
}
```

Successful response returns HTTP status 202 (Accepted) and lists all LUN properties.

Example Response:

```
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",
        ...
    }
}
```

# Delete Lun

The delete LUN command removes a single LUN in a given pool or project.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Storage pool name |
| *project* | Project name |
| *lun* | LUN name |

To monitor the amount of space to be reclaimed in the storage pool, enter the GET command for pools/*pool*. Note the amount of space for property `async_destroy_reclaim_space`. When the operation has completed, 0 (zero) is displayed.

Example Request:

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

Successful `get` returns HTTP status 204 (No Content).

Example Response:

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

# Snapshot and Clone Operations

All snapshot operations are scoped to a given pool or project. Snapshot operations can also be scoped to the filesystem or LUN level.

- The URI for all project-based snapshot operations begins with: `/api/storage/v1/pools/`*pool*`/projects/`*project*
- The URI for all filesystem-based snapshot operations begins with: `/api/storage/v1/pools/`*pool*`/projects/`*project*`/filesystems/`*filesystem*
- The URI for all LUN-based snapshot operations begins with: `/api/storage/v1/pools/`*pool*`/projects/`*project*`/luns/`*lun*

**TABLE 65** Snapshot and Clone Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /snapshots | List all local snapshots |
| GET | /pools/*pool*/projects?snaps=true | List all projects, including snapshots |
| GET | /pools/*pool*/projects/*project*/filesystems?snaps=true | List all filesystems, including snapshots |
| GET | /pools/*pool*/projects/*project*/luns?snaps=true | List all LUNs, including snapshots |
| GET | /pools/*pool*/projects/*project*/snapshots | List all snapshots for a project |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots | List all snapshots for a filesystem |
| GET | /pools/*pool*/projects/*project*/luns/*lun*/snapshots | List all snapshots for a LUN |
| GET | /pools/*pool*/projects/*project*/snapshots/*snapshot* | Get project snapshot details |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Get filesystem snapshot details |
| GET | /pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Get LUN snapshot details |
| POST | /pools/*pool*/projects/*project*/snapshots | Create a project snapshot |
| POST | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots | Create a filesystem snapshot |
| POST | /pools/*pool*/projects/*project*/luns/*lun*/snapshots | Create a LUN snapshot |
| PUT | /pools/*pool*/projects/*project*/snapshots/*snapshot* | Modify a project snapshot |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Modify a filesystem snapshot |
| PUT | /pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Modify a LUN snapshot |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/clone | Clone a filesystem snapshot |
| PUT | /pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot*/clone | Clone a LUN snapshot |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/rollback | Rollback data to the given filesystem snapshot |
| PUT | /pools/*pool*/projects/*project*/lun/*lun*/snapshots/*snapshot*/rollback | Rollback data to the given LUN snapshot |
| DELETE | /pools/*pool*/projects/*project*/snapshots/*snapshot* | Destroy a project snapshot |
| DELETE | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Destroy a filesystem snapshot |

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| DELETE | /pools/*pool*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Destroy a LUN snapshot |
| GET | /pools/*pool*/projects/*project*/snapshots/*snapshot*/dependents | List project snapshot dependents |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot*/dependents | List filesystem snapshot dependents |
| GET | /pools/*pool*/projects/*project*/lun/*lun*/snapshots/*snapshot*/dependents | List LUN snapshot dependents |
| POST | /pools/*pool*/projects/*project*/automatic | Create a new project automatic snapshot object |
| POST | /pools/*pool*/projects/*project*/automatic?convert=true | Create a new project automatic snapshot object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| GET | /pools/*pool*/projects/*project*/automatic/*automatic* | Get the specified project automatic snapshot properties |
| GET | /pools/*pool*/projects/*project*/automatic | List all project automatic snapshot objects |
| PUT | /pools/*pool*/projects/*project*/automatic/*automatic* | Modify the specified project automatic snapshot object |
| PUT | /pools/*pool*/projects/*project*/automatic/*automatic*?convert=true | Modify the specified project automatic snapshot schedule object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| DELETE | /pools/*pool*/projects/*project*/automatic/*automatic* | Destroy the specified automatic object |
| DELETE | /pools/*pool*/projects/*project*/automatic/*automatic*?convert=true | Destroy the specified automatic snapshot schedule object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| POST | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic | Create a new filesystem automatic snapshot object |
| POST | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic?convert=true | Create a new filesystem automatic snapshot object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots. |

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| | | Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic* | Get the specified filesystem automatic snapshot properties |
| GET | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic | List all filesystem automatic snapshot objects |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic* | Modify the specified filesystem automatic snapshot object |
| PUT | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*?convert=true | Modify the specified filesystem automatic snapshot schedule object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| DELETE | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic* | Destroy the specified automatic snapshot schedule object |
| DELETE | /pools/*pool*/projects/*project*/filesystems/*filesystem*/automatic/*automatic*?convert=true | Destroy the specified filesystem automatic snapshot schedule object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| POST | /pools/*pool*/projects/*project*/luns/*lun*/automatic | Create a new LUN automatic snapshot |
| POST | /pools/*pool*/projects/*project*/luns/*lun*/automatic?convert=true | Create a new LUN automatic snapshot schedule. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| GET | /pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic* | Get the specified LUN automatic snapshot properties |
| GET | /pools/*pool*/projects/*project*/luns/*lun*/automatic | List all LUN automatic snapshot objects |
| PUT | /pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic* | Modify the specified LUN automatic snapshot object |
| PUT | /pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*?convert=true | Modify the specified LUN automatic snapshot schedule object. Existing auto-generated snapshots that do not fit |

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| | | new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |
| DELETE | /pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic* | Destroy the specified LUN automatic object |
| DELETE | /pools/*pool*/projects/*project*/luns/*lun*/automatic/*automatic*?convert=true | Destroy the specified LUN automatic snapshot schedule object. Existing auto-generated snapshots that do not fit new schedules are converted to manual snapshots.<br><br>Excluding the convert property causes existing auto-generated snapshots to be destroyed. |

# List Snapshots

Lists available snapshots on an appliance. Depending on the request URI, the list contains project, filesystem, or LUN snapshots.

**TABLE 66** List Snapshot Command Forms

| Command | Append to Path /api/storage/v1/pools/*pool*/projects/*project* |
|---|---|
| List project snapshots | /snapshots |
| List filesystem snapshots | /filesystems/*share*/snapshots |
| List LUN snapshots | /lun/*share*/snapshots |

Example Request:

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

Example Response:

```
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"
    }]
}
```

# Get Snapshot

View all information about a single snapshot. Returns HTTP status 200 (OK) on success.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Create Snapshot

The create snapshot command creates snapshots for projects, filesystems, or LUNs.

- Create Project Snapshot – `POST /pools/`*pool*`/projects/`*project*`/snapshots`
- Create Filesystem Snapshot – `POST /pools/`*pool*`/projects/`*project*`/filesystems/`*share*`/snapshots`
- Create Volume Snapshot – `POST /pools/`*pool*`/projects/`*project*`/luns/`*lun*`/snapshots`

Example Request:

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

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

Example Response:

```
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"
    }
}
```

# Rename Snapshot

Renames an existing snapshot.

- **Request URI** – Snapshot, the current snapshot name
- **Request Body** – JSON object with name parameter containing new snapshot name

Example Request:

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

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

Example Response:

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

# Clone Snapshot

Makes a new filesystem or LUN from an existing snapshot.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Source pool name |
| *project* | Source project name |
| *filesystem* | Source share name for filesystem snapshot |
| *lun* | Source share name for LUN snapshot |
| *snapshot* | Source snapshot name |

Clone a filesystem:

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

Clone a volume:

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

Request body contains a JSON object with the following properties.

**TABLE 67** Clone Snapshot Properties

| Property | Type | Description |
|---|---|---|
| `pool` | string | Destination clone pool name |
| `project` | string | Destination clone project name |

| Property | Type | Description |
|---|---|---|
| lun | string | Destination LUN name for LUN snapshot |

Example Request:

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

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

Example Response:

```
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"
   }
}
```

# Rollback Snapshot

The rollback snapshot causes the source file system or LUN to be modified back to its state when the snapshot was taken. Successful response returns HTTP status 202 (Accepted) as well as the snapshot properties in JSON format.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Source pool name |
| *project* | Source project name |
| *filesystem* | Source filesystem name for filesystem snapshot |
| *lun* | Source LUN name for LUN snapshot |
| *snapshot* | Source snapshot name |

Rollback a filesystem snapshot:

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

Rollback a LUN Snapshot:

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

Example Request:

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

Example Response:

```
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"
    }
}
```

# Delete a Snapshot

The `DELETE` snapshot command deletes a project, filesystem, or LUN snapshot from the system.

The following URI parameters are used:

| | |
|---|---|
| *pool* | Source pool name |
| *project* | Source project name |
| *filesystem* | Source filesystem name |
| *lun* | LUN name |
| *snapshot* | Source snapshot name |

Delete a project snapshot:

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

Delete a filesystem snapshot:

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

Delete a filesystem LUN:

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

If the snapshot has an NDMP hold on it, add `?confirm=true` to the `DELETE` command. Note, however, that this could adversely affect NDMP operations. For more information, see "NDMP Configuration" in *Oracle ZFS Storage Appliance Administration Guide, Release OS8.8.0*.

Example Request:

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

Example result if `?confirm=true` is not added:

If `?confirm=true` is not added when an NDMP hold exists on the snapshot, then the command will fail with the following output (lines are artificially broken for readability):

```
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"}}
```

# List Snapshot Dependents

Lists dependents for a filesystem or volume. The following URI parameters are used:

| | |
|---|---|
| *pool* | System storage pool name |
| *project* | Project name |
| *filesystem* | Filesystem name |
| *lun* | LUN name |
| *snapshot* | Snapshot name |

List filesystem dependents:

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

List volume dependents:

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

Example Request:

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

Example Response:

```
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"
        }
    ]
}
```

# Schema

Manages custom schema properties.

**TABLE 68** Schema Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /schema | List all NAS schema property objects |
| GET | /schema/*property* | Get the specified NAS schema property properties |
| POST | /schema | Create a new NAS schema property |
| PUT | /schema/*property* | Modify the specified NAS schema property object |
| DELETE | /schema/*property* | Delete the specified NAS schema property object |

Each custom schema property can be set on projects, filesystems, and LUNs by adding the prefix `custom:` to the custom property name.

For example, the following PUT body modifies a custom `int` property named `priority`:

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

**TABLE 69** Schema Parameters

| Parameter | Description |
|---|---|
| `property` | Name of property (immutable) |
| `description` | Property description (for browser interface) |
| `type` | Type (“String”, “Integer”, “PositiveInteger”, “Boolean”, “EmailAddress”, “Host”) |

# List Properties

Lists schema properties.

Example Request:

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

Example Result:

```
{
    "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"
    }]
}
```

# Get Property

Gets a schema property.

Example Request:

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

Example Result:

```
{
```

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

# Create Property

Creates a new schema property.

Example Request:

```
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"}
```

Example Result:

```
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"
    }
}
```

# Modify Property

Modifies a schema property.

Example Request:

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

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

Example Result:

```
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"
    }
}
```

## Delete Property

This command deletes a schema property.

Example Request:

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

Example Result:

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

# Replication

Remote replication facilitates replication of projects and shares between appliances.

---

**Note -** Replication is a licensed feature for certain models of Oracle ZFS Storage Appliance and the replication RESTful API manages that feature. The service is available from the following URI: `https://`*hostname*`:215/api/storage/v1/replication`. For license details, refer to the Oracle Software License Agreement (SLA) and Entitlement for Hardware Systems with Integrated Software Options and the Licensing Information User Manual for the software release.

---

The Replication RESTful API manages the following resources:

- **Replication Service** – The service that manages replication tasks.
- **Replication Target** – An appliance peer that receives and stores data replicated from another appliance peer (the source). This term also refers to a configuration object on the appliance that enables it to replicate to another appliance.
- **Replication Action** – A configuration object on a source appliance specifying a project or share, a target appliance, and policy options (including how often to send updates, whether to encrypt data on the wire, and so on).
- **Replication Package** – The target-side analog of an action; the configuration object on the target appliance that manages the data replicated as part of a particular action from a particular source. Each action on a source appliance is associated with exactly one package on a target appliance and vice versa. Loss of either object requires creating a new action/package pair (and a full replication update).

The API supplies replication operations for replication actions and replication packages. The service API is used to manage the replication service and replication sources and targets.

**TABLE 70** Replication Service Commands

| Request | Append to Path /api/service/v1/services | Description |
|---|---|---|
| GET | /replication | Get replication service state properties |
| PUT | /replication/enable | Enable the replication service |
| PUT | /replication/disable | Disable the replication service |

# Get Replication Service

Gets the state of the replication service.

Example Request:

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

Example Results:

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

## Modify Replication Service State

The replication service state can be modified like any other service. See the Service RESTful API for more information.

# Replication Targets

The following table shows the available replication target commands.

**TABLE 71** Replication Target Commands

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| POST | /replication/targets | Create a new replication target |
| GET | /replication/targets/*target* | Get the specified replication target properties |
| GET | /replication/targets | List all replication target objects |
| PUT | /replication/targets/*target* | Modify the specified replication target object |
| DELETE | /replication/targets/*target* | Destroy the specified target object |

## List Replication Targets

Lists all of the available replication targets on a system.

Example Request:

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

```
Accept: application/json
```

Example Response:

```
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"
        }]
}
```

# Get Replication Target

This command lists the details of a single replication target. The target is accessed by its hostname.

Example Request:

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

Example Response:

```
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
        }
}
```

# Create Replication Target

Creates a new replication target for remote replication.

Example Request:

```
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"}
```

Example Response:

```
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"
    }
}
```

## Delete Replication Target

This command deletes an existing replication target.

Example Request:

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

Successful delete returns HTTP status 204 (No Content).

Example Response:

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

# Replication Actions

Replication actions define the rules for replicating data to replication targets. The following commands manage the replication actions.

## Using the Flat Action Interface

Requests to manage replication actions can be made directly to an appliance, without specifying a project or share.

The following table lists the base commands for managing replication actions.

**TABLE 72** Base Action Interface

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /replication/actions | List all replication action objects |
| GET | /replication/actions/*ra_id* | Get the specified replication action properties |
| PUT | /replication/actions/*ra_id* | Modify the specified replication action object |
| DELETE | /replication/actions/*ra_id* | Delete the specified replication action object |
| PUT | /replication/actions/*ra_id*/sendupdate | Start the selected replication action |
| PUT | /replication/actions/*ra_id*/cancelupdate | Stop the selected replication action |

The following table lists the commands for managing replication action schedules.

**TABLE 73** Accessing Action Schedules

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/schedules | List all replication action schedule objects |
| GET | /replication/actions/*ra_id*/schedules/*ra_schedule* | Get the specified replication action schedule properties |
| POST | /replication/actions/*ra_id*/schedules | Create a new replication action schedule |
| PUT | /replication/actions/*ra_id*/schedules/*ra_schedule* | Modify the specified replication action schedule object |
| DELETE | /replication/actions/*ra_id*/schedules/*ra_schedule* | Delete the specified replication action schedule object |

The following table lists the commands for replication automatic snapshot management.

**Note -** Share Level auto-snapshot schedules configured within Project Level replication actions cannot be accessed with the following commands. Project Level actions can have multiple auto snapshot schedules in multiple shares, and this interface does not provide a definitive way to identify all the combinations.

**TABLE 74** Accessing Replication Automatic Snapshot Configuration

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/autosnaps | Retrieve auto-snapshot configurations for the selected replication action |
| GET | /replication/actions/*ra_id*/autosnaps/*autosnaps_id* | Get the specified replication action auto-snapshot object |
| PUT | /replication/actions/*ra_id*/autosnaps | Modify the specified replication action auto-snapshot properties |
| PUT | /replication/actions/*ra_id*/autosnaps/*autosnaps_id* | Modify the specified replication action auto-snapshot object |
| DELETE | /replication/actions/*ra_id*/autosnaps/*autosnaps_id* | Delete the specified replication action auto-snapshot object |

The following table lists the command for Replication Action Statistics.

**TABLE 75** Accessing Replication Action Statistics

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/stats | Retrieve read-only replication statistics for the selected replication action |

# Replication Actions in Project, Filesystem, or LUN Context

Requests to manage replication actions can also be made in the context of a specific project, filesystem, or LUN.

The following table lists the base commands for managing replication actions.

- Project-based operations URIs begin with:

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

- Filesystem-based operations URIs begin with:

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

- LUN-based operations URIs begin with:

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

Append the following base commands to the desired context URI listed above to manage replication actions.

**TABLE 76** Project, Filesystem, or LUN Base Replication Action Interfaces

| Request | Append to Project, Filesystem, or LUN URI Listed Above | Description |
|---|---|---|
| GET | /replication/actions | List all replication action objects |
| GET | /replication/actions/*ra_id* | Get the specified replication action properties |
| POST | /replication/actions | Create a new replication action |
| PUT | /replication/actions/*ra_id* | Modify the specified replication action object |
| DELETE | /replication/actions/*ra_id* | Delete the specified replication action object |
| PUT | /replication/actions/*ra_id*/sendupdate | Start the selected replication action |
| PUT | `/replication/actions/`*ra_id*`/cancelupdate` | Stop the selected replication action |

The following table lists the base commands for managing replication schedules.

- Project-based operations URIs begin with:

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

- Filesystem-based operations URIs begin with:

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

- LUN-based operations URIs begin with:

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

Append the following base commands to the desired context URI listed above to manage replication schedules.

**TABLE 77** Project, Filesystem, or LUN Replication Action Schedules

| Request | Append to Project, Filesystem, or LUN URI Listed Above | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/schedules | List all replication action schedule objects |
| GET | /replication/actions/*ra_id*/schedules/*ra_schedule* | Get the specified replication action schedule properties |
| POST | /replication/actions/*ra_id*/schedules | Create a new replication action schedule |
| PUT | /replication/actions/*ra_id*/schedules/*ra_schedule* | Modify the specified replication action schedule object |
| DELETE | /replication/actions/*ra_id*/schedules/*ra_schedule* | Delete the specified replication action schedule object |

The following table lists the base commands for managing replication automatic snapshot configuration.

- Project-based operations URIs begin with:

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

- Filesystem-based operations URIs begin with:

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

- LUN-based operations URIs begin with:

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

Append the following base commands to the desired context URI listed above to manage replication automatic snapshot configuration.

**Note -** Share level auto-snapshot schedules configured within project level replication actions cannot be accessed with the following project-based operations. Project level actions can have multiple auto-snapshot schedules in multiple shares, and this interface does not provide a definitive way to identify all combinations.

**TABLE 78** Project, Filesystem, or LUN Replication Automatic Snapshot Configuration

| Request | Append to Project, Filesystem, or LUN URI Listed Above | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/autosnaps | Retrieve auto-snapshot configurations for a project/share's selected replication action |

| Request | Append to Project, Filesystem, or LUN URI Listed Above | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/autosnaps/*autosnaps_id* | Get a project/share's specified replication action auto-snapshot configuration |
| POST | /replication/actions/*ra_id*/autosnaps | Create a new project/share's level replication action auto-snapshot object |
| PUT | /replication/actions/*ra_id*/autosnaps | Modify a project/share's specified replication action's target auto-snapshot retention policy. |
| PUT | /replication/actions/*ra_id*/autosnaps/*autosnaps_id* | Modify the specified replication action auto-snapshot object |
| DELETE | /replication/actions/*ra_id*/autosnaps/*autosnaps_id* | Delete the specified replication action auto-snapshot object |

The following table lists the base command for accessing replication action statistics.

- Project-based operations URIs begin with:

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

- Filesystem-based operations URIs begin with:

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

- LUN-based operations URIs begin with:

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

Append the following base commands to the desired context URI listed above to access replication action statistics.

**TABLE 79** Accessing Replication Action Statistics

| Request | Append to Project, Filesystem, or LUN URI Listed Above | Description |
|---|---|---|
| GET | /replication/actions/*ra_id*/stats | Retrieve read-only replication statistics for the selected replication action |

# List Replication Actions

Gets a list of all available replication actions.

Example Request:

```
GET /api/storage/v1/replication/actions HTTP/1.1
```

```
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

Example Response:

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

{
    "actions": [{
        "href": ""
        ...
    }, {
        "href": "",
        ...
    }]
}
```

# Get Replication Action

The get replication action status command returns the status of a single replication action given by its ID.

Example Request:

```
GET /api/storage/v1/replication/actions/1438ed7f-aad3-c631-d869-9e85cd7f15b4 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

Example Response:

```
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
    }
}
```

Example Request:

The following replication action response shows an example recovery point objective (RPO) and related replica lag warning and alerts.

```
GET /api/storage/v1/replication/actions HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Content-Type:application/json
```

Example Response:

```
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"
    }
}
```

# Create Replication Action

Creates a new replication action.

Create Properties:

```
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
```

Example Request:

```
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"
}
```

Example Response:

```
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
    }
}
```

Creates a schedule for a replication action.

Example Request:

```
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"}
```

Example Response:

```
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
```

# Modify Replication Action

Modifies an existing replication action.

Example Request:

```
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}
```

Example Response:

```
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
    }
}
```

Example Request:

```
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
```

Example Response:

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


# Monitor Replication Action Progress

The get replication action status command returns the status of a single replication action given by its ID. Examine `state` and `state_description` to determine replication progress.

`state` property values:

- `sending`
- `idle`

`state_description` property values:

- `Connecting to replication target`
- `Receiving checkpoint from target`
- `Estimating size of update`
- `Building deduplication tables`

  This property value is only for deduplicated replication streams.

Example Request:

```
GET /api/storage/v1/replication/actions/1438ed7f-aad3-c631-d869-9e85cd7f15b4 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Accept: application/json
```

Example Response:

```
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"
    }
}
```

# Cancel Update

Cancels an in-progress replication update.

Example Request:

```
PUT /api/storage/v1/replication/actions/c141d88d-ffd2-6730-d489-b71905f340cc/
cancelupdate HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

Example Response:

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

# Send Update

Schedules a replication update to start as soon as possible.

Example Request:

```
PUT /api/storage/v1/replication/actions/c141d88d-ffd2-6730-d489-b71905f340cc/sendupdate
 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
```

Example Response:

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

# Delete a Replication Action

Deletes an existing replication action.

Example Request:

```
DELETE /api/storage/v1/replication/actions/e7e688b1-ff07-474f-d5cd-cac08293506e HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

Successful delete returns HTTP status 204 (No Content).

Example Response:

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

# Replication Packages

This section details replication package and source commands.

**TABLE 80** Replication Package Commands

| Request | Append to Path /api/storage/v1/replication | Description |
|---|---|---|
| GET | /packages | List all replication packages |
| GET | /packages/*package* | Get the specified replication package |
| PUT | /packages/*package* | Modify the specified replication package |
| DELETE | /packages/*package* | Destroy the specified replication package |
| PUT | /packages/*package*/cancelupdate | Run cancelupdate on the specified package |
| PUT | /packages/*package*/sever | Run sever on the specified package |
| PUT | /packages/*package*/pkgreverse | Run reverse on the specified package |
| PUT | /packages/*package*/clone | Clone the specified package |
| GET | /packages/*package*/clone/conflicts | List share property conflicts |
| GET | /packages/*package*/projects | List package projects |
| GET | /packages/*package*/projects/*project* | Get package project |
| PUT | /packages/*package*/projects/*project* | Modify package project |
| GET | /packages/*package*/projects/*project*/usage/groups | Get package project group usage |
| GET | /packages/*package*/projects/*project*/usage/users | Get package project users usage |
| GET | /packages/*package*/projects/*project*/snapshots | List all snapshot objects |
| GET | /packages/*package*/projects/*project*/snapshots/*snapshot* | Get the specified snapshot properties |
| DELETE | /packages/*package*/projects/*project*/snapshots/*snapshot* | Destroy the specified snapshot object |
| PUT | /packages/*package*/projects/*project*/snapshots/*snapshot* | Rename the package project snapshot |
| GET | /packages/*package*/projects/*project*/automatic | List all package project automatic snapshot objects |
| GET | /packages/*package*/projects/*project*/automatic/*automatic* | Get the specified package project automatic snapshot properties |
| GET | /packages/*package*/projects/*project*/filesystems | List package filesystems |

| Request | Append to Path /api/storage/v1/replication | Description |
|---|---|---|
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem* | Get package filesystem |
| PUT | /packages/*package*/projects/*project*/filesystems/*filesystem* | Modify package filesystem |
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem*/usage/groups | Get package filesystem group usage |
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem*/usage/users | Get package filesystem users usage |
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Get the specified snapshot properties |
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots | List all snapshot objects |
| DELETE | /packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Destroy the specified snapshot object |
| PUT | /packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Rename the package filesystem snapshot |
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem*/automatic | List all package filesystem automatic snapshot objects |
| GET | /packages/*package*/projects/*project*/filesystems/*filesystem*/automatic/*automatic* | Get the specified package filesystem automatic snapshot properties |
| GET | /packages/*package*/projects/*project*/luns | List package LUNs |
| GET | /packages/*package*/projects/*project*/luns/*lun* | Get package LUN |
| PUT | /packages/*package*/projects/*project*/luns/*lun* | Modify package LUN |
| GET | /packages/*package*/projects/*project*/luns/*lun*/usage/groups | Get package LUN group usage |
| GET | /packages/*package*/projects/*project*/luns/*lun*/usage/users | Get package LUN users usage |
| GET | /packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Get the specified snapshot properties |
| GET | /packages/*package*/projects/*project*/luns/*lun*/snapshots | List all snapshot objects |
| DELETE | /packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Destroy the specified snapshot object |
| PUT | /packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Rename the package LUN snapshot |
| GET | /packages/*package*/projects/*project*/luns/*lun*/automatic | List all package LUN automatic snapshot objects |
| GET | /packages/*package*/projects/*project*/luns/*lun*/automatic/*automatic* | Get the specified package LUN automatic snapshot properties |

Replication sources and their corresponding packages can also be accessed using the following commands.

**TABLE 81** Replication Source Commands

| Request | Append to Path /api/storage/v1/replication/sources | Description |
|---|---|---|
| GET | Use only /api/storage/v1/replication/sources | List replication sources |
| GET | /*source* | List replication source details |
| GET | /*source*/packages/*package* | Get the specified replication package |
| PUT | /*source*/packages/*package* | Modify the specified replication package |

| Request | Append to Path /api/storage/v1/replication/sources | Description |
|---|---|---|
| DELETE | /*source*/packages/*package* | Destroy the specified replication package |
| PUT | /*source*/packages/*package*/cancelupdate | Run cancelupdate on the specified package |
| PUT | /*source*/packages/*package*/sever | Run sever on the specified package |
| PUT | /*source*/packages/*package*/pkgreverse | Run reverse on the specified package |
| PUT | /*source*/packages/*package*/clone | Clone the specified package |
| GET | /*source*/packages/*package*/clone/conflicts | List share property conflicts |
| GET | /*source*/packages/*package*/projects | List package projects |
| GET | /*source*/packages/*package*/projects/*project* | Get package project |
| PUT | /*source*/packages/*package*/projects/*project* | Modify package project |
| GET | /*source*/packages/*package*/projects/*project*/usage/groups | Get package project group usage |
| GET | /*source*/packages/*package*/projects/*project*/usage/users | Get package project users usage |
| GET | /*source*/packages/*package*/projects/*project*/snapshots/*snapshot* | Get the specified snapshot properties |
| GET | /*source*/packages/*package*/projects/*project*/snapshots | List all snapshot objects |
| DELETE | /*source*/packages/*package*/projects/*project*/snapshots/*snapshot* | Destroy the specified snapshot object |
| PUT | /*source*/packages/*package*/projects/*project*/snapshots/*snapshot* | Rename the package project snapshot |
| GET | /*source*/packages/*package*/projects/*project*/automatic | List all package project automatic snapshot objects |
| GET | /*source*/packages/*package*/projects/*project*/automatic/*automatic* | Get the specified package project automatic snapshot properties |
| GET | /*source*/packages/*package*/projects/*project*/filesystems | List package filesystems |
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem* | Get package filesystem |
| PUT | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem* | Modify package filesystem |
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/usage/groups | Get package filesystem group usage |
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/usage/users | Get package filesystem users usage |
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Get the specified snapshot properties |
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots | List all snapshot objects |
| DELETE | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Destroy the specified snapshot object |
| PUT | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/snapshots/*snapshot* | Rename the package filesystem snapshot |

| Request | Append to Path /api/storage/v1/replication/sources | Description |
|---|---|---|
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/automatic | List all package filesystem automatic snapshot objects |
| GET | /*source*/packages/*package*/projects/*project*/filesystems/*filesystem*/automatic/*automatic* | Get the specified package filesystem automatic snapshot properties |
| GET | /*source*/packages/*package*/projects/*project*/luns | List package LUNs |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun* | Get package LUN |
| PUT | /*source*/packages/*package*/projects/*project*/luns/*lun* | Modify package LUN |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun*/usage/groups | Get package LUN group usage |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun*/usage/users | Get package LUN users usage |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Get the specified snapshot properties |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots | List all snapshot objects |
| DELETE | /*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Destroy the specified snapshot object |
| PUT | /*source*/packages/*package*/projects/*project*/luns/*lun*/snapshots/*snapshot* | Rename the package LUN snapshot |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun*/automatic | List all package LUN automatic snapshot objects |
| GET | /*source*/packages/*package*/projects/*project*/luns/*lun*/automatic/*automatic* | Get the specified package LUN automatic snapshot properties |

# List Replication Sources

Lists all available replication sources.

Example Request:

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

Example Output:

```
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"
    }]
}
```

# List Replication Packages

Lists all of the replication packages.

Example Request:

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

Example Result:

```
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"
    }
  ]
}
```

# Modify Package

Modifies the package properties.

| Property | Type | Description |
|---|---|---|
| `enabled` | boolean | Current state of replication updates |

Example Request:

```
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}
```

Example Result:

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

Example Request:

```
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"}
```

Example Result:

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

# Delete Package

Destroys a replication package.

Example Request:

```
DELETE /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec
 HTTP/1.1
```

```
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

Successful delete returns HTTP status 204 (No Content).

Example Response:

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

# Cancel Update

Cancels an ongoing update for this package.

Example Request:

```
PUT /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/
cancelupdate HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

If no update is in progress, HTTP status 409 (Conflict) is returned.

Example Response:

```
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"
    }
}
```

Example Response:

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

# Clone Package

Clones the package project.

Example Request:

```
PUT /api/v1/storage/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/clone
 HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

Example Response:

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

Successful clone returns HTTP status 202 (Accepted). A helper command can be used to determine whether there are conflicts with the clone operation.

Example Clone Conflicts Request:

```
GET /api/storage/v1/replication/packages/8373d331-de60-e590-90e8-9ad69fcb4aec/clone/
conflicts HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic ab6rt4psMWE=
```

Clone/conflicts Returns Conflicts:

```
HTTP/1.1 200 OK
X-Zfssa-Replication-Api: 1.0
Content-Type: application/json
Content-Length: 58

{
    "conflicts": "There are no conflicts."
}
```

Properties:

```
Default settings:
            target_project = (unset)
       original_mountpoint = /export
       override_mountpoint = false
                mountpoint =
```

# Sever Package

Severs a replication connection and moves the package contents into new project. This action permanently severs this package and its replicated shares from the source system, making them local projects on this system. Subsequent replication updates in either direction requires defining new actions and sending a full update.

Example Request:

```
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"}
```

Success Response:

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

## Reverse Package

Reverses the direction of replication. This action disables replication for this package and moves the contents of this package into a new local project configured to replicate back to the source. Any metadata or data changes made on the source since the last successful update are lost when the new project is first replicated back to the source.

Example Request:

```
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"}
```

Success Response:

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

# Encryption

**Note -** Encryption is a licensed feature for certain models. For details, refer to the "Oracle Software License Agreement ("SLA") and Entitlement for Hardware Systems with Integrated Software Options" and the Licensing Information User Manual for the software release.

Oracle ZFS Storage Appliance offers transparent data encryption at the project as well as the individual share (filesystems and LUNs) level. The appliance includes a built-in local

keystore and also can connect to the Oracle Key Manager (OKM) system. Each encrypted project or share requires a wrapping key from either the local or OKM keystores. The data encryption keys are managed by the storage appliance and are stored persistently encrypted by the wrapping key from the local or OKM keystore.

The following tables describe the RESTful API requests available to manage local and OKM encryption.

**TABLE 82** Local Encryption

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /encryption/local | Get local keystore properties |
| PUT | /encryption/local | Modify local keystore properties |
| GET | /encryption/local/keys | Get local keys |
| GET | /encryption/local/keys/*key* | Get local key details |
| POST | /encryption/local/keys | Create a local key |
| DELETE | /encryption/local/keys/*key* | Destroy a local key |
| GET | /encryption/local/keys/*key*/dependents | List the shares dependent on this key |

**TABLE 83** OKM Encryption

| Request | Append to Path /api/storage/v1 | Description |
|---|---|---|
| GET | /encryption/okm | Get OKM keystore properties |
| PUT | /encryption/okm | Modify OKM keystore properties |
| GET | /encryption/okm/keys | Get OKM keys |
| GET | /encryption/okm/keys/*key* | Get OKM key details |
| POST | /encryption/okm/keys | Create an OKM key |
| DELETE | /encryption/okm/keys/*key* | Destroy an OKM key |
| GET | /encryption/okm/keys/*key*/dependents | List the shares dependent on this key |

# List All LOCAL Keys

Output:

```
{
    "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"
        }]
}
```

# List a LOCAL Key

Output:

```
{
    "key": {
        "href": "/api/storage/v1/encryption/local/keys/key-000",
        "cipher": "AES",
        "keyname": "key-1"
    }
}
```

# List All OKM Keys

Output:

```
{
    "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"
        }]
}
```

# System Commands

System commands are used to obtain system identity information and perform top-level system management commands. The following table lists the available system commands.

## Appliance System Commands

The following system commands are available.

**TABLE 84** Appliance System Commands

| Request | Append to Path /api/system/v1 | Description |
|---|---|---|
| GET | /version | List the appliance hardware and software version information. |
| PUT | /reboot | Reboot the appliance. Any queued platform updates will be applied during this reboot. |
| PUT | /reboot?skip_update=true | Reboot the appliance without applying any queued platform updates. |
| PUT | /reboot?diag=true | Diagnostic reboot: reboot the appliance, gathering additional diagnostic information in the process. |
| PUT | /poweroff | Turn off the appliance. |
| PUT | /restart | Restart the management interface and gather diagnostic information. |
| PUT | /factoryreset | Reset the appliance configuration back to factory settings. |
| GET | /disks | List all system disks. |
| GET | /disks/*disk* | List the specified system disk properties. |
| GET | /memory | System memory status report. |

# Get Version

This command returns a system structure that contains system identity information. HTTP status 200 (OK) is returned for a successful command.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Power Off System

This command performs a clean shutdown of the appliance. All data services become permanently unavailable unless the appliance is part of a cluster. To power the system back on

requires either service processor access or physical access to the power switch. This command runs asynchronously and returns an HTTP status of 202 (Accepted). The appliance must be monitored to follow the status of the actual command.

Example Request:

```
PUT /api/system/v1/poweroff HTTP/1.1
Host: zfs-storage.example.com
```

# Reboot System

This command performs a clean power cycle of the appliance. All services are temporarily unavailable. This command runs asynchronously and returns HTTP status 202 (Accepted). The appliance must be monitored to follow the status of the actual command.

**Note -** If a pending platform update is available to the appliance, it will be applied during this reboot. To perform a reboot without applying the pending platform update, use the `/reboot?skip_update=true` command instead.

Example Request:

```
PUT /api/system/v1/reboot HTTP/1.1
Host: zfs-storage.example.com
```

Example Request:

```
PUT /api/system/v1/reboot?skip_update=true HTTP/1.1
Host: zfs-storage.example.com
```

# Restart System Management

This command restarts the management interface and gathers diagnostic information. This command runs asynchronously and returns HTTP status 202 (Accepted). The appliance must be monitored to follow the status of the actual command.

Example Request:

```
PUT /api/system/v1/restart HTTP/1.1
Host: zfs-storage.example.com
```

# Diagnostic Reboot

This command reboots the appliance, gathering additional diagnostic information in the process. This command runs asynchronously and returns HTTP status 202 (Accepted). The appliance must be monitored to follow the status of the actual command.

**Note -** If there is a pending platform update available to the appliance, it will be not applied during this diagnostic reboot.

Example Request:

```
PUT /api/system/v1/reboot?diag=true HTTP/1.1
Host: zfs-storage.example.com
```

# Factory Reset

This command restores the appliance configuration to the original factory settings. All configuration changes are lost, and the appliance must be taken through initial setup as when first installed. This command runs asynchronously and returns HTTP status 202 (Accepted). The appliance must be monitored to follow the status of the actual command. Since this command can result in a loss of all configuration data, the query parameter `?confirm=true` must be added for the command to succeed.

Example Request:

```
PUT /api/system/v1/factoryreset?confirm=true HTTP/1.1
Host: zfs-storage.example.com
```

# System Support Bundles

The following support bundle commands are available.

**TABLE 85** Support Bundle Commands

| Request | Append to Path /api/system/v1 | Description |
|---|---|---|
| GET | /bundles | List all support bundles |
| GET | /bundles/*bundle* | Get the specified bundle data or properties |

| Request | Append to Path /api/system/v1 | Description |
|---|---|---|
| POST | /bundles | Build a support bundle and upload it to Oracle Support. |
| PUT | /bundles/*bundle*/retry | Retry upload of the specified bundle |
| PUT | /bundles/*bundle*/cancel | Cancel upload of the specified bundle |
| PUT | /bundles/*bundle*/send | Upload the specified bundle to Oracle Support with an optional SR number. |
| DELETE | /bundles/*bundle* | Destroy the specified bundle |

# Create Support Bundle

Creates a new support bundle to help resolve a service request. A Service Request (SR) number must be supplied to associate the support bundle with the open service request and send it to Oracle Support. The SR number must be in `3-`*nnnnnnnnnn* format. For the support bundle to be automatically uploaded to Oracle Support, the Phone Home settings must be registered with valid MOS credentials that have upload permissions.

Example Request:

```
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"}
```

Example Response:

```
HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
```

If a Service Request Number (SRN) is not provided, the system will build a local bundle instead.

Example Request:

```
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
```

Example Response:

```
{
  "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"
  }
}
```

# List Support Bundles

This command lists all support bundles being processed or collected by the system. After a support bundle is uploaded to Oracle Support, the support bundle is removed from the system.

Example Request:

```
GET /api/system/v1/bundles HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

Example Result:

```
{
    "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"
    }],
}
```

# Get Support Bundle

Gets properties from a single bundle.

Example Request:

```
GET /api/system/v1/bundles/9604155c-928b-cf97-c826-cda9fc17ac57 HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

Example Result:

```
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"
    }
}
```

# Cancel Support Bundle

This command cancels automatic upload of a support bundle.

Example Request:

```
PUT /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3/cancel HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
```

Example Response:

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

# Retry Support Bundle Upload

This command creates a new bundle upload job that attempts to upload a bundle to Oracle Support. The get bundle command can be used to monitor the status of the support bundle upload.

Example Request:

```
PUT /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3/retry HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
```

Example Response:

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

To retry a bundle upload using a different Service Request (SR) number, use the `send` command. If an SR number is not provided, the system will retry the upload using the original SR number.

**Note -** An SR number is required when running `send` on a locally generated bundle, or else an error will be thrown.

Example Request:

```
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"}
```

Example Response:

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

# Upload Support Bundle

A support bundle that is not automatically uploaded to Oracle Support can be uploaded manually.

**Note -** An SR number is required when running `send` on a locally generated bundle, or else an error will be thrown.

Example Request:

```
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"}
```

Example Result:

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

# Delete Support Bundle

This command removes a support bundle from the appliance.

Example Request:

```
DELETE /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3 HTTP/1.1
Authorization: Basic abhadbfsMWE=
Host: zfs-storage.example.com:215
```

Example Response:

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

# System Updates

These commands manage system update images.

**TABLE 86** Update Commands

| Request | Append to Path /api/system/v1 | Description |
|---|---|---|
| GET | /updates | List all system updates |
| GET | /updates/*update* | Get the specified system update properties |
| GET | /update/platform | Show the update status of platform firmware (Refers to the Service Processor and System Board firmware on the controller) |
| GET | /update/firmware | Show the update status of component firmware (Refers to disk and SSD firmware and also Disk Enclosure IOM firmware) |

| Request | Append to Path /api/system/v1 | Description |
|---|---|---|
| PUT | /updates/*update* | Modify update settings |
| PUT | /updates/*update*/upgrade | Upgrade to the specified update image |
| PUT | /updates/*update*/check | Run upgrade health checks for the specified update image |
| PUT | /updates/*update*/rollback | Rollback to the specified update image |
| PUT | /updates-apply | Apply deferred incompatible updates |
| DELETE | /updates/*update* | Destroy the specified system update |
| POST | /updates | Load an update image onto the appliance |

**TABLE 87** System Update Properties

| Property | Type | Description |
|---|---|---|
| `version` | String | Update media version |
| `release_date` | DateTime | Update release date |
| `install_date` | DateTime | Update latest installation date; if not installed, date of download to appliance |
| `status` | String | Update media status (immutable) |
| `update_deferred` | ChooseOne | Deferred setting: `onreboot` or `onrequest` |

Deferred updates notice:

```
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.
```

# List System Updates

Example request to get system updates:

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

Example Response:

```
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"
    }]
}
```

# Get System Update

Gets properties for a single update image.

Example Request:

```
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
```

Example Response:

```
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"
    }
}
```

# Get Platform Firmware Update Status

Gets the update status for pending platform firmware updates. Platform Firmware is a collective term that refers to the Service Processor and System Board firmware on the controller.

Example Request:

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

Example Response:

```
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"
    }
}
```

# Get Component Firmware Update Status

Gets the number of pending, failed, and in-progress component firmware updates. Component firmware a collective term that refers to disk and SSD firmware and also Disk Enclosure IOM firmware.

Example Request:

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

Example Response:

```
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
    }
}
```

# Upload System Update

This command uploads a new system update image.

Example Upload Command Using `curl`:

```
$ 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
```

After the image is uploaded and is unpacked, the properties of the update image are returned. The HTTP status is set to 201 (Created) on success and the relative location of the new image is returned in the location header.

Example Results:

```
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"
    }
}
```

# Upgrade

This command loads the update image and reboots the appliance to the specified update image. The specified image status should be equal to "previous" or the command fails.

Example Request:

```
PUT /api/system/v1/updates/nas@2013.08.25,1-0/upgrade
Host: zfs-storage.example.com:215
Authorization: Basic abcefgMWE=
Content-Length: 0
```

Example Response:

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

# Rollback

Rollback reboots the appliance to a previous update image.

Example Request:

```
PUT /api/system/v1/updates/nas@2013.08.24,1-0/rollback
Host: zfs-storage.example.com:215
Authorization: Basic abcefgMWE=
Content-Length: 0
```

Example Response:

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

# Delete Update Image

Removes an unused update image from the appliance.

Example Request:

```
DELETE /api/system/v1/updates/nas@2013.08.13,1-0 HTTP/1.1
Host: zfs-storage.example.com:215
Authorization: Basic abcefgMWE=
```

Example Response:

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

# RESTful API User Service

The RESTful API User service is used to configure local management users and user preferences on the appliance.

## User Service Commands

The following user service commands are available.

**TABLE 88** User Service Commands

| Request | Append to Path /api/user/v1 | Description |
|---|---|---|
| GET | Use only /api/user/v1 | List the user service commands |
| GET | /users | List summary information for all users |
| GET | /users/*user* | Get detail information about a specific user |
| DELETE | /users/*user* | Remove a local user from the system |
| POST | /users | Create a new local user, clone an existing user as new user, or add an administrator from a network directory |
| PUT | /users/*user* | Modify user properties |
| PUT | /users/*user*/preferences | Modify user preferences |
| GET | /users/*user*/preferences | Get user's preferences |
| POST | /users/*user*/exceptions | Create new user authorization exceptions |
| GET | /users/*user*/exceptions/*auth* | Get the specified user authorization exceptions properties |
| GET | /users/*user*/exceptions | List all user authorization exceptions objects |
| PUT | /users/*user*/exceptions/*auth* | Modify the specified user authorization exceptions object |
| DELETE | /users/*user*/exceptions/*auth* | Destroy the specified auth object |
| POST | /users/*user*/preferences/keys | Create a new user ssh keys |
| GET | /users/*user*/preferences/keys/*key* | Get the specified user ssh keys properties |
| GET | /users/*user*/preferences/keys | List all user ssh keys objects |

| Request | Append to Path /api/user/v1 | Description |
|---|---|---|
| PUT | /users/*user*/preferences/keys/*key* | Modify the specified ssh key for the given user |
| DELETE | /users/*user*/preferences/keys/*key* | Destroy the specified key object |

# List Users

Each user has the following summary properties available.

**TABLE 89** User Properties

| Property | Type | Description |
|---|---|---|
| `logname` | string | Username (immutable after creation) |
| `uid` | number | User ID, not enabled for Directory users |
| `fullname` | string | Full name |
| `initial_password` | string | Password |
| `require_annotation` | boolean | Flag to require session annotation |
| `roles` | string | This user's roles |
| `kiosk_mode` | boolean | Kiosk user |
| `kiosk_screen` | string | Kiosk screen |

Example Request:

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

Example Response:

```
{
    "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": [
            ]
        }
    }
}
```

# Get User

Gets detailed information about a user and includes user preferences and authorization exceptions. Each authorization exception type defines its own properties. The user preferences properties are shown.

**TABLE 90** User Preferences

| Property | Type | Description |
|---|---|---|
| `locale` | string | Locality |
| `login_screen` | string | Initial login screen |
| `session_timeout` | string | Session timeout in minutes |
| `advanced_analytics` | string | Make available advanced analytics statistics |

Each user can have ssh keys specified as part of the defined preferences.

**TABLE 91** SSH Key Properties

| Property | Type | Description |
|---|---|---|
| `type` | string | The type of SSH key: either RSA or DSA |
| `key` | string | The contents of the SSH key |
| `comment` | string | A comment associated with this SSH key |

Example Request:

```
GET /api/user/v1/users/admin1 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: application/json
```

Example Response:

```
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"]
    }
}
```

# Create User

To understand more about users and user types, see "Understanding Users and Roles" in *Oracle ZFS Storage Appliance Administration Guide, Release OS8.8.0*.

This command uses three forms:

- **Create a new user** – Creates a new user
- **Clone an existing user** – Clones a new user from an existing user
- **Add an administrator** – Allows a user defined in an enterprise directory to administer the appliance.

In all three cases, a `POST` request to users with JSON-formatted properties in the body is sent.

**TABLE 92** Create New User Properties

| Property | Type | Description |
|---|---|---|
| `logname` | string | New user's login name (required) |
| `uid` | number | Optional user ID |
| `fullname` | string | New user's full name (required) |

| Property | Type | Description |
|---|---|---|
| type | string | "local", "data", "nologin" (defaults to "local") |
| initial_password | string | Initial user password ("local" and "data" only) |
| require_annotation | boolean | Optional flag to require session annotation ("local" only) |

**TABLE 93** Clone User Properties

| Property | Type | Description |
|---|---|---|
| user | string | Source user name |
| uid | number | User ID, not enabled for Directory users |
| clonename | string | New clone's login name |
| fullname | string | New clone user's full name (not for Directory users) |
| password | string | New clone user's password (not for Directory or No-login users) |

**TABLE 94** Add Administrator Properties

| Property | Type | Description |
|---|---|---|
| type | string | Directory users |
| logname | string | Directory user login name |

**EXAMPLE 1** Creating a Local User

Example Request:

```
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"
}
```

Example Result:

```
{
    "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": [
            ]
        }
    }
}
```

**EXAMPLE 2** Creating a Directory User

Example Request:

```
POST /api/user/v1/users
{
    "type": "directory",
    "logname": "admin3"
}
```

Example Result:

```
{
    "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": [
            ]
        }
    }
}
```

**EXAMPLE 3** Creating a Data-only User

Example Request:

```
POST /api/user/v1/users
{
    "type": "data",
    "logname": "admin3",
    "initial_password": "password",
    "fullname": "Administrator",
    "uid": 5000000
}
```

Example Result:

```
{
    "user":
    {
        "href": "/api/user/v1/users/data",
        "logname": "admin3",
        "type": "data",
        "uid": 5000000,
        "fullname": "Administrator",
        "initial_password": "password"
    }
}
```

**EXAMPLE 4** Creating a No-login User

Example Request:

```
POST /api/user/v1/users
{
    "type": "nologin",
    "logname": "admin3",
    "fullname": "Administrator",
    "uid": 5000001
}
```

Example Result:

```
{
    "user":
    {
        "href": "/api/user/v1/users/admin3",
        "logname": "admin3",
        "type": "nologin",
        "uid": 5000001,
        "fullname": "Administrator"
    }
}
```

# Modify Users

Modifies user properties directly. User resources: exceptions, preferences, and ssh keys can be added, modified or removed. "UID" and "Type" are immutable once created.

Example Request:

```
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}
```

Example Result:

```
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"]
    }
}
```

# Delete Users

Deletes a user from the system.

Example Request:

```
DELETE /api/user/v1/users/admin1 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

Example Result:

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

# Workflow and Script Commands

This service is used to manage workflows. A workflow is a script that is uploaded to and managed by the appliance. Workflows can be parameterized and executed in a first-class fashion from either the browser interface or the command line interface. Workflows can also be executed as alert actions or at a designated time; thus, they can allow the appliance to be extended in ways that capture specific policies and procedures, and they can be used to formally encode best practices for a particular organization or application.

## Workflow and Script Service Commands

The following table shows the workflow service commands.

**TABLE 95** Workflow Service Commands

| Request | Append to Path /api/workflow/v1 | Description |
|---|---|---|
| GET | Use only /api/workflow/v1 | List the workflow service commands |
| POST | /workflows | Upload a new workflow onto the appliance |
| GET | /workflows | List all workflows |
| GET | /workflows/*workflow* | List the specified workflow properties |
| PUT | `/workflows/`*workflow* | Modify the specified workflow properties |
| PUT | /workflows/*workflow*/execute | Execute the specified workflow |
| DELETE | /workflows/*workflow* | Destroy the specified workflow |
| POST | /scripts | Upload and run a script |
| GET | /scripts | List all running scripts |
| GET | /scripts/*script* | Reconnect to a running script |
| DELETE | /scripts/*script* | Stop a running script |

## Upload Workflow

Uploads a workflow to the appliance.

Example Request:

```
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) }
};
```

Example Result:

```
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
    }
}
```

# List Workflows

Lists all workflows installed on an appliance. If the query parameter `showhidden=true` is set, the list includes workflows that are normally hidden.

Example Request:

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

Example Response:

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

```
}
```

# Get Workflow

Gets properties for a single workflow. In the header, if `Accept` is specified as `application/javascript`, it returns the content of the workflow, otherwise it returns workflow properties.

Example request where `Accept` is specified as `application/javascript`:

```
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
```

Example Response:

```
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.' );
     }
    };
```

Example request where `Accept` is specified as `application/json`:

```
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
```

Example Response:

```
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
    }
}
```

# Modify a Workflow

You can modify properties for a single workflow by sending a PUT request to a workflow resource.

Example Request:

```
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}
```

Example Response:

```
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": ""
    }
}
```

# Execute a Workflow

Executes a workflow script and return the results. Any workflow parameters must be passed in a JSON object within the body. On success HTTP status 202 (Accepted) is returned along with a JSON object with a single result property containing the workflow output.

Example Request:

```
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"}
```

Example Result:

```
HTTP/1.1 202 Accepted
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 34

{
    "result": "tweet tweet tweet\n"
}
```

# Delete Workflow

Deletes a workflow script from the appliance.

Example Request:

```
DELETE /api/workflow/v1/workflows/f4fe892f-cf46-4d6a-9026-cd0c0cce9971 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfs-storage.example.com:215
Accept: */*
```

Example Result:

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

# Upload and Run a Script

Uploads and runs a script on the appliance.

A root user can view and access all scripts uploaded to the appliance. A non-root user can only view and access their own scripts.

For more information on scripting, see "Working with CLI Scripting" in *Oracle ZFS Storage Appliance Administration Guide, Release OS8.8.0*.

This script lists all shares on the appliance.

Example Request:

```
$ 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
```

Example Result:

```
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
```

# List All Running Scripts

Use the following command to list all running scripts.

A root user can view and access all scripts uploaded to the appliance. A non-root user can only view and access their own scripts.

For more information on scripting, see "Working with CLI Scripting" in *Oracle ZFS Storage Appliance Administration Guide, Release OS8.8.0*.

Example Request:

```
$ 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: */*
```

Example Result:

```
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"
        }
    ]
}
```

# Reconnect to a Running Script

A root user can reconnect to any running script uploaded to the appliance. A non-root user can only reconnect to their own running scripts.

For more information on scripting, see "Working with CLI Scripting" in *Oracle ZFS Storage Appliance Administration Guide, Release OS8.8.0*.

Example Request:

```
$ 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
```

Example Result:

```
{
    "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": []
}
```

# Stop a Running Script

A root user can delete any running script uploaded to the appliance. A non-root user can only access and delete their own running scripts.

For more information on scripting, see "Working with CLI Scripting" in *Oracle ZFS Storage Appliance Administration Guide, Release OS8.8.0*.

Example Request:

```
$ 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: */*
```

Example Result:

```
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 Clients

Any HTTP client can be used as a RESTful client. Even the BUI can return RESTful API `GET` results by typing in a resource URL. Mozilla Firefox has a RESTful client module that can be installed to make RESTful requests (`https://addons.mozilla.org/en-us/firefox/addon/restclient/`). This module allows `PUT`, `POST`, and `DELETE` requests as well as the normal HTTP `GET` requests.

RESTful clients must use TLS protocols because the SSLv2/3 protocols are no longer supported. Curl clients must use `curl` version 7.34.0 or higher.

This section contains more detailed information about various RESTful clients.

## Curl Rest Client

Curl clients must use `curl` version 7.34.0 or higher. Two common CLI-based HTTP clients are `wget` and `curl`. This section shows several examples of using `curl` to make RESTful API calls and similar functionality can be accomplished using `wget`.

### Get Resource Data

This example shows how to use a simple HTTP `GET` request to obtain some JSON data:

```
$ 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"
    }
}
```

# Create a New Resource

This example shows how to send JSON data in a request to create a new resource:

```
$ 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": {}
      }
   }
}
```

# Modify an Existing Resource

This example modifies a user's session timeout:

```
$ 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": {}
    }
}
```

# Delete an Existing Resource

This command removes a user from the system:

```
$ 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 Client

A Python RESTful API client is provided along with a REST test library to aid in test development of RESTful services.

Example RESTful Client Program:

```
>>> 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')]
```

The opener can then be used to open requests that are already pre-authenticated and ready to send/receive JSON data.

## Get a Resource

The following Python code can be used to get data from any RESTful API resource.

Example GET:

```
>>> 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"

            }
    ]
}
```

# Create a Resource

Example Python code to create a new resource:

```
>>> 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
    }
}
```

# Modify a Resource

Example Python code to modify an existing resource:

```
>>> 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
    }
}
```

# Delete an Existing Resource

Example Python code to delete an existing resource:

```
>>> 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
```