Oracle® ZFS Storage Appliance RESTful API Guide, Release 2013.1.3.0
Ce logiciel et la documentation qui l’accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d’utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d’interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu’elles soient exemptes d’erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l’accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l’utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s’applique:

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d’applications de gestion des informations. Ce logiciel ou matériel n’est pas conçu ni n’est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d’applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l’utilisation de ce logiciel ou matériel pour ce type d’applications.

Oracle et Java sont des marques déposées d’Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d’autres propriétaires qu’Oracle.

Intel et Intel Xeon sont des marques ou des marques déposées d’Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d’Advanced Micro Devices. UNIX est une marque déposée d’The Open Group.

Ce logiciel ou matériel et la documentation qui l’accompagnent peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l’accès à des contenus, produits ou services tiers, ou à leur utilisation.
Cluster Links ......................................................................................... 57
Setup Cluster ......................................................................................... 58
Chassis ......................................................................................................... 58
  List Chassis ..........................................................................................  58
  Get Chassis Components ....................................................................... 60
  Get Hardware Component ....................................................................... 61
  Modify Component Property .................................................................... 63

Log Commands ............................................................................................. 65
  Log Commands ............................................................................................. 65
  List Logs ...................................................................................................... 65
  Get Log Entries ............................................................................................. 66
  Download Logs ............................................................................................. 68
  Download Log .............................................................................................. 68

Network Commands ............................................................................................. 69
  Networking Configuration ............................................................................... 69
  Network Datalinks ......................................................................................... 69
    List Network Datalinks ........................................................................... 71
    Get Network Datalink ............................................................................. 72
    Create Network Datalink ......................................................................... 72
    Modify Network Datalink ........................................................................ 73
    Delete Network Datalink ......................................................................... 74
  Network Devices ........................................................................................... 74
    List Network Devices ............................................................................. 75
    Get Network Device ............................................................................... 75
  Network Interfaces ......................................................................................... 76
    List Network Interfaces ........................................................................... 77
    Get Network Interface ........................................................................... 77
    Create Network Interface ....................................................................... 78
    Modify Network Interface ....................................................................... 79
    Delete Network Interface ....................................................................... 79
  Network Routes ............................................................................................. 80
    List Routes ............................................................................................. 80
    Get Route ............................................................................................... 81
    Add Route ............................................................................................... 82
    Delete Route ............................................................................................ 82

RESTful API Problem Service .......................................................................... 83
Get Target Group ................................................................. 108
Create a Target Group .......................................................... 109
Delete a Target Group ............................................................ 109

Service Commands .............................................................. 111
Service Commands ................................................................. 111
List Services ........................................................................ 111
Get Service .......................................................................... 115
Change Service State ............................................................ 116
Modify Service Configuration ................................................. 116
Service Resources ............................................................... 117

RESTful API Storage Service .................................................. 119
Storage Pool Operations ......................................................... 119
List Pools ............................................................................. 119
Get Pool ................................................................................ 120
Configure Pool .................................................................... 121
Add Storage to a Pool ............................................................ 123
Pool Scrub ........................................................................... 123
Unconfigure Pool ................................................................ 123
Project Operations ............................................................... 124
  List Projects ..................................................................... 126
  Get Project Properties ....................................................... 127
  Create Project .................................................................. 128
  Modify Project .................................................................. 129
  Delete Project .................................................................. 130
Project Usage ....................................................................... 130
Filesystem Operations .......................................................... 130
  List Filesystems ............................................................... 132
  Get Filesystem ................................................................. 133
  Create Filesystem ............................................................ 135
  Modify Filesystem ............................................................ 136
  Delete Filesystem ............................................................. 137
  Filesystem Quota and Usage ............................................. 137
LUN Operations ................................................................... 138
  List LUNS ....................................................................... 139
  Get LUN ......................................................................... 139
  Create a New LUN .......................................................... 141
  Modify LUN ..................................................................... 142
Contents

Delete Lun ................................................................................................................. 143
Snapshot and Clone Operations .................................................................................. 143
    List Snapshots ................................................................................................... 146
    Get Snapshot .................................................................................................... 147
    Create Snapshot ............................................................................................... 147
    Rename Snapshot ............................................................................................. 148
    Clone Snapshot .................................................................................................. 148
    Rollback Snapshot ............................................................................................ 150
    Delete a Snapshot ............................................................................................. 151
    List Snapshot Dependents ................................................................................... 151
Schema ........................................................................................................................ 152
    List Properties .................................................................................................. 153
    Get Property ...................................................................................................... 154
    Create Property ................................................................................................ 154
    Modify Property ................................................................................................ 155
    Delete Property ................................................................................................ 155
Replication .................................................................................................................. 155
    Get Replication Service ................................................................................... 156
    Modify Replication Service State .................................................................... 157
Replication Targets .................................................................................................... 157
    List Replication Targets .................................................................................. 157
    Get Replication Target ..................................................................................... 158
    Create Replication Target ............................................................................... 158
    Delete Replication Target ............................................................................... 159
Replication Actions ................................................................................................... 159
    List Replication Actions .................................................................................. 163
    Get Replication Action ..................................................................................... 163
    Create Replication Action ............................................................................... 164
    Modify Replication Action ............................................................................... 165
    Cancel Update .................................................................................................. 166
    Send Update ..................................................................................................... 166
    Delete a Replication Action ............................................................................. 166
Replication Packages ................................................................................................. 167
    List Replication Sources .................................................................................. 171
    List Replication Packages ............................................................................... 171
    Modify Package ................................................................................................ 172
    Delete Package ................................................................................................ 172
    Cancel Update .................................................................................................. 173
    Clone Package .................................................................................................. 173
    Sever Package .................................................................................................. 174
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow Service Commands</td>
<td>197</td>
</tr>
<tr>
<td>List Workflows</td>
<td>197</td>
</tr>
<tr>
<td>Get Workflow</td>
<td>198</td>
</tr>
<tr>
<td>Modify a Workflow</td>
<td>199</td>
</tr>
<tr>
<td>Execute a Workflow</td>
<td>200</td>
</tr>
<tr>
<td>Delete Workflow</td>
<td>200</td>
</tr>
<tr>
<td>Upload Workflow</td>
<td>201</td>
</tr>
<tr>
<td>RESTful Clients</td>
<td>203</td>
</tr>
<tr>
<td>Curl Rest Client</td>
<td>203</td>
</tr>
<tr>
<td>Get Resource Data</td>
<td>203</td>
</tr>
<tr>
<td>Create a New Resource</td>
<td>204</td>
</tr>
<tr>
<td>Modify an Existing Resource</td>
<td>204</td>
</tr>
<tr>
<td>Delete an Existing Resource</td>
<td>205</td>
</tr>
<tr>
<td>Python RESTful Client</td>
<td>205</td>
</tr>
<tr>
<td>Get a Resource</td>
<td>206</td>
</tr>
<tr>
<td>Create a Resource</td>
<td>206</td>
</tr>
<tr>
<td>Modify a Resource</td>
<td>207</td>
</tr>
<tr>
<td>Delete an Existing Resource</td>
<td>208</td>
</tr>
</tbody>
</table>
The 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 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: root
  X-Auth-Key: letmein-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: guigqoRE4g89ngb
  ```
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.

Example: A client makes some requests for a service that is running version number “2.1”.

<table>
<thead>
<tr>
<th>Request Version</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>False - Major number does not match</td>
</tr>
<tr>
<td>v2</td>
<td>True - Major number matches minor is backwards compatible</td>
</tr>
<tr>
<td>v2.1</td>
<td>True - Major and minor numbers match</td>
</tr>
<tr>
<td>v2.2</td>
<td>False - Major matches but minor is a newer revision</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>resources&gt;</td>
<td>List all resources</td>
</tr>
<tr>
<td>GET</td>
<td>resources/&lt;name&gt;</td>
<td>Get a JSON object describing the selected resource</td>
</tr>
<tr>
<td>POST</td>
<td>resources&gt;</td>
<td>Create a new resource</td>
</tr>
<tr>
<td>PUT</td>
<td>resources/&lt;name&gt;</td>
<td>Modify the selected resource</td>
</tr>
<tr>
<td>DELETE</td>
<td>resources/&lt;name&gt;</td>
<td>Delete the selected resource</td>
</tr>
</tbody>
</table>

### HTTP Response Body

All response data is encoded in JavaScript Object Notation (JSON) format as defined by RFC 4627 (http://tools.ietf.org/html/rfc4627.html). 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:

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

Some GET commands return a list of resources.
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 will take optional query parameters that will modify or enhance the data returned. See the documentation for each resource for details. Not every resource will support every query parameter. This section only documents the common query parameters that will be used when a resource does implement the specified query parameter.

**TABLE 2**  
Common Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>props=true</td>
<td>List property metadata for a resource (default is false)</td>
</tr>
</tbody>
</table>
### Query Parameter: props

The props query parameter can be used on many `GET` commands that return a single resource instance. It will return a 'props' data element that contains a list of property metadata.

**TABLE 3**  
Property Metadata Values

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Property name</td>
</tr>
<tr>
<td>label</td>
<td>Description of property</td>
</tr>
<tr>
<td>immutable</td>
<td>Flag indicating that property cannot be modified</td>
</tr>
<tr>
<td>type</td>
<td>Property type: String, Integer, Boolean...</td>
</tr>
<tr>
<td>choices</td>
<td>For enumerated properties, an array of available values</td>
</tr>
</tbody>
</table>

### 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

The start query parameter is supported on the same commands that support the limit parameter. It gives the start index that is used to begin the returned data. For resources that support time values, the index may be a time value such as "20130531T01:13:58".

### Appliance Errors

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

**JSON Fault Response:**

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

### TABLE 4  Common Error Codes

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR_INVALID_ARG</td>
<td>400</td>
<td>Invalid input argument</td>
</tr>
<tr>
<td>ERR_UNKNOWN_ARG</td>
<td>400</td>
<td>Extra unhandled input argument</td>
</tr>
<tr>
<td>ERR_MISSING_ARG</td>
<td>400</td>
<td>Required input argument missing</td>
</tr>
<tr>
<td>ERR_UNAUTHORIZED</td>
<td>401</td>
<td>This user is not authorized to execute command</td>
</tr>
<tr>
<td>ERR_DENIED</td>
<td>403</td>
<td>Operation denied</td>
</tr>
<tr>
<td>ERR_STATE_CHANGED</td>
<td></td>
<td>Conflict in system state</td>
</tr>
<tr>
<td>ERR_NOT_FOUND</td>
<td>404</td>
<td>The requested item was not found</td>
</tr>
<tr>
<td>ERR_OBJECT_EXISTS</td>
<td>409</td>
<td>Request creates an object that already exists</td>
</tr>
<tr>
<td>ERR_OVER_LIMIT</td>
<td>413</td>
<td>Input request too large to handle</td>
</tr>
<tr>
<td>ERR_UNSUPPORTED_MEDIA</td>
<td>415</td>
<td>Requested media type is not supported by request</td>
</tr>
<tr>
<td>ERR_NOT_IMPLEMENTED</td>
<td>501</td>
<td>Operation not implemented</td>
</tr>
<tr>
<td>ERR_BUSY</td>
<td>503</td>
<td>Service not available due to limited resources</td>
</tr>
</tbody>
</table>
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://zfssa.example.com: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>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/access/v1</td>
<td>Lists RESTful API service access points</td>
</tr>
<tr>
<td>POST</td>
<td>/api/access/v1</td>
<td>Creates a login session</td>
</tr>
<tr>
<td>DELETE</td>
<td>/api/access/v1</td>
<td>Logs out of a session</td>
</tr>
</tbody>
</table>

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: joedamn
X-Auth-Key: letmein
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: guigqpQRE4g99ngb

Example Response:

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

{
    "service": {
        "name": "appliance",
        "methods": [
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: letmein-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: guigqpORE4gB9ngb
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: guigqpORE4gB9ngb

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>
<thead>
<tr>
<th>Request</th>
<th>Path /api/alert/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/</td>
<td>List the alert service commands</td>
</tr>
<tr>
<td>POST</td>
<td>/thresholds</td>
<td>Create a new alert threshold watch</td>
</tr>
<tr>
<td>GET</td>
<td>/thresholds/&lt;threshold&gt;</td>
<td>Get the specified alert threshold watch properties</td>
</tr>
<tr>
<td>GET</td>
<td>/thresholds</td>
<td>List all alert threshold watch objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/thresholds/&lt;threshold&gt;</td>
<td>Modify the specified alert threshold watch object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/thresholds/&lt;threshold&gt;</td>
<td>Destroy the specified threshold object</td>
</tr>
<tr>
<td>POST</td>
<td>/actions</td>
<td>Create a new alert actions</td>
</tr>
<tr>
<td>GET</td>
<td>/actions/&lt;actions&gt;</td>
<td>Get the specified alert actions properties</td>
</tr>
<tr>
<td>GET</td>
<td>/actions</td>
<td>List all alert actions objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/actions/&lt;actions&gt;</td>
<td>Modify the specified alert actions object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/actions/&lt;actions&gt;</td>
<td>Destroy the specified actions object</td>
</tr>
<tr>
<td>POST</td>
<td>/actions/&lt;actions&gt;</td>
<td>Create a new alert actions action</td>
</tr>
<tr>
<td>GET</td>
<td>/actions/&lt;actions&gt;/&lt;action&gt;</td>
<td>Get the specified alert actions action properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/actions/&lt;actions&gt;/&lt;action&gt;</td>
<td>Modify the specified alert actions action object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/actions/&lt;actions&gt;/&lt;action&gt;</td>
<td>Destroy the specified action object</td>
</tr>
<tr>
<td>GET</td>
<td>/events</td>
<td>Listen for new alert events</td>
</tr>
</tbody>
</table>
**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>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>Default</td>
<td>Unique identifier for the watch (&quot;immutable&quot;)</td>
</tr>
<tr>
<td>type</td>
<td>ChooseOne</td>
<td>Whether to post alert when the stat exceeds the limit (normal) or falls below the limit (inverted) [&quot;normal&quot;, &quot;inverted&quot;]</td>
</tr>
<tr>
<td>limit</td>
<td>PositiveInteger</td>
<td>Limit value for the statistic</td>
</tr>
<tr>
<td>minpost</td>
<td>Duration</td>
<td>Minimum time condition must hold before posting alert</td>
</tr>
<tr>
<td>days</td>
<td>ChooseOne</td>
<td>Only post alert on particular days [&quot;all&quot;, &quot;weekdays&quot;, &quot;weekends&quot;]</td>
</tr>
<tr>
<td>window_start</td>
<td>TimeOfDay</td>
<td>Only post alerts between window_start and window_end [&quot;none&quot;, &quot;00:00&quot;, &quot;00:30&quot;, &quot;01:00&quot;, &quot;01:30&quot;, &quot;02:00&quot;, &quot;02:30&quot;, &quot;03:00&quot;, &quot;03:30&quot;, &quot;04:00&quot;, &quot;04:30&quot;, &quot;05:00&quot;, &quot;05:30&quot;, &quot;06:00&quot;, &quot;06:30&quot;, &quot;07:00&quot;, &quot;07:30&quot;, &quot;08:00&quot;, &quot;08:30&quot;, &quot;09:00&quot;, &quot;09:30&quot;, &quot;10:00&quot;, &quot;10:30&quot;, &quot;11:00&quot;, &quot;11:30&quot;, &quot;12:00&quot;, &quot;12:30&quot;, &quot;13:00&quot;, &quot;13:30&quot;, &quot;14:00&quot;, &quot;14:30&quot;, &quot;15:00&quot;, &quot;15:30&quot;, &quot;16:00&quot;, &quot;16:30&quot;, &quot;17:00&quot;, &quot;17:30&quot;, &quot;18:00&quot;, &quot;18:30&quot;, &quot;19:00&quot;, &quot;19:30&quot;, &quot;20:00&quot;, &quot;20:30&quot;, &quot;21:00&quot;, &quot;21:30&quot;, &quot;22:00&quot;, &quot;22:30&quot;, &quot;23:00&quot;, &quot;23:30&quot;]</td>
</tr>
<tr>
<td>window_end</td>
<td>TimeOfDay</td>
<td>Only post alerts between window_start and window_end [&quot;none&quot;, &quot;00:00&quot;, &quot;00:30&quot;, &quot;01:00&quot;, &quot;01:30&quot;, &quot;02:00&quot;, &quot;02:30&quot;, &quot;03:00&quot;, &quot;03:30&quot;, &quot;04:00&quot;, &quot;04:30&quot;, &quot;05:00&quot;, &quot;05:30&quot;, &quot;06:00&quot;, &quot;06:30&quot;, &quot;07:00&quot;, &quot;07:30&quot;, &quot;08:00&quot;, &quot;08:30&quot;, &quot;09:00&quot;, &quot;09:30&quot;, &quot;10:00&quot;, &quot;10:30&quot;, &quot;11:00&quot;, &quot;11:30&quot;, &quot;12:00&quot;, &quot;12:30&quot;, &quot;13:00&quot;, &quot;13:30&quot;, &quot;14:00&quot;, &quot;14:30&quot;, &quot;15:00&quot;, &quot;15:30&quot;, &quot;16:00&quot;, &quot;16:30&quot;, &quot;17:00&quot;, &quot;17:30&quot;, &quot;18:00&quot;, &quot;18:30&quot;, &quot;19:00&quot;, &quot;19:30&quot;, &quot;20:00&quot;, &quot;20:30&quot;, &quot;21:00&quot;, &quot;21:30&quot;, &quot;22:00&quot;, &quot;22:30&quot;, &quot;23:00&quot;, &quot;23:30&quot;]</td>
</tr>
</tbody>
</table>
### Alert Thresholds

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00&quot;, &quot;02:30&quot;</td>
<td>&quot;03:00&quot;, &quot;03:30&quot;, &quot;04:00&quot;, &quot;04:30&quot;, &quot;05:00&quot;, &quot;05:30&quot;, &quot;06:00&quot;, &quot;06:30&quot;, &quot;07:00&quot;, &quot;07:30&quot;, &quot;08:00&quot;, &quot;08:30&quot;, &quot;09:00&quot;, &quot;09:30&quot;, &quot;10:00&quot;, &quot;10:30&quot;, &quot;11:00&quot;, &quot;11:30&quot;, &quot;12:00&quot;, &quot;12:30&quot;, &quot;13:00&quot;, &quot;13:30&quot;, &quot;14:00&quot;, &quot;14:30&quot;, &quot;15:00&quot;, &quot;15:30&quot;, &quot;16:00&quot;, &quot;16:30&quot;, &quot;17:00&quot;, &quot;17:30&quot;, &quot;18:00&quot;, &quot;18:30&quot;, &quot;19:00&quot;, &quot;19:30&quot;, &quot;20:00&quot;, &quot;20:30&quot;, &quot;21:00&quot;, &quot;21:30&quot;, &quot;22:00&quot;, &quot;22:30&quot;, &quot;23:00&quot;, &quot;23:30&quot;</td>
<td>(&quot;immutable&quot;)</td>
</tr>
</tbody>
</table>

**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: zfssa.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: zfssa.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: letmein
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


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: zfssa.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: zfssa.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"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active_directory_degraded</td>
<td>Boolean</td>
<td>Filter should match active_directory_degraded events [true or false]</td>
</tr>
<tr>
<td>smb_kerberos_client_authentication_degraded</td>
<td>Boolean</td>
<td>Filter should match smb_kerberos_client_authentication_degraded events [true or false]</td>
</tr>
</tbody>
</table>

### TABLE 9  Alert Actions "all"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_defects</td>
<td>Boolean</td>
<td>Filter should match all_defects events [true, false]</td>
</tr>
<tr>
<td>service_alerts</td>
<td>Boolean</td>
<td>Filter should match service_alerts events true, false</td>
</tr>
<tr>
<td>all_hardware_faults</td>
<td>Boolean</td>
<td>Filter should match all_hardware_faults events true, false</td>
</tr>
</tbody>
</table>

### TABLE 10  Alert Actions "appliance software"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obstacles_to_system_software_update</td>
<td>Boolean</td>
<td>Filter should match obstacles_to_system_software_update events [true, false]</td>
</tr>
<tr>
<td>operating_system_kernel_panic</td>
<td>Boolean</td>
<td>Filter should match operating_system_kernel_panic events [true or false]</td>
</tr>
</tbody>
</table>

### TABLE 11  Alert Actions backup

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup_finished</td>
<td>Boolean</td>
<td>Filter should match backup_finished events [true or false]</td>
</tr>
<tr>
<td>backup_started</td>
<td>Boolean</td>
<td>Filter should match backup_started events [true or false]</td>
</tr>
</tbody>
</table>
### Table 12: Alert Actions "cluster"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster_i/o_link_down</td>
<td>Boolean</td>
<td>Filter should match cluster_i/o_link_down events [true or false]</td>
</tr>
<tr>
<td>cluster_i/o_link_failed</td>
<td>Boolean</td>
<td>Filter should match cluster_i/o_link_failed events [true or false]</td>
</tr>
<tr>
<td>cluster_i/o_link_up</td>
<td>Boolean</td>
<td>Filter should match cluster_i/o_link_up events [true or false]</td>
</tr>
<tr>
<td>unexpected_peer_error_occurred</td>
<td>Boolean</td>
<td>Filter should match unexpected_peer_error_occurred events [true or false]</td>
</tr>
<tr>
<td>communication_to_peer_lost</td>
<td>Boolean</td>
<td>Filter should match communication_to_peer_lost events [true or false]</td>
</tr>
<tr>
<td>clusterPeer_panicked</td>
<td>Boolean</td>
<td>Filter should match clusterPeer_panicked events [true or false]</td>
</tr>
<tr>
<td>failed_to_set_sp_root_password_on_cluster_peer</td>
<td>Boolean</td>
<td>Filter should match failed_to_set_sp_root_password_on_cluster_peer events [true or false]</td>
</tr>
<tr>
<td>cluster_rejoin_failed_on_peer</td>
<td>Boolean</td>
<td>Filter should match cluster_rejoin_failed_on_peer events [true or false]</td>
</tr>
<tr>
<td>cluster_rejoin_mismatch_on_peer</td>
<td>Boolean</td>
<td>Filter should match cluster_rejoin_mismatch_on_peer events [true or false]</td>
</tr>
<tr>
<td>cluster_rejoin_completed_on_peer</td>
<td>Boolean</td>
<td>Filter should match cluster_rejoin_completed_on_peer events [true or false]</td>
</tr>
<tr>
<td>clusterPeer_lost_communication_token</td>
<td>Boolean</td>
<td>Filter should match clusterPeer_lost_communication_token events [true or false]</td>
</tr>
<tr>
<td>cluster_rejoin_failed</td>
<td>Boolean</td>
<td>Filter should match cluster_rejoin_failed events [true or false]</td>
</tr>
<tr>
<td>cluster_rejoin_mismatch</td>
<td>Boolean</td>
<td>Filter should match cluster_rejoin_mismatch events [true or false]</td>
</tr>
<tr>
<td>cluster_rejoin_completed</td>
<td>Boolean</td>
<td>Filter should match cluster_rejoin_completed events [true or false]</td>
</tr>
<tr>
<td>cluster_takeover_complete</td>
<td>Boolean</td>
<td>Filter should match cluster_takeover_complete events [true or false]</td>
</tr>
<tr>
<td>resource_import_failed_during_cluster_takeover</td>
<td>Boolean</td>
<td>Filter should match resource_import_failed_during_cluster_takeover events [true or false]</td>
</tr>
<tr>
<td>local_cluster_communication_token_lost</td>
<td>Boolean</td>
<td>Filter should match local_cluster_communication_token_lost events [true or false]</td>
</tr>
</tbody>
</table>
### TABLE 13  Alert Actions "custom"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>patterns</td>
<td>Default</td>
<td>FMA event patterns</td>
</tr>
</tbody>
</table>

### TABLE 14  Alert Actions "hardware"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fibre_channel_port_down</td>
<td>Boolean</td>
<td>Filter should match fibre_channel_port_down events [true or false]</td>
</tr>
<tr>
<td>multiple_transient_fibre_channel_port_status_changes</td>
<td>Boolean</td>
<td>Filter should match multiple_transient_fibre_channel_port_status_changes events [true or false]</td>
</tr>
<tr>
<td>transient_fibre_channel_port_status_change</td>
<td>Boolean</td>
<td>Filter should match transient_fibre_channel_port_status_change events [true or false]</td>
</tr>
<tr>
<td>fibre_channel_port_up</td>
<td>Boolean</td>
<td>Filter should match fibre_channel_port_up events [true or false]</td>
</tr>
<tr>
<td>network_port_down</td>
<td>Boolean</td>
<td>Filter should match network_port_down events [true or false]</td>
</tr>
<tr>
<td>network_port_up</td>
<td>Boolean</td>
<td>Filter should match network_port_up events [true or false]</td>
</tr>
<tr>
<td>chassis_connected_to_system</td>
<td>Boolean</td>
<td>Filter should match chassis_connected_to_system events [true or false]</td>
</tr>
<tr>
<td>chassis_removed</td>
<td>Boolean</td>
<td>Filter should match chassis_removed events [true or false]</td>
</tr>
<tr>
<td>hardware_component_inserted</td>
<td>Boolean</td>
<td>Filter should match hardware_component_inserted events [true or false]</td>
</tr>
<tr>
<td>hardware_component_removed</td>
<td>Boolean</td>
<td>Filter should match hardware_component_removed events [true or false]</td>
</tr>
<tr>
<td>disk_inserted</td>
<td>Boolean</td>
<td>Filter should match disk_inserted events [true or false]</td>
</tr>
<tr>
<td>disk_removed</td>
<td>Boolean</td>
<td>Filter should match disk_removed events [true or false]</td>
</tr>
<tr>
<td>i/o_path_added</td>
<td>Boolean</td>
<td>Filter should match i/o_path_added events [true or false]</td>
</tr>
<tr>
<td>i/o_path_removed</td>
<td>Boolean</td>
<td>Filter should match i/o_path_removed events [true or false]</td>
</tr>
<tr>
<td>service_processor_offline_or_unavailable</td>
<td>Boolean</td>
<td>Filter should match service_processor_offline_or_unavailable events [true or false]</td>
</tr>
<tr>
<td>service_processor_online_after_outage</td>
<td>Boolean</td>
<td>Filter should match service_processor_online_after_outage events [true or false]</td>
</tr>
</tbody>
</table>
### TABLE 15  Alert Actions "hardware_faults"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_hardware_faults</td>
<td>Boolean</td>
<td>Filter should match all_hardware_faults events [true or false]</td>
</tr>
</tbody>
</table>

### TABLE 16  Alert Actions "ndmp"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_ndmp_restore</td>
<td>Boolean</td>
<td>Filter should match invalid_ndmp_restore events [true or false]</td>
</tr>
<tr>
<td>backup_finished</td>
<td>Boolean</td>
<td>Filter should match backup_finished events [true or false]</td>
</tr>
<tr>
<td>backup_started</td>
<td>Boolean</td>
<td>Filter should match backup_started events [true or false]</td>
</tr>
<tr>
<td>restore_finished</td>
<td>Boolean</td>
<td>Filter should match restore_finished events [true or false]</td>
</tr>
<tr>
<td>restore_started</td>
<td>Boolean</td>
<td>Filter should match restore_started events [true or false]</td>
</tr>
</tbody>
</table>

### TABLE 17  Alert Actions "network"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datalink_failed</td>
<td>Boolean</td>
<td>Filter should match datalink_failed events [true or false]</td>
</tr>
<tr>
<td>datalink_ok</td>
<td>Boolean</td>
<td>Filter should match datalink_ok events [true or false]</td>
</tr>
<tr>
<td>network_port_down</td>
<td>Boolean</td>
<td>Filter should match network_port_down events [true or false]</td>
</tr>
<tr>
<td>network_port_up</td>
<td>Boolean</td>
<td>Filter should match network_port_up events [true or false]</td>
</tr>
<tr>
<td>ip_address_conflict</td>
<td>Boolean</td>
<td>Filter should match ip_address_conflict events [true or false]</td>
</tr>
<tr>
<td>ip_address_conflict_resolved</td>
<td>Boolean</td>
<td>Filter should match ip_address_conflict_resolved events [true or false]</td>
</tr>
<tr>
<td>ip_interface_degraded</td>
<td>Boolean</td>
<td>Filter should match ip_interface_degraded events [true or false]</td>
</tr>
<tr>
<td>ip_interface_failed</td>
<td>Boolean</td>
<td>Filter should match ip_interface_failed events [true or false]</td>
</tr>
<tr>
<td>ip_interface_ok</td>
<td>Boolean</td>
<td>Filter should match ip_interface_ok events [true or false]</td>
</tr>
</tbody>
</table>
### TABLE 18  Alert Actions "replication"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>receive_failed_(unsupported_version)</td>
<td>Boolean</td>
<td>Filter should match receive_failed_(unsupported_version) events [true or false]</td>
</tr>
<tr>
<td>receive_failed_(cancelled)</td>
<td>Boolean</td>
<td>Filter should match receive_failed_(cancelled) events [true or false]</td>
</tr>
<tr>
<td>receive_failed_(all_others)</td>
<td>Boolean</td>
<td>Filter should match receive_failed_(all_others) events [true or false]</td>
</tr>
<tr>
<td>receive_failed_(out_of_space)</td>
<td>Boolean</td>
<td>Filter should match receive_failed_(out_of_space) events [true or false]</td>
</tr>
<tr>
<td>receive_failed_(package_not_upgraded)</td>
<td>Boolean</td>
<td>Filter should match receive_failed_(package_not_upgraded) events [true or false]</td>
</tr>
<tr>
<td>receive_finished</td>
<td>Boolean</td>
<td>Filter should match receive_finished events [true or false]</td>
</tr>
<tr>
<td>receive_started</td>
<td>Boolean</td>
<td>Filter should match receive_started events [true or false]</td>
</tr>
<tr>
<td>send_failed_(unsupported_version)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(unsupported_version) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(cancelled)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(cancelled) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(all_others)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(all_others) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(connectivity)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(connectivity) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(out_of_space)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(out_of_space) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(remote_verification)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(remote_verification) events [true or false]</td>
</tr>
<tr>
<td>send_finished</td>
<td>Boolean</td>
<td>Filter should match send_finished events [true or false]</td>
</tr>
<tr>
<td>send_skipped_(already_running)</td>
<td>Boolean</td>
<td>Filter should match send_skipped_(already_running) events [true or false]</td>
</tr>
<tr>
<td>send_started</td>
<td>Boolean</td>
<td>Filter should match send_started events [true or false]</td>
</tr>
</tbody>
</table>

### TABLE 19  Alert Actions "replication_source"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>send_failed_(unsupported_version)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(unsupported_version) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(cancelled)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(cancelled) events [true or false]</td>
</tr>
</tbody>
</table>
### Alert Actions

**TABLE 20** Alert Actions "replication_target"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>send_failed_(all_others)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(all_others) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(connectivity)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(connectivity) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(out_of_space)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(out_of_space) events [true or false]</td>
</tr>
<tr>
<td>send_failed_(remote_verification)</td>
<td>Boolean</td>
<td>Filter should match send_failed_(remote_verification) events [true or false]</td>
</tr>
<tr>
<td>send_finished</td>
<td>Boolean</td>
<td>Filter should match sendFinished events [true or false]</td>
</tr>
<tr>
<td>send_skipped_(already_running)</td>
<td>Boolean</td>
<td>Filter should match send_skipped_(already_running) events [true or false]</td>
</tr>
<tr>
<td>send_started</td>
<td>Boolean</td>
<td>Filter should match send_started events [true or false]</td>
</tr>
</tbody>
</table>

**TABLE 21** Alert Actions "restore"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>restore_finished</td>
<td>Boolean</td>
<td>Filter should match restore_finished events [true or false]</td>
</tr>
<tr>
<td>restore_started</td>
<td>Boolean</td>
<td>Filter should match restore_started events [true or false]</td>
</tr>
</tbody>
</table>
**TABLE 22** Alert Actions "scrk"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>support_bundle_build_failed</td>
<td>Boolean</td>
<td>Filter should match support_bundle_build_failed events [true or false]</td>
</tr>
<tr>
<td>support_bundle_sent</td>
<td>Boolean</td>
<td>Filter should match support_bundle_sent events [true or false]</td>
</tr>
<tr>
<td>support_bundle_upload_failed</td>
<td>Boolean</td>
<td>Filter should match support_bundle_upload_failed events [true or false]</td>
</tr>
<tr>
<td>an_update_is_available_on_my_oracle_support.</td>
<td>Boolean</td>
<td>Filter should match an_update_is_available_on_my_oracle_support. events [true or false]</td>
</tr>
<tr>
<td>no_updates_available.</td>
<td>Boolean</td>
<td>Filter should match no_updates_available. events [true or false]</td>
</tr>
<tr>
<td>the_appliance_failed_to_verify_an_update_is_available.</td>
<td>Boolean</td>
<td>Filter should match the_appliance_failed_to_verify_an_update_is_available. events [true or false]</td>
</tr>
</tbody>
</table>

**TABLE 23** Alert Actions "shadow"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shadow_migration_complete</td>
<td>Boolean</td>
<td>Filter should match shadow_migration_complete events [true or false]</td>
</tr>
</tbody>
</table>

**TABLE 24** Alert Actions "smf"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_failures</td>
<td>Boolean</td>
<td>Filter should match service_failures events [true or false]</td>
</tr>
</tbody>
</table>

**TABLE 25** Alert Actions "thresholds"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thresholdid</td>
<td>Default</td>
<td>UUID of watch whose alerts should match</td>
</tr>
</tbody>
</table>

**TABLE 26** Alert Actions "zfs_pool"

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resilver_finished</td>
<td>Boolean</td>
<td>Filter should match resilver_finished events [true or false]</td>
</tr>
<tr>
<td>resilver_started</td>
<td>Boolean</td>
<td>Filter should match resilver_started events [true or false]</td>
</tr>
<tr>
<td>scrub_finished</td>
<td>Boolean</td>
<td>Filter should match scrub_finished events [true or false]</td>
</tr>
<tr>
<td>scrub_started</td>
<td>Boolean</td>
<td>Filter should match scrub_started events [true or false]</td>
</tr>
</tbody>
</table>
### Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hot_spare_activated</td>
<td>Boolean</td>
<td>Filter should match hot_spare_activated events [true or false]</td>
</tr>
</tbody>
</table>

## 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:**

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

**Example Response:**

```plaintext
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": "Joe.Admin@acme.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
        }
    ]
}
Alert Actions

[RESTful API Alert Service]

```
{
  "action": "actions-002",
  "category": "thresholds",
  "href": "/api/alert/v1/actions/actions-002",
  "action-000": {
    "address": "Joe.Admin@acme.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": "Joe.Admin@acme.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.

Category values typically include:

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

Example Request:

```
POST /api/alert/v1/actions HTTP/1.1
Host: zfssa.example.com:215
X-Auth-Session: userqghq84vbdv
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: zfssa.example.com:215
X-Auth-Session: userqghq84vbdv
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: zfssa.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: zfssa.example.com:215
X-Auth-Session: uerqghq84vbdv
Content-Type: application/json
Content-Length: 68
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": "Joe.Admin@acme.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: zfssa.example.com:215
X-Auth-Session: uerqghq84vbdv
Content-Type: application/json
Content-Length: 28

{"address": "Joseph.Admin@acme.com"}

Example Response:

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

{
   "action": {
      "href": "/api/alert/v1/actions/actions-001/action-000",
      "handler": "email",
      "address": "Joseph.Admin@acme.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: zfssa.example.com:215
X-Auth-Session: uerghq84vbdv

HTTP/1.1 204 No Content
Analytics Services

Analytics is a facility for graphing a variety of statistics in real time and recording data for later retrieval. It provides for long-term monitoring and short-term analysis. Analytics makes use of DTrace to dynamically create custom statistics that allow different layers of the operating system stack be analyzed in detail.

Available Analytics Services

The following Analytics services are available at: http://zfssa.example.com/api/analytics/v1.0/

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /analytics/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/worksheets</td>
<td>List analytics service information</td>
</tr>
<tr>
<td>POST</td>
<td>/worksheets</td>
<td>Create a new analytics dataset</td>
</tr>
<tr>
<td>GET</td>
<td>/worksheets/&lt;worksheet&gt;</td>
<td>Get the specified analytics dataset properties</td>
</tr>
<tr>
<td>GET</td>
<td>/worksheets</td>
<td>List all analytics dataset objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/worksheets/&lt;worksheet&gt;</td>
<td>Modify the specified analytics dataset object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/worksheets/&lt;worksheet&gt;</td>
<td>Destroy the specified worksheet object</td>
</tr>
<tr>
<td>PUT</td>
<td>/worksheets/&lt;worksheet&gt;/suspend</td>
<td>Suspend all worksheet datasets</td>
</tr>
<tr>
<td>PUT</td>
<td>/worksheets/&lt;worksheet&gt;/resume</td>
<td>Resume all worksheet datasets</td>
</tr>
<tr>
<td>POST</td>
<td>/worksheets/&lt;worksheet&gt;/datasets</td>
<td>Create a new worksheet dataset</td>
</tr>
<tr>
<td>GET</td>
<td>/worksheets/&lt;worksheet&gt;/datasets/&lt;dataset&gt;</td>
<td>Get the specified worksheet dataset properties</td>
</tr>
<tr>
<td>GET</td>
<td>/worksheets/&lt;worksheet&gt;/datasets</td>
<td>List all worksheet dataset objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/worksheets/&lt;worksheet&gt;/datasets/&lt;dataset&gt;</td>
<td>Modify the specified worksheet dataset object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/worksheets/&lt;worksheet&gt;/datasets/&lt;dataset&gt;</td>
<td>Destroy the specified dataset object</td>
</tr>
<tr>
<td>POST</td>
<td>/datasets</td>
<td>Create a new analytics dataset</td>
</tr>
<tr>
<td>GET</td>
<td>/datasets/&lt;dataset&gt;</td>
<td>Get the specified analytics dataset properties</td>
</tr>
<tr>
<td>GET</td>
<td>/datasets</td>
<td>List all analytics dataset objects</td>
</tr>
</tbody>
</table>
### Analytics Settings

The following properties let you collect all analytic data or set the number of hours of data to retain.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>retain_second_data</td>
<td>Retention interval in hours for per-second data</td>
</tr>
<tr>
<td>retain_minute_data</td>
<td>Retention interval in hours for per-minute data</td>
</tr>
<tr>
<td>retain_hour_data</td>
<td>Retention interval in hours for per-hour data</td>
</tr>
</tbody>
</table>

### Get Settings

Gets the current values of analytics setting properties.

Example Request:

```
GET /api/analytics/v1/settings HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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,
```
Modify Settings

The modify settings command is used to modify analytics settings such as data retention values.

Example Request:

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

{"retain_hour_data":600, "retain_minute_data":400, "retain_second_data":200}

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
    }
}

Analytics Worksheets

A worksheet is the BUI screen on which statistics are graphed. Multiple statistics can be plotted at the same time, and worksheets may 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctime</td>
<td>Time and date when this worksheet was created</td>
</tr>
<tr>
<td>mtime</td>
<td>Time and date when this worksheet was last modified</td>
</tr>
<tr>
<td>name</td>
<td>Name of this worksheet</td>
</tr>
</tbody>
</table>
List Worksheets

Lists all currently configured analytics worksheets.

Example Request:

GET /api/analytics/v1/worksheets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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/ab59bcb...",
    "uuid": "ab59bcb-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/ab59bcb-080a-cf1a-98c9-9f485bc3a43d HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.com:215
Accept: application/json

Example Response:

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

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. Dataset configuration uses the following properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the underlying statistic for this dataset</td>
</tr>
<tr>
<td>drilldowns</td>
<td>Drilldowns currently highlighted, if any</td>
</tr>
<tr>
<td>seconds</td>
<td>Number of seconds being displayed for this dataset</td>
</tr>
</tbody>
</table>

Example Request:

GET /api/analytics/v1/worksheets/name=myworksheet/datasets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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
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

### TABLE 27 Dataset Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of underlying statistic</td>
</tr>
<tr>
<td>grouping</td>
<td>Group to which this statistic belongs</td>
</tr>
<tr>
<td>explanation</td>
<td>Explanation of underlying statistic</td>
</tr>
<tr>
<td>incore</td>
<td>Bytes of dataset data in-core</td>
</tr>
<tr>
<td>size</td>
<td>Bytes of dataset data on-disk</td>
</tr>
<tr>
<td>suspended</td>
<td>Boolean indicating whether dataset is currently suspended</td>
</tr>
<tr>
<td>activity</td>
<td>Pending dataset activity flag</td>
</tr>
</tbody>
</table>

All properties except for suspended are immutable.

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.bytes[op]
- io.disks[utilization=95][disk]
- io.ops
- io.ops[disk]
- io.ops[op]
- iscsi.bytes
- iscsi.ops
- ndmp.diskkb
- nfs2.ops
List Datasets

Lists all configured analytic datasets.

Example Request:

GET /api/analytics/v1/datasets HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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"
    }]}
Get Dataset

Gets properties from the specified dataset.

Example Request:

```
GET /api/analytics/v1/datasets/nfs4.ops HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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: letmein
```
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 data within a dataset can be pruned at a granularity value of second, minute, or hour.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>granularity</td>
<td>Prune granularity (second, minute, hour)</td>
</tr>
<tr>
<td>endtime</td>
<td>Prune data collected prior to given endtime</td>
</tr>
</tbody>
</table>

The endtime is an ISO 8601 time/date string (e.g. 20130531T01:13:58)

Example Request:

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

Example Response:
Get Dataset Data

Gets data from an analytic dataset.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>The time to start collecting sample data</td>
</tr>
<tr>
<td>seconds</td>
<td>Number of seconds to collect sample data (Default = 1)</td>
</tr>
</tbody>
</table>

The startTime can be one of the following:
- An ISO 8601 time/date string (e.g. 20130531T01:13:58)
- Sample index number
- The string literal now

If start is not supplied, it sets to the current time minus the number of seconds of sample data desired. The start time cannot be in the future. If the number of seconds to collect data goes past the current time, the server waits for each sample before returning the data.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>The time of the first sample returned</td>
</tr>
<tr>
<td>sample</td>
<td>The sample index of the first sample returned</td>
</tr>
<tr>
<td>data</td>
<td>Array of sample data</td>
</tr>
</tbody>
</table>

Example Request (to collect 3 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: example.zfssa.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
---
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>
<thead>
<tr>
<th>Request</th>
<th>Path /hardware/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/cluster</td>
<td>Get cluster properties and cluster resource list</td>
</tr>
<tr>
<td>GET</td>
<td>/cluster/resources /<a href="">resource:path</a></td>
<td>Get properties for the specified cluster resource</td>
</tr>
<tr>
<td>PUT</td>
<td>/cluster/resources /<a href="">resource:path</a></td>
<td>Modify the specified cluster resource</td>
</tr>
<tr>
<td>PUT</td>
<td>/cluster/failback</td>
<td>Fail back all resources assigned to the cluster peer</td>
</tr>
<tr>
<td>PUT</td>
<td>/cluster/takeover</td>
<td>Take over all resources assigned to the cluster peer</td>
</tr>
<tr>
<td>PUT</td>
<td>/cluster/unconfigure</td>
<td>Unconfigure a clustered appliance to standalone mode</td>
</tr>
<tr>
<td>GET</td>
<td>/cluster/links</td>
<td>Get cluster card link status</td>
</tr>
<tr>
<td>PUT</td>
<td>/cluster/setup</td>
<td>Run through initial cluster setup</td>
</tr>
</tbody>
</table>

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: tanana: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": ["10.80.231.58"],
            "href": "/hardware/v1/cluster/resources/resources/net/ixgbe0",
            "owner": "tanana",
            "type": "singleton",
            "user_label": "Untitled Interface"
         },
         "zfs/gold": {
            "details": ["821G"],
            "href": "/hardware/v1/cluster/resources/resources/zfs/gold",
            "owner": "tanana",
            "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/gold 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 failover, takeback, and unconfigure. All commands take a PUT request to the cluster resource with the name of the command appended. On success, both commands return HTTP Status 202 (Accepted).

Example Request:

```
PUT /api/hardware/v1/cluster/failback HTTP/1.1
Authorization: Basic abcd123MWE=
Host: zfssa.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: zfssa.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
        clustron2_embedded:0/clustron_uart:1 = AKCIOS_TIMEDOUT
        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 AKCLOS_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: zfssa.example.com:215
Accept: application/json

{"nodename": "zfssa-storage-2", "password": "letmein"}
```

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>
<thead>
<tr>
<th>Request</th>
<th>Path /hardware/v1.0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/chassis</td>
<td>List hardware chassis</td>
</tr>
<tr>
<td>GET</td>
<td>/chassis/&lt;chassis&gt;</td>
<td>Get the specified hardware chassis properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/chassis/&lt;chassis&gt;</td>
<td>Modify the specified hardware chassis properties</td>
</tr>
<tr>
<td>GET</td>
<td>/chassis/&lt;chassis&gt;/&lt;fru_type&gt;</td>
<td>List hardware chassis components</td>
</tr>
<tr>
<td>GET</td>
<td>/chassis/&lt;chassis&gt;/&lt;fru_type&gt;</td>
<td>Get the specified chassis component properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/chassis/&lt;chassis&gt;/&lt;fru_type&gt;</td>
<td>Modify hardware chassis component properties</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>name</td>
<td>Chassis name</td>
</tr>
</tbody>
</table>
## Chassis

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>model</td>
<td>Chassis model number</td>
</tr>
<tr>
<td>string</td>
<td>manufacturer</td>
<td>Chassis manufacturer</td>
</tr>
<tr>
<td>string</td>
<td>serial</td>
<td>Chassis serial number</td>
</tr>
<tr>
<td>string</td>
<td>revision</td>
<td>Chassis revision level</td>
</tr>
<tr>
<td>string</td>
<td>part</td>
<td>Chassis replacement part number</td>
</tr>
<tr>
<td>boolean</td>
<td>faulted</td>
<td>Fault indicator</td>
</tr>
<tr>
<td>string</td>
<td>fru</td>
<td>FMRI representation of the chassis</td>
</tr>
<tr>
<td>string</td>
<td>uuid</td>
<td>Chassis uuid identifier</td>
</tr>
</tbody>
</table>

**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-Apni: 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",
        "name": " fills the blank",
```
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"
   }
}
"revision": "A310",
"rpm": 7200,
"serial": "001210R37LVD YHJ37LVD",
"size": 300592982016,
"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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>device</td>
<td>The FRU device ID</td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>boolean</td>
<td>faulted</td>
<td>Flag indicating if FRU is faulted</td>
</tr>
<tr>
<td>string</td>
<td>fru</td>
<td>FMRI representation of a FRU</td>
</tr>
<tr>
<td>string</td>
<td>interface</td>
<td>FRU interface type</td>
</tr>
<tr>
<td>string</td>
<td>label</td>
<td>FRU location label</td>
</tr>
<tr>
<td>boolean</td>
<td>locate</td>
<td>Locate indicator on flag</td>
</tr>
<tr>
<td>string</td>
<td>manufacturer</td>
<td>FRU manufacturer</td>
</tr>
<tr>
<td>string</td>
<td>model</td>
<td>FRU model</td>
</tr>
<tr>
<td>string</td>
<td>part</td>
<td>FRU part number</td>
</tr>
<tr>
<td>boolean</td>
<td>present</td>
<td>FRU presence indicator</td>
</tr>
<tr>
<td>number</td>
<td>rpm</td>
<td>Platter RPM (disk only)</td>
</tr>
<tr>
<td>string</td>
<td>serial</td>
<td>FRU serial number</td>
</tr>
<tr>
<td>number</td>
<td>size</td>
<td>FRU size (capacity)</td>
</tr>
<tr>
<td>string</td>
<td>type</td>
<td>Component type</td>
</tr>
<tr>
<td>string</td>
<td>use</td>
<td>Component usage enumeration</td>
</tr>
</tbody>
</table>

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

```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": "H7230AS605UN3.0T",
        "pathcount": 4,
        "present": true,
        "revision": "A310",
        "rpm": 7200,
        "serial": "001210R322ED YH332ED",
        "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: zfssa.example.com:215
X-Auth-User: root
X-Auth-Key: letmein
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.

Log Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/log/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/logs</td>
<td>List the log service commands</td>
</tr>
<tr>
<td>GET</td>
<td>/logs/&lt;log&gt;</td>
<td>List log entries for the selected log</td>
</tr>
<tr>
<td>GET</td>
<td>/collect/&lt;log&gt;</td>
<td>Download a collection of specified log entries</td>
</tr>
<tr>
<td>GET</td>
<td>/collect</td>
<td>Download a collection of all log entries</td>
</tr>
</tbody>
</table>

List Logs

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.

Example Request:

```
GET /api/log/v1/logs HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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-Appliance: 1.0
```
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, they should be downloaded using the collect function described below.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start=&lt;index&gt;</td>
<td>Start returning logs from the given index/time</td>
</tr>
</tbody>
</table>
Property | Description
---|---
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, 20130724T03:55.

**Note** - 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: tanana:215
Accept: application/json

Example Result:

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

{  
  "logs": [  
    {  
      "address": "10.159.84.34",
      "annotation": "",
      "summary": "User logged in",
      "timestamp": "20131022T22:54:19",
      "user": "root"
    },  
    {  
      "address": "10.159.84.34",
      "annotation": "",
      "summary": "Destroyed share \"gold:tst.volumes.py.34111.project/tst.volumes.py.34111.lun.7\"",
      "timestamp": "20131022T22:52:43",
      "user": "root"
    },  
    {  
      "summary": "Joined workgroup \"RESTTESTWG\"",
      "timestamp": "20131022T22:54:23",
      "user": "<system>"
    },  
    {  
      "address": "10.159.84.34",
      "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. 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.
Network Datalinks

TABLE 28  Network Datalink Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/datalinks</td>
<td>Create a new network datalink</td>
</tr>
<tr>
<td>GET</td>
<td>/datalinks/&lt;datalink&gt;</td>
<td>Get the specified network datalink properties</td>
</tr>
<tr>
<td>GET</td>
<td>/datalinks</td>
<td>List all network datalink objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/datalinks/&lt;datalink&gt;</td>
<td>Modify the specified network datalink object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/datalinks/&lt;datalink&gt;</td>
<td>Destroy the specified datalink object</td>
</tr>
</tbody>
</table>

TABLE 29  Physical Device Datalink Properties

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

TABLE 30  VNIC Device Datalink Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>String</td>
<td>“vnic” (“immutable”)</td>
</tr>
<tr>
<td>label</td>
<td>NetworkLabel</td>
<td>Label</td>
</tr>
<tr>
<td>links</td>
<td>ChooseOne</td>
<td>Links [“ixgbe0”]</td>
</tr>
<tr>
<td>mtu</td>
<td>PositiveInteger</td>
<td>Max transmission unit (MTU)</td>
</tr>
<tr>
<td>id</td>
<td>VLAN</td>
<td>VLAN ID</td>
</tr>
</tbody>
</table>

TABLE 31  VLAN Device Datalink Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>String</td>
<td>“vlan” (“immutable”)</td>
</tr>
<tr>
<td>label</td>
<td>NetworkLabel</td>
<td>Label</td>
</tr>
<tr>
<td>links</td>
<td>ChooseOne</td>
<td>Links [“ixgbe0”]</td>
</tr>
<tr>
<td>mtu</td>
<td>PositiveInteger</td>
<td>Max transmission unit (MTU)</td>
</tr>
<tr>
<td>id</td>
<td>VLAN</td>
<td>VLAN ID</td>
</tr>
</tbody>
</table>
TABLE 32  Aggregation Based Device Datalink Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>String</td>
<td>&quot;aggregation&quot; (&quot;immutable&quot;)</td>
</tr>
<tr>
<td>label</td>
<td>NetworkLabel</td>
<td>Label</td>
</tr>
<tr>
<td>links</td>
<td>ChooseN</td>
<td>Links [&quot;igb1&quot;, &quot;igb0&quot;, &quot;ixgbe2&quot;, &quot;ixgbe3&quot;, &quot;igb4&quot;, &quot;igb3&quot;, &quot;ixgbe1&quot;, &quot;igb2&quot;, &quot;igb5&quot;]</td>
</tr>
<tr>
<td>jumbo</td>
<td>Boolean</td>
<td>Use Jumbo Frames [&quot;true&quot;, &quot;false&quot;] (&quot;deprecated&quot;)</td>
</tr>
<tr>
<td>mtu</td>
<td>PositiveInteger</td>
<td>Max transmission unit (MTU)</td>
</tr>
<tr>
<td>policy</td>
<td>ChooseOne</td>
<td>Policy [&quot;L2&quot;, &quot;L3&quot;, &quot;L4&quot;, &quot;L2+L3&quot;, &quot;L2+L4&quot;, &quot;L3+L4&quot;]</td>
</tr>
<tr>
<td>mode</td>
<td>ChooseOne</td>
<td>Mode [&quot;active&quot;, &quot;passive&quot;, &quot;off&quot;]</td>
</tr>
<tr>
<td>timer</td>
<td>ChooseOne</td>
<td>Timer [&quot;short&quot;, &quot;long&quot;]</td>
</tr>
<tr>
<td>key</td>
<td>Integer</td>
<td>Aggregation Key (&quot;immutable&quot;)</td>
</tr>
</tbody>
</table>

TABLE 33  IP-Partition-Based Device Datalink Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>String</td>
<td>&quot;partition&quot; (&quot;immutable&quot;)</td>
</tr>
<tr>
<td>label</td>
<td>NetworkLabel</td>
<td>Label</td>
</tr>
<tr>
<td>links</td>
<td>ChooseOne</td>
<td>Links</td>
</tr>
<tr>
<td>pkey</td>
<td>Pkey</td>
<td>Partition Key</td>
</tr>
<tr>
<td>linkmode</td>
<td>ChooseOne</td>
<td>Link Mode [&quot;cm&quot;, &quot;ud&quot;]</td>
</tr>
</tbody>
</table>

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:

```json
{
    "datalinks": [{
        "href": "/api/network/v1/datalinks/ixgbe0",
        ...
    }, {
        "href": "/api/network/v1/datalinks/ixgbe1",
```
Get Network Datalink

The GET methods 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:

```json
{
  "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: zfssa.example.com:215
X-Auth-User: root
X-Auth-Key: letmein
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

Lists the physical network devices on the system. There are no modifiable properties on physical network devices.

<table>
<thead>
<tr>
<th>TABLE 34</th>
<th>Network Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Path /network/v1</td>
</tr>
<tr>
<td>GET</td>
<td>/devices/&lt;device&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/devices</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 35</th>
<th>Network Device Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>active</td>
<td>Boolean flag indicating if device is active</td>
</tr>
<tr>
<td>duplex</td>
<td>Duplex of device</td>
</tr>
<tr>
<td>factory_mac</td>
<td>Factory MAC address</td>
</tr>
<tr>
<td>media</td>
<td>Device media</td>
</tr>
<tr>
<td>speed</td>
<td>Device speed, in megabits/second</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>up</td>
<td>Boolean flag indicating if device is operational</td>
</tr>
</tbody>
</table>

### List Network Devices

Lists all network devices.

Example Request:

```
GET /api/network/v1/devices HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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

Gets the properties from a single network device.

Example Request:

```
GET /api/network/v1/devices/ixgbe0 HTTP/1.1
Authorization: Basic ab6rt4psMWE=
Host: example.zfssa.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 36** Network Interface Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/network/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/interfaces</td>
<td>Create a new network interface</td>
</tr>
<tr>
<td>GET</td>
<td>/interfaces/&lt;interface&gt;</td>
<td>Get the specified network interface properties</td>
</tr>
<tr>
<td>GET</td>
<td>/interfaces</td>
<td>List all network interface objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/interfaces/&lt;interface&gt;</td>
<td>Modify the specified network interface object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/interfaces/&lt;interface&gt;</td>
<td>Destroy the specified interface object</td>
</tr>
</tbody>
</table>

**TABLE 37** Network Interface Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>Flag indicating if administration is allowed on this interface</td>
</tr>
<tr>
<td>class</td>
<td>Class type (&quot;ip&quot;, &quot;ipmp&quot;) (immutable after create)</td>
</tr>
<tr>
<td>curaddrs</td>
<td>Current IP Addresses (immutable)</td>
</tr>
<tr>
<td>enable</td>
<td>Flag indicating if this interface is enabled</td>
</tr>
<tr>
<td>label</td>
<td>User label for interface</td>
</tr>
<tr>
<td>links</td>
<td>Chose a network link for this interface</td>
</tr>
<tr>
<td>state</td>
<td>State of Interface (immutable)</td>
</tr>
<tr>
<td>v4addrs</td>
<td>IPv4 Addresses</td>
</tr>
<tr>
<td>v6dhcp</td>
<td>IPv4 DHCP flag</td>
</tr>
<tr>
<td>v6addrs</td>
<td>IPv6 Addresses</td>
</tr>
</tbody>
</table>
Network Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v6dhcp</td>
<td>IPv6 DHCP flag</td>
</tr>
</tbody>
</table>

**List Network Interfaces**

Lists all of the configured network interfaces.

Example Request:

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

Example Response:

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

```json
{
    "interfaces": [
        {
            "href": "/api/network/v1/interfaces/ixgbe0",
            "v4addrs": ["10.80.231.58/24"
        },
        {
            "href": "/api/network/v1/interfaces/ixgbe1",
            "v4addrs": ["10.80.231.59/24"
        },
        {
            "href": "/api/network/v1/interfaces/ixgbe2",
            "v4addrs": ["10.80.231.60/24"
        },
        {
            "href": "/api/network/v1/interfaces/ixgbe3",
            "v4addrs": ["10.80.231.61/24"
        }
    ]
}
```

**Get Network Interface**

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: zfssa.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": ["10.80.231.58/24"],
      "enable": true,
      "href": "/api/network/v1/interfaces/ixgbe0",
      "interface": "ixgbe0",
      "label": "Untitled Interface",
      "links": ["ixgbe0"],
      "state": "up",
      "v4addrs": ["10.80.231.58/24"],
      "v4dhcp": false,
      "v6addrs": [],
      "v6dhcp": false
   }
}

Create Network Interface

Creates a new network interface.

Example Request:

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

{
   "class": "ip",
   "links": ["ixgbe3"],
   "v4addrs": "192.168.1.9/24"
}

Example Response:

HTTP/1.1 201 Created
X-Zfssa-Appliance_Api: 1.0
Location: /api/network/v1/interfaces/ixgbe3
Modify Network Interface

Modifies an existing network interface.

Example Request:

```
PUT /api/network/v1/interfaces/ixgbe3 HTTP/1.1
{
    "v4addrs": ["192.168.1.99/24"],
    "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": ["192.168.1.9/24"],
    "enable": true,
    "href": "/api/network/v1/interfaces/ixgbe3",
    "interface": "ixgbe3",
    "label": "Demo Rest",
    "links": ["ixgbe3"],
    "state": "failed",
    "v4addrs": ["192.168.1.99/24"],
    "v4dhcp": false,
    "v6addrs": [],
    "v6dhcp": false
}
```

Delete Network Interface

Deletes an existing network interface.

Example Request:

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

Example Result:

```
HTTP/1.1 204 No Content
```
Network Routes

Manages network routes.

**TABLE 38**  Manage Network Routes

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/network/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/routes</td>
<td>Create a new network route</td>
</tr>
<tr>
<td>GET</td>
<td>/routes/&lt;route&gt;</td>
<td>Get the specified network route properties</td>
</tr>
<tr>
<td>GET</td>
<td>/routes</td>
<td>List all network route objects</td>
</tr>
<tr>
<td>DELETE</td>
<td>/routes/&lt;route&gt;</td>
<td>Destroy the specified route object</td>
</tr>
<tr>
<td>GET</td>
<td>/routing</td>
<td>Get net routing properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/routing</td>
<td>Modify net routing properties</td>
</tr>
</tbody>
</table>

**TABLE 39**  Network Route Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Type of route such as “system” or “static” (immutable)</td>
</tr>
<tr>
<td>family</td>
<td>Address family (either IPv4 or IPv6)</td>
</tr>
<tr>
<td>destination</td>
<td>Route destination address</td>
</tr>
<tr>
<td>gateway</td>
<td>Gateway address</td>
</tr>
<tr>
<td>interface</td>
<td>Network datalink interface</td>
</tr>
</tbody>
</table>

The href path to each route uses the 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: zfssa.example.com:215
Accept: application/json
```

Example Result:

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

{
    "routes": [
        {
            "destination": "0.0.0.0",
            "family": "IPv4",
            "gateway": "10.80.231.1",
            "href": "/api/network/v1/routing/route-000",
            "interface": "ixgbe0",
            "mask": 0,
            "route": "route-000",
            "type": "static"
        },
        {
            "destination": "10.80.231.0",
            "family": "IPv4",
            "gateway": "10.80.231.58",
            "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=10.80.231.0 HTTP/1.1
Authorization: Basic abcd1234MWE=
Host: zfssa.example.com:215
Accept: application/json

Example Result:

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

{
    "route": {
        "destination": "10.80.231.0",
        "family": "IPv4",
        "gateway": "10.80.231.58",
        "href": "/api/network/v1/routes/route-001",
        "interface": "ixgbe0",
        "mask": 24,
        "route": "route-001",
        "type": "system"
    }
}
Network Routes

{  

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: zfssa.example.com:215  
Content-Type: application/json  
Content-Length: 164

{  
"family": "IPv4",  
"destination": "0.0.0.0",  
"mask": "0",  
"gateway": "10.11.12.1",  
"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: zfssa.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>
<thead>
<tr>
<th>TABLE 40</th>
<th>Problem Service Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Path /problem/v1</td>
</tr>
<tr>
<td>GET</td>
<td>List the problem service commands</td>
</tr>
<tr>
<td>GET</td>
<td>/problems</td>
</tr>
<tr>
<td>GET</td>
<td>/problems/&lt;problem&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/problems/&lt;problem&gt;/markrepaired</td>
</tr>
<tr>
<td>PUT</td>
<td>Mark the specified problem uuid as repaired</td>
</tr>
</tbody>
</table>

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

```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-dcb4-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.

**URI Input Parameters**

uuid – The UUID of a single problem.

**Example Request:**
GET /api/problem/v1.0/problems/0d30be41-b50d-4d03-dcb4-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-dcb4-edb69ee080f8",
    ...
  }
}
A problem can be marked as repaired with the repair problem command.

**Input Parameters** `uuid` – The UUID of the problem to be marked repaired.

Example Request:

```plaintext
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):

```plaintext
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>
<thead>
<tr>
<th>TABLE 41</th>
<th>Role Service Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Path /role/v1</td>
</tr>
<tr>
<td>GET</td>
<td>/roles/&lt;role&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/roles</td>
</tr>
<tr>
<td>PUT</td>
<td>/roles/&lt;role&gt;</td>
</tr>
<tr>
<td>DELETE</td>
<td>/roles/&lt;role&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>/roles</td>
</tr>
<tr>
<td>PUT</td>
<td>/roles/&lt;role&gt;/revoke</td>
</tr>
<tr>
<td>POST</td>
<td>/roles/&lt;role&gt;/authorizations</td>
</tr>
<tr>
<td>GET</td>
<td>/roles/&lt;role&gt;/authorizations /&lt;auth&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/roles/&lt;role&gt;/authorizations</td>
</tr>
<tr>
<td>PUT</td>
<td>/roles/&lt;role&gt;/authorizations /&lt;auth&gt;</td>
</tr>
</tbody>
</table>
List Roles

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

**TABLE 42** Role Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>name</td>
<td>Role name (immutable after creation)</td>
</tr>
<tr>
<td>string</td>
<td>description</td>
<td>Description of role</td>
</tr>
</tbody>
</table>

Example Request:

GET /api/role/v1/roles HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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 query parameter to true.

Example Request:
Create Role

This command creates a new role.

TABLE 43 Create New Role Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>name</td>
<td>New role's name (required)</td>
</tr>
<tr>
<td>string</td>
<td>clone</td>
<td>Name of role to clone original properties (optional)</td>
</tr>
<tr>
<td>string</td>
<td>description</td>
<td>Role description (required)</td>
</tr>
</tbody>
</table>

Example Request:

POST /api/role/v1/roles HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.example.com:215
Accept: application/json
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: zfssa.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

Revoke Role

Revoke Role
Revokes a role from all users.

Example Request:

PUT /api/role/v1/role_worksheets/revoke HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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

Delete Role

Delete Role
Deletes a role from the system. If the role is still assigned to one or more users, the query
parameter "confirm" must be set to "true".

Example Request:

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

Example Result:

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

List Role Authorizations

List Role Authorizations

List Role Authorizations
Lists the authorizations for the selected role.

Example Request:

GET /api/role/v1/roles/role_workflow/authorizations HTTP/1.1
Create Role Authorization

Authorization: Basic abcefgMWE=
Host: zfssa.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

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

Example Request:

POST /api/role/v1/roles/role_workflow/authorizations HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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",
    
```
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: zfssa.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: zfssa.example.com:215
Delete Role Authorization

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 may even be a cable between the initiator and the target, but in most cases there is some type of switching 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.
TABLE 44  Initiator Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /san/v1.0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/initiators</td>
<td>List all SAN initiators for the given protocol (FC, iSCSI, SRP) objects</td>
</tr>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/initiators /&lt;initiator&gt;</td>
<td>Get the specified SAN initiator for the given protocol (FC, iSCSI, SRP) properties</td>
</tr>
<tr>
<td>POST</td>
<td>/&lt;protocol&gt;/initiators</td>
<td>Create a new SAN initiator for the given protocol (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>PUT</td>
<td>/&lt;protocol&gt;/initiators /&lt;initiator&gt;</td>
<td>Modify the specified SAN initiator for the given protocol (FC, iSCSI, SRP) object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/&lt;protocol&gt;/initiators /&lt;initiator&gt;</td>
<td>Destroy the specified initiator object</td>
</tr>
</tbody>
</table>

These commands use the following URI parameters.

TABLE 45  URI Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>The NAS protocol for the initiator (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>initiator</td>
<td>The iqn, wwn or eui of the initiator</td>
</tr>
</tbody>
</table>

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

TABLE 46  Initiator Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>all</td>
<td>Alias for this initiator</td>
</tr>
<tr>
<td>initiator</td>
<td>fc</td>
<td>Port world wide name for this initiator (WWN)</td>
</tr>
<tr>
<td>ign</td>
<td>iscsi</td>
<td>iSCSI qualified name for this initiator</td>
</tr>
<tr>
<td>chapuser</td>
<td>iscsi</td>
<td>Challenge handshake auth protocol (CHAP) user name</td>
</tr>
<tr>
<td>chapsecret</td>
<td>iscsi</td>
<td>Challenge handshake auth protocol (CHAP) secret</td>
</tr>
<tr>
<td>initiator</td>
<td>srp</td>
<td>Extended Unique Identifier (EUI)</td>
</tr>
</tbody>
</table>

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",
    "initiator": "iqn.1986-03.com.sun:02:02",
    "chapsecret": "",
    "chapuser": ""
  },
  {"alias": "init-01",
    "initiator": "iqn.1986-03.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.1986-03.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",
    "initiator": "iqn.1986-03.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:

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

{
    "initiator": "iqn.1986-03.com.sun:02:02",
    "alias":"init-02"
}
```

Example Response:

```plaintext
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.1986-03.com.sun:02:02

{
    "initiator": {
        "alias": "init-02",
        "initiator": ".iqn.1986-03.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.1986-03.com.sun:01/ HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json

{
   "alias": "init-01-secure",
   "chapuser": "chuck",
   "chapsecret": "igottheintersect"
}

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",
      "iqn": "iqn.1986-03.com.sun:01",
      "chapsecret": "igottheintersect",
      "chapuser": "chuck"
   }
}

Delete an Initiator

Removes an initiator from the appliance.

Example Request:

DELETE /api/san/v1/iscsi/initiators/iqn.1986-03.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.
### TABLE 47  Initiator Group Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /san/v1.0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/initiator-groups</td>
<td>List all SAN initiator groups for the given protocol (FC, iSCSI, SRP) objects</td>
</tr>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/initiator-groups /&lt;name&gt;</td>
<td>Get the specified SAN initiator group for the given protocol (FC, iSCSI, SRP) properties</td>
</tr>
<tr>
<td>POST</td>
<td>/&lt;protocol&gt;/initiator-groups</td>
<td>Create a new SAN initiator group for the given protocol (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>PUT</td>
<td>/&lt;protocol&gt;/initiator-groups /&lt;name&gt;</td>
<td>Modify the specified SAN initiator group for the given protocol (FC, iSCSI, SRP) object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/&lt;protocol&gt;/initiator-groups /&lt;name&gt;</td>
<td>Destroy the specified name object</td>
</tr>
</tbody>
</table>

These commands use the following URI parameters.

### TABLE 48  URI Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>The NAS protocol for the initiator (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>name</td>
<td>The name of the initiator group.</td>
</tr>
</tbody>
</table>

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

### 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": [{
```
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"
  
  "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>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The name of the initiator group</td>
</tr>
<tr>
<td>initiators</td>
<td>array</td>
<td>An array of existing initiator IQN properties</td>
</tr>
</tbody>
</table>

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.1986-03.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.1986-03.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 below.

<table>
<thead>
<tr>
<th>TABLE 50</th>
<th>Target Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request</strong></td>
<td><strong>Path /san/v1.0</strong></td>
</tr>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/targets</td>
</tr>
</tbody>
</table>
Targets

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /san/v1.0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/targets/&lt;target&gt;</td>
<td>Get the specified SAN target for the given protocol (FC, iSCSI, SRP) properties</td>
</tr>
<tr>
<td>POST</td>
<td>/&lt;protocol&gt;/targets</td>
<td>Create a new SAN target for the given protocol (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>PUT</td>
<td>/&lt;protocol&gt;/targets/&lt;target&gt;</td>
<td>Modify the specified SAN target for the given protocol (FC, iSCSI, SRP) object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/&lt;protocol&gt;/targets/&lt;target&gt;</td>
<td>Destroy the specified target object</td>
</tr>
</tbody>
</table>

The commands take the following URI parameters

**TABLE 51** URI Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>protocol</td>
<td>The SAN protocol (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>string</td>
<td>target</td>
<td>The target ID (IQN, WWN, EUI)</td>
</tr>
</tbody>
</table>

All the "get" target commands return target properties, and the "create" and modify "target" commands use the following properties as input.

**TABLE 52** Target Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>iscsi</td>
<td>Simple human readable name</td>
</tr>
<tr>
<td>iqn</td>
<td>iscsi</td>
<td>The iSCSI qualified name</td>
</tr>
<tr>
<td>state</td>
<td>iscsi</td>
<td>State of the iSCSI target (&quot;online&quot;, &quot;offline&quot;)</td>
</tr>
<tr>
<td>auth</td>
<td>iscsi</td>
<td>Optional authentication type (&quot;none&quot;, &quot;chap&quot;)</td>
</tr>
<tr>
<td>targetchapuser</td>
<td>iscsi</td>
<td>Optional CHAP user authentication</td>
</tr>
<tr>
<td>targetchapsecret</td>
<td>iscsi</td>
<td>Optional CHAP secret authentication</td>
</tr>
<tr>
<td>interfaces</td>
<td>iscsi</td>
<td>List of network interfaces that target is available</td>
</tr>
<tr>
<td>wwn</td>
<td>fc</td>
<td>Worldwide name for this target</td>
</tr>
<tr>
<td>port</td>
<td>fc</td>
<td>Physical location of the port</td>
</tr>
<tr>
<td>mode</td>
<td>fc</td>
<td>Mode of this port (initiator or target)</td>
</tr>
<tr>
<td>speed</td>
<td>fc</td>
<td>Negotiated speed of this port</td>
</tr>
<tr>
<td>discovered_ports</td>
<td>fc</td>
<td>Number of discovered remote initiator ports</td>
</tr>
<tr>
<td>alias</td>
<td>srp</td>
<td>Alias for the SRP target</td>
</tr>
<tr>
<td>eui</td>
<td>srp</td>
<td>Extended unique identifier for this target</td>
</tr>
</tbody>
</table>
The following properties are used for getting iSCSI target group information.

**TABLE 53  Target Group Properties**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>protocol</td>
<td>The target group protocol (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>string</td>
<td>name</td>
<td>The iSCSI target group name</td>
</tr>
<tr>
<td>array</td>
<td>targets</td>
<td>A list of iSCSI target IQN group members</td>
</tr>
</tbody>
</table>

**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.1986-03.com.sun:02:72b6fa9a-96c4-e511-db19-aadb9bac2052",
            "iqn": "iqn.1986-03.com.sun:02:72b6fa9a-96c4-e511-db19-aadb9bac2052",
            ...
        },
        {
            "alias": "tst.volumes.py.96238.target",
            "iqn": "iqn.1986-03.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.1986-03.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
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":"letmeinnowplease", "auth":"chap",
 "targetchapuser":"argus"}

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",
    "targetchapuser": "letmeinnowplease",
    "alias": "test-target",
    "href": "/api/san/v1/iscsi/targets/alias=test-target",
    "auth": "none",
    "targetchapuser": "",
    "targetchapsecret": "",
    "interfaces": ["ixgbe0"]
  }
}
"interfaces": ["ixgbe0"]
}
}

## Delete a Target

Removes a SAN target from the system.

Example Request:

DELETE /api/san/v1/iscsi/targets/iqn.1986-03.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.

### TABLE 54 Target Group Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /san/v1.0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/target-groups</td>
<td>List all SAN target group for the given protocol (FC, iSCSI, SRP) objects</td>
</tr>
<tr>
<td>GET</td>
<td>/&lt;protocol&gt;/target-groups/&lt;target-group&gt;</td>
<td>Get the specified SAN target group for the given protocol (FC, iSCSI, SRP) properties</td>
</tr>
<tr>
<td>POST</td>
<td>/&lt;protocol&gt;/target-groups</td>
<td>Create a new SAN target group for the given protocol (FC, iSCSI, SRP)</td>
</tr>
<tr>
<td>PUT</td>
<td>/&lt;protocol&gt;/target-groups/&lt;target-group&gt;</td>
<td>Modify the specified SAN target group for the given protocol (FC, iSCSI, SRP) object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/&lt;protocol&gt;/target-groups/&lt;target-group&gt;</td>
<td>Destroy the specified target-group object</td>
</tr>
</tbody>
</table>

These commands use the following URI parameters.

### TABLE 55 URI Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>The NAS protocol for the initiator (FC, iSCSI, SRP)</td>
</tr>
</tbody>
</table>
### 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.1986-03.com.sun:02:31d26d2e-6aa0-6054-fe58-8b1f5b008b00"
         ]
      }, {
         "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 named "target-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:
Target Groups

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

{
    "group": {
        "href": "/api/san/v1/iscsi/target-groups/test-group",
        "name": "test-group",
        "targets": [
            "iqn.1986-03.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.1986-03.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.1986-03.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
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>
<thead>
<tr>
<th>Request</th>
<th>Path /service/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/services</td>
<td>List all services</td>
</tr>
<tr>
<td>GET</td>
<td>/services/&lt;service&gt;</td>
<td>Get configuration and status for the specified service</td>
</tr>
<tr>
<td>PUT</td>
<td>/services/&lt;service&gt;/enable</td>
<td>Enable the specified service</td>
</tr>
<tr>
<td>PUT</td>
<td>/services/&lt;service&gt;/disable</td>
<td>Disable the specified service</td>
</tr>
</tbody>
</table>

### 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": {
            "size": 2
        },
        "name": "smb"
    }, {
        "status": "online",
        "href": "/api/service/v1/services/dns",
        "log": {
            "size": 4
        },
        "name": "dns"
    }, {
        "status": "online",
        "href": "/api/service/v1/services/dynrouting",
        "log": {
            "size": 81
        },
        "name": "dynrouting"
    }, {
        "status": "disabled",
        "href": "/api/service/v1/services/ftp",
        "log": {
            "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",
        "name": "idmap"}]}
"log": {
  "href": "/api/log/v1/logs/system-idmap:default",
  "size": 15
},
"name": "idmap"
},
  {
  "status": "online",
  "href": "/api/service/v1/services/ipmp",
  "log": {
    "size": 3
  },
  "name": "ipmp"
},
  {
  "status": "online",
  "href": "/api/service/v1/services/iscsi",
  "log": {
    "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": {
    "size": 6
  },
  "name": "nfs"
},
  {
  "status": "disabled",
  "href": "/api/service/v1/services/nis",
  "log": {
    "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": {
   "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/ssh",
"log": {
   "size": 3
   },
"name": "ssh"
},
{
"<status>": "disabled",
"href": "/api/service/v1/services/syslog",
"name": "syslog"
},
{
"<status>": "online",
"href": "/api/service/v1/services/tags"}
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.

- **service** - The name of the service
- **state** - New service state ('enable', 'disable')

Example Request Using URI Parameters:

```
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 57** Service Sub Resource Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/services/&lt;service&gt;/&lt;resource&gt;</td>
<td>List service sub-resource</td>
</tr>
<tr>
<td>PUT</td>
<td>/services/&lt;service&gt;/&lt;resource &gt;/&lt;href&gt;</td>
<td>Modify sub-resource</td>
</tr>
<tr>
<td>POST</td>
<td>/services/&lt;service&gt;/&lt;resource&gt;</td>
<td>Create a new sub-resource</td>
</tr>
<tr>
<td>DELETE</td>
<td>/services/&lt;service&gt;/&lt;resource &gt;/&lt;href&gt;</td>
<td>Destroy an sub-resource</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>TABLE 58</th>
<th>Storage Pool Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Path /api/storage/v1</td>
</tr>
<tr>
<td>GET</td>
<td>/pools</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>/pools</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/scrub</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/scrub</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>TABLE 59</th>
<th>Storage Pool Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Name</td>
</tr>
<tr>
<td>string</td>
<td>pool</td>
</tr>
</tbody>
</table>
Get Pool

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>profile</td>
<td>Data device profile</td>
</tr>
<tr>
<td>string</td>
<td>state</td>
<td>The pool state (&quot;online&quot;, &quot;offline&quot;, &quot;exported&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>asn</td>
<td>The appliance serial number that owns the pool</td>
</tr>
<tr>
<td>string</td>
<td>peer</td>
<td>In a clustered system, the ASN of the peer node</td>
</tr>
<tr>
<td>string</td>
<td>owner</td>
<td>The hostname of the system that owns the pool</td>
</tr>
</tbody>
</table>

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

```json
{
    "pools": [
        {
            "profile": "mirror3",
            "name": "platinum",
            "peer": "00000000-0000-0000-0000-000000000000",
            "state": "online",
            "owner": "zfs-storage",
            "asn": "2f4aeeb3-b670-ee53-e0a7-d8e0ae410749"
        },
        {
            "profile": "raidz1",
            "name": "gold",
            "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/gold HTTP/1.1
Host: zfs-storage.example.com
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 query parameter properties to "true".

Example Request:

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

{  
  "name": "silver",
}

Example Response:

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

"props": [{  
  "choices": ["custom" ],
  "label": "Chassis 0",
  "name": "0",
}]}
Configure Pool

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

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

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

Example Response:

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

{
    "pool": {
        "asn": "314d252e-c42b-e844-dab1-a3bca600ba63",
        "errors": [],
        "name": "silver",
        "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 a PUT request to the pool href with the body containing the desired number of storage devices to add to the pool.

Example Request:

PUT /api/storage/v1/pools/gold HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbsfMWE=
Content-Type: application/json
Accept: application/json

{
    "2-data": 8
}

Example Response:

HTTP/1.1 202 Accepted

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/silver HTTP/1.1
Host: zfs-storage.example.com
Authorization: Basic abhadbsfMWE=

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 60** Project Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/storage/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/projects</td>
<td>List all projects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects</td>
<td>List projects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;</td>
<td>Get project details</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects</td>
<td>Create a project</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;</td>
<td>Modify a project</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;</td>
<td>Destroy a project</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/usage/groups</td>
<td>Get project group usage</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/usage/groups/&lt;group&gt;</td>
<td>Get project usage for the specified group</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/usage/users</td>
<td>Get project user usage</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/usage/users/&lt;user&gt;</td>
<td>Get project usage for the specified user</td>
</tr>
</tbody>
</table>

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

**TABLE 61** Project Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>aclinherit</td>
<td>ACL inheritance behavior (&quot;discard&quot;, &quot;noallow&quot;, &quot;restricted&quot;, &quot;passthrough&quot;, &quot;passthrough-x&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>aclmode</td>
<td>ACL behavior on mode change (&quot;discard&quot;, &quot;mask&quot;, &quot;passthrough&quot;)</td>
</tr>
<tr>
<td>boolean</td>
<td>atime</td>
<td>Update access time on read flag</td>
</tr>
<tr>
<td>string</td>
<td>canonical_name</td>
<td>Canonical name</td>
</tr>
<tr>
<td>string</td>
<td>checksum</td>
<td>Block checksum (&quot;fletcher2&quot;, &quot;fletcher4&quot;, &quot;sha256&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>compression</td>
<td>Data compression setting (&quot;off&quot;, &quot;lzjb&quot;, &quot;gzip-2&quot;, &quot;gzip&quot;, &quot;gzip-9&quot;)</td>
</tr>
<tr>
<td>number</td>
<td>copies</td>
<td>Number of additional replication copies</td>
</tr>
<tr>
<td>datetime</td>
<td>creation</td>
<td>Date and time of project (or LUN, filesystem) creation</td>
</tr>
<tr>
<td>boolean</td>
<td>dedup</td>
<td>Data deduplication flag</td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>string</td>
<td>default_group</td>
<td>Project default filesystem group: &quot;other&quot;</td>
</tr>
<tr>
<td>string</td>
<td>default_permissions</td>
<td>Project default filesystem permissions &quot;700&quot;</td>
</tr>
<tr>
<td>boolean</td>
<td>default_sparse</td>
<td>Project default LUN sparse data flag</td>
</tr>
<tr>
<td>string</td>
<td>default_user</td>
<td>Project default filesystem user: &quot;nobody&quot;</td>
</tr>
<tr>
<td>number</td>
<td>default_volblocksize</td>
<td>Project default LUN blocksize: 8192</td>
</tr>
<tr>
<td>number</td>
<td>default_volsize</td>
<td>Project default LUN Size</td>
</tr>
<tr>
<td>boolean</td>
<td>exported</td>
<td>Exported flag</td>
</tr>
<tr>
<td>string</td>
<td>logbias</td>
<td>Synchronous write bias (&quot;latency&quot;, &quot;throughput&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>mountpoint</td>
<td>Share mountpoint default &quot;/export/proj-01&quot;</td>
</tr>
<tr>
<td>string</td>
<td>name</td>
<td>Project Name</td>
</tr>
<tr>
<td>boolean</td>
<td>nbmand</td>
<td>Non-blocking mandatory locking flag</td>
</tr>
<tr>
<td>boolean</td>
<td>nodestroy</td>
<td>Prevent destruction flag</td>
</tr>
<tr>
<td>number</td>
<td>quota</td>
<td>Project Quota Size in bytes</td>
</tr>
<tr>
<td>string</td>
<td>origin</td>
<td>Clone origin</td>
</tr>
<tr>
<td>string</td>
<td>pool</td>
<td>Pool namels</td>
</tr>
<tr>
<td>boolean</td>
<td>readonly</td>
<td>Data is read only if set to true</td>
</tr>
<tr>
<td>string</td>
<td>recordsize</td>
<td>Database record size &quot;128k&quot;</td>
</tr>
<tr>
<td>number</td>
<td>reservation</td>
<td>Data Reservation Size</td>
</tr>
<tr>
<td>boolean</td>
<td>rstchown</td>
<td>Restrict Ownership change flag</td>
</tr>
<tr>
<td>string</td>
<td>secondarycache</td>
<td>Secondary Cache Usage (&quot;all&quot;, &quot;metadata&quot;, &quot;none&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>sharedav</td>
<td>HTTP share (&quot;off&quot;, &quot;rw&quot;, &quot;ro&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>shareftp</td>
<td>FTP share (&quot;off&quot;, &quot;rw&quot;, &quot;ro&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>sharesfs</td>
<td>NFS share (&quot;off&quot;, &quot;on&quot;, &quot;ro&quot;, &quot;rw&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>shareftp</td>
<td>SFTP share (&quot;off&quot;, &quot;rw&quot;, &quot;ro&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>shareftp</td>
<td>SMB/CIFS share (&quot;off&quot;, &quot;rw&quot;, &quot;ro&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>shareftp</td>
<td>TFTP share (&quot;off&quot;, &quot;rw&quot;, &quot;ro&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>snapdir</td>
<td>.zfs/snapshots visibility (&quot;hidden&quot;, &quot;visible&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>snaplabel</td>
<td>Scheduled Snapshot Label</td>
</tr>
<tr>
<td>boolean</td>
<td>vscan</td>
<td>Virus Scan flag</td>
</tr>
</tbody>
</table>
List Projects

This command lists all of the projects in a given pool. 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.

**Query Parameters filter**— A simple string match filter that requires a property within the project to contain the same filter string within its value.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
</tbody>
</table>

Example Request:

```
GET /api/storage/v1/pools/gold/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 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.

**TABLE 63** Get Project URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
</tbody>
</table>

Example Request to List Project Named "proj-01" in the "gold" Pool:

GET /api/storage/v1/pools/gold/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

```json
{
  "project": {
    "default_volblocksize": 8192.0,
    "logbias": "latency",
    "creation": "20130411T20:02:35",
    "nodestroy": false,
    "dedup": false,
    "sharefs": "on",
    "sharesmb": "off",
    "default_permissions": "700",
    "mountpoint": "/export",
    "snaplabel": "",
    "id": "042919bb-0882-d903-0000-000000000000",
    "readonly": false,
    "rrsrc_actions": [],
    "compression": "off",
    "shareftp": "",
    "default_sparse": false,
    "snapdir": "hidden",
    "aclmode": "discard",
    "copies": 1,
    "aclinherit": "restricted",
    "shareftp": "",
    "canonical_name": "gold/local/default",
    "recordsize": 131072.0,
    "usage": {
      "available": 1758424767306.0,
      "loading": false,
      "quota": 0.0,
    }
  }
}
```
Create Project

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

TABLE 64  URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
</tbody>
</table>

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/gold/projects HTTP/1.1
Host: zfs-storage.example.com
Content-Type: application/json
Accept: application/json
```
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

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

Modify Project

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

**TABLE 65**  URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
</tbody>
</table>

**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/gold/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/gold/projects/new-name

{
    "project": {
        "name": "new-name",
        "share nfs": "rw",
        "compression": "gzip-9",
        ...
    }
}

Delete Project

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

TABLE 66 URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
</tbody>
</table>

Example Request:

DELETE /api/storage/v1/pools/gold/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 67  Filesystem Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/storage/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/filesystems</td>
<td>List all filesystems</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems</td>
<td>List filesystems</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;</td>
<td>Get filesystem details</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems</td>
<td>Create a filesystem</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;</td>
<td>Modify a filesystem</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;</td>
<td>Destroy a filesystem</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/groups</td>
<td>Get filesystem group usage</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/groups/&lt;group&gt;</td>
<td>Get filesystem usage for the specified group</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/groups</td>
<td>Create a filesystem group quota</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/groups/&lt;name&gt;</td>
<td>Modify a filesystem group quota</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/users</td>
<td>Get filesystem user usage</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/users/&lt;user&gt;</td>
<td>Get filesystem usage for the specified user</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/users</td>
<td>Create a filesystem user quota</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/usage/users/&lt;name&gt;</td>
<td>Modify a filesystem user quota</td>
</tr>
</tbody>
</table>

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

### TABLE 68  Filesystem Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>casesensitivity</td>
<td>Case Sensitivity setting (&quot;mixed&quot;, &quot;sensitive&quot; or &quot;insensitive&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>group</td>
<td>The group name</td>
</tr>
</tbody>
</table>
List Filesystems

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

**Query Parameters - filter**– A simple string match filter that requires a property within the project to contain the same filter string within its value.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
</tbody>
</table>

Example Request:

```
GET /api/storage/v1/pools/gold/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",
```
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>The filesystem name</td>
</tr>
</tbody>
</table>

Example Request (to list project named "proj-01"):

GET /api/storage/v1/pools/gold/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,  
      "pool": "gold",  
      "project": "proj-01",  
      "name": "filesystem-02",  
      ...  
   },  
   ...  
}  


"sharefs": "on",
"sharesmb": "off",
"mountpoint": "/export/grape",
"snaplabel": "",
"id": "424ca2ec-b3fa-df86-0000-000000000000",
"readonly": false,
"rrsrc_actions": [],
"compression": "off",
"shareftp": "",
"source": {
  "logbias": "default",
  "dedup": "default",
  "sharefs": "inherited",
  "sharesmb": "off",
  "mountpoint": "inherited",
  "rrsrc_actions": "local",
  "compression": "default",
  "shareftp": "inherited",
  "snapdir": "default",
  "aclmode": "default",
  "copies": "default",
  "aclinherit": "default",
  "shareftp": "inherited",
  "readonly": "default",
  "secondarcache": "default",
  "exported": "inherited",
  "vschan": "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": "platinum/local/default/grape",
"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
},
"secondarcache": "all",
"collection": "local"
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.

TABLE 71 URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>The filesystem name</td>
</tr>
</tbody>
</table>

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 "joe"):

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

{
   "name": "share-01",
   "owner": "joe"
}

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/gold/projects/proj-01/filesystems/share-01

{
   "filesystem": {
      "name": "share-01",
      "pool": "gold",
      "collection": "local",
      "project": "proj-01",
      "owner": "joe"
   }
}

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.

Request Parameters - Filesystem Properties – Any of the filesystem or project properties can be modified

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>The filesystem name</td>
</tr>
</tbody>
</table>

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

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

{
   "name": "new-name",
   "owner": "nobody",
}

Example Response:
HTTP/1.1 202 Accepted
Content-Type: application/json

{
   "filesystem": {
      "name": "new-name",
      "pool": "gold",
      "collection": "local",
      "project": "proj-01",
      "owner": "nobody"
   }...
}

Delete Filesystem

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

**TABLE 73**  URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>The filesystem name</td>
</tr>
</tbody>
</table>

Example Request:

DELETE /api/storage/v1/pools/gold/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.
All LUN or volume operations are scoped to a given pool or project. The following LUN commands are available.

**TABLE 74  Volume Commands**

<table>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/luns</td>
<td>List all LUNS</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns</td>
<td>List LUNS</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;</td>
<td>Get LUN details</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns</td>
<td>Create a LUN</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;</td>
<td>Modify a LUN</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;</td>
<td>Destroy a LUN</td>
</tr>
</tbody>
</table>

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

**TABLE 75  Volume Properties**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>assignednumber</td>
<td>The assigned LU number.</td>
</tr>
<tr>
<td>boolean</td>
<td>fixednumber</td>
<td>Flag to fix LU number at current value</td>
</tr>
<tr>
<td>string</td>
<td>initiatorgroup</td>
<td>The initiator group</td>
</tr>
<tr>
<td>string</td>
<td>lunguid</td>
<td>STMF GUID</td>
</tr>
<tr>
<td>string</td>
<td>lunnumber</td>
<td>The LU number. Either a number or &quot;auto&quot;</td>
</tr>
<tr>
<td>string</td>
<td>project</td>
<td>The project name (immutable)</td>
</tr>
<tr>
<td>object</td>
<td>source</td>
<td>Lists source of properties (&quot;local&quot;, &quot;inherited&quot;)</td>
</tr>
<tr>
<td>boolean</td>
<td>sparse</td>
<td>Flag to enable thin provisioning</td>
</tr>
<tr>
<td>string</td>
<td>status</td>
<td>Logical unit status (&quot;online&quot;, &quot;offline&quot;)</td>
</tr>
<tr>
<td>string</td>
<td>targetgroup</td>
<td>The target group</td>
</tr>
<tr>
<td>object</td>
<td>usage</td>
<td>Lists LUN usage statistics</td>
</tr>
<tr>
<td>number</td>
<td>volblocksize</td>
<td>Volume block size</td>
</tr>
<tr>
<td>number</td>
<td>volsize</td>
<td>Volume size</td>
</tr>
<tr>
<td>boolean</td>
<td>writecache</td>
<td>Flag to enable write cache</td>
</tr>
</tbody>
</table>
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 then 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
</tbody>
</table>

Example Request (to list LUNS within Project "proj-01"):  
GET /api/storage/v1/pools/gold/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

```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.
TABLE 77  URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>lun</td>
<td>The lun name</td>
</tr>
</tbody>
</table>

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

```plaintext
GET /api/storage/v1/pools/gold/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:

```plaintext
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": "600144F0D8E0AE4100005176F0A60001",
        "source": {
            "compression": "default",
            "checksum": "inherited",
            "logbias": "default",
            "dedup": "default",
            "copies": "default",
            "exported": "inherited",
            "rrsrc_actions": "inherited",
            "secondarycache": "default"
        },
        "canonical_name": "platinum/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": "platinum",
"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.

TABLE 78  URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
</tbody>
</table>

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/gold/projects/proj-01/luns HTTP/1.1
Host: zfs-storage.example.com
Accept: application/json

Request JSON:
```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,
}
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

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

Modify LUN

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>lun</td>
<td>The LUN name</td>
</tr>
</tbody>
</table>

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/gold/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/gold/projects/proj-01/luns/new-name

{
   "lun": {
      "name": "new-name",
      "pool": "gold",
      "collection": "local",
      "project": "proj-01",
      ...
   }
}

Delete Lun

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Storage pool name</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>lun</td>
<td>The LUN name</td>
</tr>
</tbody>
</table>

Example Request:

DELETE /pools/gold/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>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/snapshots</td>
<td>List all local snapshots</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/snapshots</td>
<td>List all project snapshots</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots</td>
<td>List all filesystem snapshots</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots</td>
<td>List all LUN snapshots</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Get project snapshot details</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Get filesystem snapshot details</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Get LUN snapshot details</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/snapshots</td>
<td>Create a project snapshot</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots</td>
<td>Create a filesystem snapshot</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots</td>
<td>Create a LUN snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Modify a project snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Modify a filesystem snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Modify a LUN snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots/&lt;snapshot&gt;/clone</td>
<td>Clone a filesystem snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;/clone</td>
<td>Clone a LUN snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots/&lt;snapshot&gt;/rollback</td>
<td>Rollback data to the given filesystem snapshot</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;/rollback</td>
<td>Rollback data to the given LUN snapshot</td>
</tr>
</tbody>
</table>
## Snapshot and Clone Operations

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/storage/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Destroy a project snapshot</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Destroy a filesystem snapshot</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;</td>
<td>Destroy a LUN snapshot</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/snapshots/&lt;snapshot&gt;/dependents</td>
<td>List project snapshot dependents</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/snapshots/&lt;snapshot&gt;/dependents</td>
<td>List filesystem snapshot dependents</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;/dependents</td>
<td>List LUN snapshot dependents</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/automatic</td>
<td>Create a new project automatic snapshot</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/automatic/&lt;automatic&gt;</td>
<td>Get the specified project automatic snapshot properties</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/automatic</td>
<td>List all project automatic snapshot objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/automatic/&lt;automatic&gt;</td>
<td>Modify the specified project automatic snapshot object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/automatic/&lt;automatic&gt;</td>
<td>Destroy the specified automatic object</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/automatic</td>
<td>Create a new filesystem automatic snapshot</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/automatic/&lt;automatic&gt;</td>
<td>Get the specified filesystem automatic snapshot properties</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/automatic</td>
<td>List all filesystem automatic snapshot objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/automatic/&lt;automatic&gt;</td>
<td>Modify the specified filesystem automatic snapshot object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/automatic/&lt;automatic&gt;</td>
<td>Destroy the specified automatic object</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic</td>
<td>Create a new LUN automatic snapshot</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic/&lt;automatic&gt;</td>
<td>Get the specified LUN automatic snapshot properties</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic</td>
<td>List all LUN automatic snapshot objects</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic/&lt;automatic&gt;</td>
<td>Modify the specified LUN automatic snapshot object</td>
</tr>
</tbody>
</table>
### Snapshot and Clone Operations

**Request**

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic/&lt;automatic&gt;</td>
<td>Destroy the specified automatic object</td>
</tr>
</tbody>
</table>

### List Snapshots

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

**TABLE 82** List Snapshot Command Forms

<table>
<thead>
<tr>
<th>Command</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Project Snapshots</td>
<td>/api/storage/v1/pools/{pool}/projects/{project}/snapshots</td>
</tr>
<tr>
<td>List filesystem snapshots</td>
<td>/api/storage/v1/pools/{pool}/projects/{project}/filesystems/{share}/snapshots</td>
</tr>
<tr>
<td>List lun snapshots</td>
<td>/api/storage/v1/pools/{pool}/projects/{project}/lun/{share}/snapshots</td>
</tr>
</tbody>
</table>

**Example Request:**

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

**Example Response:**

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

```json
{
    "snapshots": [
        {
            "id": "3fbbcccf-d058-4502-8844-6feffdf4cb5",
            "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": "76b0950a-8594-4e5b-8dce-0d9c696350",
            "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/gold/projects/default/snapshots/snap-001
Accept: application/json

Example Response:

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

{
    "snapshot": {
        "id": "3fbbcccf-d058-4502-8844-6feefd4cb5",
        "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/gold/projects/default/snapshots
Content-Type: application/json

{"name": "initial-backup"}

Example Response:

HTTP/1.1 201 Created
Content-Type: application/json
Location: /pools/gold/projects/default/snapshot/initial-backup
Snapshot and Clone Operations

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

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/gold/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/gold/projects/default/snapshot/initial-backup
```

Clone Snapshot

Makes a new filesystem or LUN from an existing snapshot.

**Request URI** Clone Filesystem:

```
PUT /pools/{pool}/projects/{project}/filesystems/{share}/snapshots/{snap}/clone
```

**Clone Volume:**
PUT /pools/{pool}/projects/{project}/luns/{lun}/snapshots/{snapshot}/clone

TABLE 83   URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Source pool name</td>
</tr>
<tr>
<td>project</td>
<td>Source project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>Source share name (for filesystem snapshot)</td>
</tr>
<tr>
<td>lun</td>
<td>Source share name (for LUN snapshot)</td>
</tr>
<tr>
<td>snapshot</td>
<td>Source snapshot name</td>
</tr>
</tbody>
</table>

Request body contains a JSON object with the following properties.

TABLE 84   Clone Snapshot Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>pool</td>
<td>Destination clone pool name</td>
</tr>
<tr>
<td>string</td>
<td>project</td>
<td>Destination clone project name</td>
</tr>
<tr>
<td>string</td>
<td>lun</td>
<td>Destination LUN name (for LUN snapshot)</td>
</tr>
</tbody>
</table>

Example Request:

PUT /api/storage/v1/pools/gold/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/gold/projects/rest/filesystem/snap01clone01
Content-Type: application/json; charset=utf-8

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

Rollback a filesystem snapshot:

PUT /pools/{pool}/projects/{project}/filesystems/{share}/snapshots/{snap}/rollback

Rollback a LUN Snapshot:

PUT /pools/{pool}/projects/{project}/luns/{lun}/snapshots/{snapshot}/rollback

**TABLE 85 URI Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Source pool name</td>
</tr>
<tr>
<td>project</td>
<td>Source project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>Source filesystem name (for filesystem snapshot)</td>
</tr>
<tr>
<td>lun</td>
<td>Source LUN name (for LUN snapshot)</td>
</tr>
<tr>
<td>snapshot</td>
<td>Source snapshot name</td>
</tr>
</tbody>
</table>

Example Request:

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

Example Response:

HTTP/1.1 202 Accepted
Location: /pools/gold/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",
    "..."canonical_name": "gold/local/rest/snap01clone01"
  }
}
Delete a Snapshot

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

- Delete a single snapshot with the given snapshot - DELETE /snapshots/{snapshot_id}
- Delete a project snapshot with the given pool name, project name, and snapshot name - DELETE /pools/{pool}/projects/{project}/snapshots/{snapshot_name}
- Delete a filesystem snapshot with the given pool name, project name, filesystem name, and snapshot name - DELETE /pools/{pool}/projects/{project}/filesystems/{share}/snapshots/{snapshot_name}
- Delete a filesystem LUN with the given pool name, project name, filesystem name, and snapshot name - DELETE /pools/{pool}/projects/{project}/luns/{lun}/snapshots/{snapshot_name}

Example Request:

PUT /pools/gold/projects/default/filesystems/fs-01/snapshots/initial-backup HTTP/1.1

Successful delete returns HTTP Status 204 (not Content).

HTTP/1.1 204 No-Content

List Snapshot Dependents

Lists dependents for a filesystem or volume.

<table>
<thead>
<tr>
<th>Command</th>
<th>Command Forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Filesystem Dependents</td>
<td>/filesystems/{share}/snapshots/{snapshot}/dependents</td>
</tr>
<tr>
<td>/api/storage/v1/pools/{pool}/projects/{project}</td>
<td></td>
</tr>
</tbody>
</table>
List Volume Dependents

/lun/{lun}/snapshots/{snapshot}/dependents

**TABLE 87** URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>Name of system storage pool</td>
</tr>
<tr>
<td>project</td>
<td>The project name</td>
</tr>
<tr>
<td>filesystem</td>
<td>The filesystem name</td>
</tr>
<tr>
<td>lun</td>
<td>The LUN name</td>
</tr>
</tbody>
</table>

**Example Request:**

GET /api/storage/v1/pools/gold/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

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

**Schema**

Manages custom schema properties.
<table>
<thead>
<tr>
<th>TABLE 88</th>
<th>Schema Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request</strong></td>
<td><strong>Path/api/storage/v1</strong></td>
</tr>
<tr>
<td>GET</td>
<td>/schema</td>
</tr>
<tr>
<td>GET</td>
<td>/schema/property</td>
</tr>
<tr>
<td>POST</td>
<td>/schema</td>
</tr>
<tr>
<td>PUT</td>
<td>/schema/property</td>
</tr>
<tr>
<td>DELETE</td>
<td>/schema/property</td>
</tr>
</tbody>
</table>

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 customer int property named "priority":

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

<table>
<thead>
<tr>
<th>TABLE 89</th>
<th>Schema Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>property</td>
<td>Name of property (immutable)</td>
</tr>
<tr>
<td>description</td>
<td>Property description (for browser interface)</td>
</tr>
<tr>
<td>type</td>
<td>Type (&quot;String&quot;, &quot;Integer&quot;, &quot;PositiveInteger&quot;, &quot;Boolean&quot;, &quot;EmailAddress&quot;, &quot;Host&quot;)</td>
</tr>
</tbody>
</table>

### List Properties

Lists schema properties.

**Example Request:**

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

**Example Result:**

```json
{
    "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: zfssa.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

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 of the Oracle ZFS Storage Appliance and the replication RESTful API manages that feature. The service is available from the following URI: https://host.example.com:215/api/storage/v1/replication

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 90 Replication Service Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/service/v1/services</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/replication</td>
<td>Get replication service state properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/enable</td>
<td>Enable the replication service</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/disable</td>
<td>Disable the replication service</td>
</tr>
</tbody>
</table>

### Get Replication Service

Gets the state of the replication service.

Example Request:

```
GET /api/service/v1/services/replication HTTP/1.1
Host: zfssa.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>
<thead>
<tr>
<th>Request</th>
<th>Path ( /api/service/v1/services )</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>( /replication/targets )</td>
<td>Create a new replication target</td>
</tr>
<tr>
<td>GET</td>
<td>( /replication/targets/&lt;target&gt; )</td>
<td>Get the specified replication target properties</td>
</tr>
<tr>
<td>GET</td>
<td>( /replication/targets )</td>
<td>List all replication target objects</td>
</tr>
<tr>
<td>PUT</td>
<td>( /replication/targets/&lt;target&gt; )</td>
<td>Modify the specified replication target object</td>
</tr>
<tr>
<td>DELETE</td>
<td>( /replication/targets/&lt;target&gt; )</td>
<td>Destroy the specified target object</td>
</tr>
</tbody>
</table>

**List Replication Targets**

Lists all of the available replication targets on a system.

Example Request:

GET  /api/service/v1/services/replication/targets HTTP/1.1
Host: zfssa-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

{
Get Replication Target

This command lists the details of a single replication target including the list of available storage pools that can be used for destination data. The target is accessed by its href (target-000) or by a <name>=<value> selection.

Example Request:
GET api/service/v1/services/replication/targets/hostname=luxor 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

{
    "hostname=luxor": {
        "actions": 0,
        "address": "10.80.231.52:216",
        "asn": "fa5bf303-0dcb-e20d-ac92-cd129ccd2c81",
        "hostname": "luxor",
        "href": "/service/v1/services/replication/targets/target-000",
        "label": "luxor"
    }
}

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: example.zfssa.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 54

{"hostname":"example", "root_password":"letmein", "label":"east"}

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": "east"
  }
}

**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.
### TABLE 92 Replication Action Commands

<table>
<thead>
<tr>
<th>Request</th>
<th>/api/storage/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/replication/actions</td>
<td>List all replication action objects</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/actions/&lt;ra_id&gt;</td>
<td>Get the specified replication action properties</td>
</tr>
<tr>
<td>POST</td>
<td>/replication/actions</td>
<td>Create a new replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/actions/&lt;ra_id&gt;</td>
<td>Modify the specified replication action object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/replication/actions/&lt;ra_id&gt;</td>
<td>Delete the specified replication action object</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/actions/&lt;ra_id&gt; /sendupdate</td>
<td>Start the selected replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/actions/&lt;ra_id&gt; /cancelupdate</td>
<td>Stop the selected replication action</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/actions/&lt;ra_id&gt; /schedules</td>
<td>List all replication action schedule objects</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/actions/&lt;ra_id&gt; /schedules/&lt;ra_schedule&gt;</td>
<td>Get the specified replication action schedule properties</td>
</tr>
<tr>
<td>POST</td>
<td>/replication/actions/&lt;ra_id&gt; /schedules</td>
<td>Create a new replication action schedule</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/actions/&lt;ra_id&gt; /schedules/&lt;ra_schedule&gt;</td>
<td>Modify the specified replication action schedule object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/replication/actions/&lt;ra_id&gt; /schedules/&lt;ra_schedule&gt;</td>
<td>Delete the specified replication action schedule object</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions</td>
<td>List all replication action objects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Get the specified replication action properties</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions</td>
<td>Create a new replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Modify the specified replication action object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Delete the specified replication action object</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/sendupdate</td>
<td>Start the selected replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/cancelupdate</td>
<td>Stop the selected replication action</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/schedules</td>
<td>List all replication action schedule objects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Get the specified replication action schedule properties</td>
</tr>
</tbody>
</table>
## Replication Actions

<table>
<thead>
<tr>
<th>Request</th>
<th>/api/storage/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/schedules</td>
<td>Create a new replication action schedule</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Modify the specified replication action schedule object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Delete the specified replication action schedule object</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions</td>
<td>List all replication action objects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Get the specified replication action properties</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions</td>
<td>Create a new replication action object</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Modify the specified replication action object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Delete the specified replication action object</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication</td>
<td>Get filesystem replication action settings</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication</td>
<td>Modify filesystem replication action settings</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/sendupdate</td>
<td>Start the selected replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/cancelupdate</td>
<td>Stop the selected replication action</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/schedules</td>
<td>List all replication action schedule objects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Get the specified replication action schedule properties</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/schedules</td>
<td>Create a new replication action schedule</td>
</tr>
<tr>
<td>Request</td>
<td>/api/storage/v1</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Modify the specified replication action schedule object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/filesystems/&lt;filesystem&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Delete the specified replication action schedule object</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions</td>
<td>List all replication action objects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Get the specified replication action properties</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions</td>
<td>Create a new replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Modify the specified replication action object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;</td>
<td>Delete the specified replication action object</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication</td>
<td>Get LUN replication action settings</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication</td>
<td>Modify LUN replication action settings</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/sendupdate</td>
<td>Start the selected replication action</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/cancelupdate</td>
<td>Stop the selected replication action</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/schedules</td>
<td>List all replication action schedule objects</td>
</tr>
<tr>
<td>GET</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Get the specified replication action schedule properties</td>
</tr>
<tr>
<td>POST</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/schedules</td>
<td>Create a new replication action schedule</td>
</tr>
<tr>
<td>PUT</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Modify the specified replication action schedule object</td>
</tr>
<tr>
<td>DELETE</td>
<td>/pools/&lt;pool&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/replication/actions/&lt;ra_id&gt;/schedules/&lt;ra_schedule&gt;</td>
<td>Delete the specified replication action schedule object</td>
</tr>
</tbody>
</table>
List Replication Actions

Gets a list of all available replication actions.

Example Request:

GET /api/storage/v1/replication/actions HTTP/1.1
Authorization: Basic ab6rt4psMW==
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 ab6rt4psMW==
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",
    "continuous": false,
    "enabled": true,
Create Replication Action

Creates a new replication action.

Create Properties:

Initial values:

- target = (unset)
- pool = (unset)
- enabled = true
- continuous = false
- include_snaps = true
- max_bandwidth = unlimited
- bytes_sent = 0
- estimated_size = 0
- estimated_time_left = 0
- average_throughput = 0
- use_ssl = true

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

```json
{
    "pool": "gold",
    "project": "blue1",
    "share": "fs1",
    "target": "38094753-6c90-49ed-aa92-995a296d432a",
    "use_ssl": true
}
```
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,
        "continuous": false,
        "enabled": true,
        "max_bandwidth": 0,
        "collection": "local",
        "estimated_size": 0.0,
        "state": "idle",
        "href": "/api/storage/v1/replication/pools/gold/projects/blah1/shares/fs1/actions/8373d331-de60-e590-90e8-9ad69fcb4aec",
        "average_throughput": 0.0,
        "use_ssl": true,
        "estimated_time_left": 0.0,
        "share": "fs1",
        "id": "8373d331-de60-e590-90e8-9ad69fcb4aec",
        "pool": "gold",
        "include_snaps": true
    }
}

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": {

Replication Actions

{
  "target_id": "407642ae-91b5-681c-de5e-afcd5cbf2974",
  "continuous": false,
  "enabled": true,
  "max_bandwidth": 0,
  "dedup": false,
  "use_ssl": false,
  "id": "c141d88d-ffd2-6730-d489-b71905f340cc",
  "include_snaps": true
}

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

Replication source and package commands.

<table>
<thead>
<tr>
<th>TABLE 93</th>
<th>Replication Source and Package Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>/api/storage/v1</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;</td>
</tr>
<tr>
<td>DELETE</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt; /cancelupdate</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/sever</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/reverse</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/clone</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/clone/conflicts</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/projects</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/projects /&lt;project&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/projects /&lt;project&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; /packages/&lt;package&gt;/projects /&lt;project&gt;/usage/groups</td>
</tr>
<tr>
<td>Request</td>
<td><code>/api/storage/v1</code></td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/usage/users</code></td>
</tr>
<tr>
<td>POST</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/snapshots</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/snapshots/&lt;snapshot&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/snapshots</code></td>
</tr>
<tr>
<td>DELETE</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/snapshots/&lt;snapshot&gt;</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/snapshots/&lt;snapshot&gt;</code></td>
</tr>
<tr>
<td>POST</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/automatic</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/automatic /&lt;automatic&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/automatic</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/automatic /&lt;automatic&gt;</code></td>
</tr>
<tr>
<td>DELETE</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/automatic /&lt;automatic&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/usage/groups</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects /</code></td>
</tr>
<tr>
<td>Request</td>
<td>/api/storage/v1</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>POST</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ snapshots</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ snapshots /&lt;snapshot&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ snapshots</td>
</tr>
<tr>
<td>DELETE</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ snapshots /&lt;snapshot&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ snapshots /&lt;snapshot&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ automatic</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ automatic /&lt;automatic&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ automatic</td>
</tr>
<tr>
<td>PUT</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ automatic /&lt;automatic&gt;</td>
</tr>
<tr>
<td>DELETE</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/filesystems /&lt;filesystem&gt;/ automatic /&lt;automatic&gt;</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/luns</td>
</tr>
<tr>
<td>GET</td>
<td>/replication/sources/&lt;source&gt; / packages/&lt;package&gt;/projects / &lt;project&gt;/luns/&lt;lun&gt;</td>
</tr>
<tr>
<td>Request</td>
<td><code>/api/storage/v1</code></td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>PUT</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/usage/groups</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/usage/users</code></td>
</tr>
<tr>
<td>POST</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots</code></td>
</tr>
<tr>
<td>DELETE</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/snapshots/&lt;snapshot&gt;</code></td>
</tr>
<tr>
<td>POST</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic/&lt;automatic&gt;</code></td>
</tr>
<tr>
<td>GET</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic</code></td>
</tr>
<tr>
<td>PUT</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic/&lt;automatic&gt;</code></td>
</tr>
<tr>
<td>DELETE</td>
<td><code>/replication/sources/&lt;source&gt;/packages/&lt;package&gt;/projects/&lt;project&gt;/luns/&lt;lun&gt;/automatic/&lt;automatic&gt;</code></td>
</tr>
</tbody>
</table>
**List Replication Sources**

Lists all available replication sources.

Example Request:

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

Example Output:

```plaintext
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/zfssa-repl-host",
    "ip_address": "10.80.231.58:216",
    "name": "zfssa-repl-host",
    "source": "source-000"
  }
}
```

**List Replication Packages**

Lists all of the replication packages from the specified replication source.

Example Request:

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

Example Result:

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

{
  "packages": [{
    "enabled": true,
    "href": "/api/v1/storage/replication/sources/zfssa-repl/packages/package-008",
    "id": "b2dc835a-a5a0-6c74-f7e9-b75c357e841f",
    "last_result": "unknown",
    "last_sync": "unknown",
```
Modify Package

Modifies the package properties.

TABLE 94 Modify Package Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>enabled</td>
<td>Current state of replication updates</td>
</tr>
</tbody>
</table>

Example Request:

```plaintext
PUT /api/storage/v1/replication/sources/zfssa-repl/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

Delete Package

Destroys a replication package.

Example Request:

```plaintext
DELETE /api/storage/v1/replication/sources/zfssa-repl/packages
```
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/sources/zfssa-repl/packages/
   8373d331-de60-e590-90e8-9ad69fcb4aecd/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

```json
{
   "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/sources/zfssa-repl/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 if there are conflicts with the clone operation.

Example Clone Conflicts Request:

GET /api/storage/v1/replication/sources/zfssa-repl/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.\n"
}

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/sources/zfssa-repl/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/sources/zfssa-repl/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

---

**LICENSE NOTICE:** Encryption may be evaluated free of charge, but the feature requires that an independent license be purchased separately for use in production. After the evaluation period, this feature must either be licensed or deactivated. Oracle reserves the right to audit for licensing compliance at any time. For details, refer to the "Oracle Software License Agreement ("SLA") and Entitlement for Hardware Systems with Integrated Software Options."

---

The 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 95** Local Encryption

<table>
<thead>
<tr>
<th>Request</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/encryption/local</td>
<td>Get local keystore properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/encryption/local</td>
<td>Modify local keystore properties</td>
</tr>
<tr>
<td>GET</td>
<td>/encryption/local/keys</td>
<td>Get local keys</td>
</tr>
<tr>
<td>GET</td>
<td>/encryption/local/keys/&lt;key&gt;</td>
<td>Get local key details</td>
</tr>
<tr>
<td>POST</td>
<td>/encryption/local/keys</td>
<td>Create a local key</td>
</tr>
<tr>
<td>DELETE</td>
<td>/encryption/local/keys/&lt;key&gt;</td>
<td>Destroy a local key</td>
</tr>
<tr>
<td>GET</td>
<td>/encryption/local/keys/&lt;key&gt;/dependents</td>
<td>List the shares dependent on this key</td>
</tr>
</tbody>
</table>
List a LOCAL Key

Output:

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

List All OKM Keys

Output:

```json
{
  "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>
<thead>
<tr>
<th>Request</th>
<th>Path /api/system/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/version</td>
<td>List the appliance hardware and software version information</td>
</tr>
<tr>
<td>PUT</td>
<td>/diagreboot</td>
<td>Reboot the appliance, gathering additional diagnostic information in the process</td>
</tr>
<tr>
<td>PUT</td>
<td>/reboot</td>
<td>Reboot the appliance</td>
</tr>
<tr>
<td>PUT</td>
<td>/poweroff</td>
<td>Turn off the appliance</td>
</tr>
<tr>
<td>PUT</td>
<td>/restart</td>
<td>Restart the management interface and gather diagnostic information</td>
</tr>
<tr>
<td>PUT</td>
<td>/factoryreset</td>
<td>Reset the appliance configuration back to factory settings</td>
</tr>
<tr>
<td>GET</td>
<td>/disks</td>
<td>List all system disks</td>
</tr>
<tr>
<td>GET</td>
<td>/disks/&lt;disk&gt;</td>
<td>List the specified system disk properties</td>
</tr>
<tr>
<td>GET</td>
<td>/memory</td>
<td>System memory status report</td>
</tr>
</tbody>
</table>

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:
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.

Example Request:

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

Restart System Management

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

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.

Example Request:

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

Factory Reset

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, a query parameter "confirm=true" must be set or the command fails.
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 98 Support Bundle Commands**

<table>
<thead>
<tr>
<th>Request</th>
<th>Path /api/system/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/bundles</td>
<td>List all support bundles</td>
</tr>
<tr>
<td>GET</td>
<td>/bundles/&lt;bundle&gt;</td>
<td>Get the specified bundle data or properties</td>
</tr>
<tr>
<td>POST</td>
<td>/bundles</td>
<td>Build a support bundle and upload it to Oracle Support.</td>
</tr>
<tr>
<td>PUT</td>
<td>/bundles/&lt;bundle&gt;/retry</td>
<td>Retry upload of the specified bundle</td>
</tr>
<tr>
<td>PUT</td>
<td>/bundles/&lt;bundle&gt;/cancel</td>
<td>Cancel upload of the specified bundle</td>
</tr>
<tr>
<td>DELETE</td>
<td>/bundles/&lt;bundle&gt;</td>
<td>Destroy the specified bundle</td>
</tr>
</tbody>
</table>

Create Support Bundle

Creates a new support bundle to help resolve a service request. A Service Request Number (SRN) must be supplied to associate the support bundle with the open service request. The SRN 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 abhadbfshWE=
Host: zfssa.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 23

{"srn": "3-0123456789"}

Example Response:

HTTP/1.1 201 Created
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: zfssa.example.com:215
Accept: */*

Example Result:

```json
{
    "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"
        }
    ],
    "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: zfssa.example.com:215
Accept: */*

Example Result:

HTTP/1.1 200 OK
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 165

```json

```
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 abhadbsfMWE=
Host: zfssa.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 abhadbsfMWE=
Host: zfssa.example.com:215
```

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.

Example Request:

GET /api/system/v1/bundles/9aef7c38-073c-603f-f35c-be64e26e90e3 HTTP/1.1  
Authorization: Basic abhadbsMWE=  
Host: zfssa.example.com:215  
Accept: application/octet-stream

Example Result:

HTTP/1.1 200 OK  
Last-Modified: Mon, 16 Sep 2013 02:59:21 GMT  
Content-Length: 112726389  
Content-Type: application/x-tar  
Accept-Ranges: bytes  
Content-Encoding: gzip

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 abhadbsMWE=  
Host: zfssa.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>
<thead>
<tr>
<th>Request</th>
<th>Path /api/system/v1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/updates</td>
<td>List all system updates</td>
</tr>
<tr>
<td>GET</td>
<td>/updates/&lt;update&gt;</td>
<td>Get the specified system update properties</td>
</tr>
</tbody>
</table>
List System Updates

**Request**

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/api/system/v1/updates-firmware</td>
<td>List the remaining components to be upgraded, along with the time of the last attempt, and their current status</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/system/v1/updates/&lt;update&gt;</td>
<td>Modify update settings</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/system/v1/updates/&lt;update&gt;/upgrade</td>
<td>Upgrade to the specified update image</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/system/v1/updates/&lt;update&gt;/check</td>
<td>Run upgrade health checks for the specified update image</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/system/v1/updates/&lt;update&gt;/rollback</td>
<td>Rollback to the specified update image</td>
</tr>
<tr>
<td>PUT</td>
<td>/api/system/v1/updates-apply</td>
<td>Apply deferred incompatible updates</td>
</tr>
<tr>
<td>DELETE</td>
<td>/api/system/v1/updates/&lt;update&gt;</td>
<td>Destroy the specified system update</td>
</tr>
<tr>
<td>POST</td>
<td>/api/system/v1/updates</td>
<td>Load an update image onto the appliance</td>
</tr>
</tbody>
</table>

**TABLE 100** System Update Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>String</td>
<td>Update media version</td>
</tr>
<tr>
<td>date</td>
<td>DateTime</td>
<td>Update release date</td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>Update media status (immutable)</td>
</tr>
<tr>
<td>update_deferred</td>
<td>ChooseOne</td>
<td>Deferred setting [&quot;onreboot&quot;, &quot;onrequest&quot;]</td>
</tr>
</tbody>
</table>

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: zfssa.example.com:215
Accept: application/json

Example Response:

HTTP/1.1 200 OK
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: zfssa.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": {  
        "date": "Sat Aug 24 2013 17:54:23 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"
    }
}
Upload System Update

This command uploads a new system update image.

Example Upload Command Using curl:

```sh
curl --user root:letmein -k --data-binary @nas@2013.08.24,1-0.pkg.gz \
    --header "Content-Type: application/octet-stream" \n    https://zfssa.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": {
        "date": "Sat Aug 24 2013 17:54:23 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:

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

Example Response:
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: zfssa.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: zfssa.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>
<thead>
<tr>
<th>TABLE 101 User Service Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>DELETE</td>
</tr>
<tr>
<td>POST</td>
</tr>
<tr>
<td>PUT</td>
</tr>
<tr>
<td>PUT</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>POST</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td>PUT</td>
</tr>
<tr>
<td>DELETE</td>
</tr>
<tr>
<td>POST</td>
</tr>
<tr>
<td>GET</td>
</tr>
</tbody>
</table>
List Users

Each user has the following summary properties available.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>logname</td>
<td>Username (immutable after creation)</td>
</tr>
<tr>
<td>string</td>
<td>fullname</td>
<td>Full Name</td>
</tr>
<tr>
<td>string</td>
<td>initial_password</td>
<td>Password</td>
</tr>
<tr>
<td>boolean</td>
<td>require_annotation</td>
<td>Flag to require session annotation</td>
</tr>
<tr>
<td>string</td>
<td>roles</td>
<td>This user’s roles</td>
</tr>
<tr>
<td>boolean</td>
<td>kiosk_mode</td>
<td>Kiosk user</td>
</tr>
<tr>
<td>string</td>
<td>kiosk_screen</td>
<td>Kiosk screen</td>
</tr>
</tbody>
</table>

Example Request:

GET /api/user/v1/users HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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: 394

{  "users": [{
    "fullname": "Joe Admin",
    "href": "/api/user/v1/users/joe",
    ...
  }, {
    "fullname": "Super-User",
    "href": "/api/user/v1/users/root",
    ...
  }]}
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 103  User Preferences**

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>locale</td>
<td>Locality</td>
</tr>
<tr>
<td>string</td>
<td>login_screen</td>
<td>Initial login screen</td>
</tr>
<tr>
<td>string</td>
<td>session_timeout</td>
<td>Session timeout in minutes</td>
</tr>
<tr>
<td>string</td>
<td>advanced_analytics</td>
<td>Make available advanced analytics statistics</td>
</tr>
</tbody>
</table>

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

**TABLE 104  SSH Key Properties**

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>type</td>
<td>The type of SSH key: either RSA or DSA</td>
</tr>
<tr>
<td>string</td>
<td>key</td>
<td>The contents of the SSH key</td>
</tr>
<tr>
<td>string</td>
<td>comment</td>
<td>A comment associated with this SSH key</td>
</tr>
</tbody>
</table>

Example Request:

```
GET /api/user/v1/users/joe HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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
```

```json
{
  "user": {
    "fullname": "Joe Admin",
    "href": "/api/user/v1/users/joe",
```
"initial_password": "DummyPassword",
"kiosk_mode": false,
"kiosk_screen": "status/dashboard",
"logname": "joe",
"require_annotation": false,
"roles": ["basic"]
}
}

Create User

This command uses three forms:
- **Create a new local user** - Creates a new local user
- **Clone an existing user** - Clones a new user from an existing user
- **Add an administrator** - Requires the `netuser` property set with the name of the network user

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

Create a new local user has the following properties:

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>logname</td>
<td>New user's login name (required)</td>
</tr>
<tr>
<td>string</td>
<td>fullname</td>
<td>New user's full name (required)</td>
</tr>
<tr>
<td>string</td>
<td>initial_password</td>
<td>Initial user password (required)</td>
</tr>
<tr>
<td>boolean</td>
<td>require_annotation</td>
<td>Optional flag to require session annotation</td>
</tr>
</tbody>
</table>

Clone an existing user has the following required properties:

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>user</td>
<td>Source user name</td>
</tr>
<tr>
<td>string</td>
<td>clonename</td>
<td>New clones login name</td>
</tr>
<tr>
<td>string</td>
<td>fullname</td>
<td>New clone user's full name (local only)</td>
</tr>
<tr>
<td>boolean</td>
<td>password</td>
<td>New clone user password (local only)</td>
</tr>
</tbody>
</table>

Add an administrator has the following properties:
TABLE 107 Net User Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>netuser</td>
<td>Net user login name</td>
</tr>
</tbody>
</table>

Example Request (to create a local user):

POST /api/user/v1/users HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.example.com:215
Accept: application/json
Content-Type: application/json
Content-Length: 71

{"logname":"joe", "fullname":"Joe Admin", "initial_password":"letmein"}

Example Result:

HTTP/1.1 201 Created
X-Zfssa-Appliance-Api: 1.0
Content-Type: application/json
Content-Length: 386
Location: /api/user/v1/users/joe

```
{
    "user": {
        "href": "/api/user/v1/users",
        "logname": "joe",
        "fullname": "Joe Admin",
        "initial_password": "DummyPassword",
        "require_annotation": false,
        "kiosk_mode": false,
        "kiosk_screen": "status/dashboard",
        "roles": ["basic"],
        "exceptions": {},
        "preferences": {
            "locale": "C",
            "login_screen": "status/dashboard",
            "session_timeout": 15,
            "advanced_analytics": false,
            "keys": {}
        }
    }
}
```

Modify Users

Modifies user properties directly. User resources: exceptions, preferences, and ssh keys can be added, modified or removed.
Delete Users

Example Request:

PUT /api/user/v1/users/joe HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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/joe",
        "logname": "joe",
        "fullname": "Joe Admin",
        "initial_password": "DummyPassword",
        "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/joe HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.example.com:215
Accept: */*

Example Result:

HTTP/1.1 204 No Content
X-Zfssa-Appliance-Api: 1.0
Workflow 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 Service Commands

The following table shows the workflow service commands.

<table>
<thead>
<tr>
<th>TABLE 108</th>
<th>Workflow Service Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Path /api/workflow/v1</td>
</tr>
<tr>
<td>GET</td>
<td>/api/workflow/v1</td>
</tr>
<tr>
<td>GET</td>
<td>/workflows</td>
</tr>
<tr>
<td>GET</td>
<td>/workflows/&lt;workflow&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/workflows/&lt;workflow&gt;</td>
</tr>
<tr>
<td>PUT</td>
<td>/workflows/&lt;workflow&gt;/execute</td>
</tr>
<tr>
<td>DELETE</td>
<td>/workflows/&lt;workflow&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>/workflows</td>
</tr>
</tbody>
</table>

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: zfssa.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-a658-f6fb19f1172c",
      ...
    },
    {
      "description": "Removes the artifacts from the appliance used by Oracle Enterprise
Manager",
      "href": "/api/workflow/v1/workflows/bd7214fc-6bba-c7ad-ed1f-942c0189e757",
      ...
    }
  ]
}
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: zfssa.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."
}  }
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/execute HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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"
}

Delete Workflow

Deletes a workflow script from the appliance.

Example Request:

DELETE /api/workflow/v1/workflows/f4fe892f-cf46-4d6a-9026-cd0c8ccee9971 HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.example.com:215
Upload Workflow

Uploads a workflow to the appliance.

Example Request:

```
POST /api/workflow/v1/workflows HTTP/1.1
Authorization: Basic abcefgMWE=
Host: zfssa.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: jkremer/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
}
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. This section contains more detailed information about various RESTful clients.

Curl Rest Client

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://zfssa.example.com:215/api/nas/v1/pools/gold
```

HTTP/1.1 200 OK
Date: Tue, 23 Jul 2013 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

```json
{
  "pool": {
    "profile": "mirror",
    "name": "gold",
    "usage": {
      "available": 895468984832.0,
      "total": 895500681216.0,
      "dedupratio": 100,
      "used": 31696384.0
    },
    "peer": "00000000-0000-0000-0000-000000000000",
  }
}
```
Create a New Resource

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

```bash
$ curl --user ${USER}:${PASSWORD} -s -k -i -X POST -d @-  
-H "Content-Type: application/json" 
https://zfssa-host.example.com:215/api/user/v1/users <<JSON
> {"logname": "rest_user", 
> "fullname": "REST User", 
> "initial_password": "letmein"}
> JSON

HTTP/1.1 201 Created
Date: Tue, 23 Jul 2013 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": "DummyPassword",
    "require_annotation": false,
    "kiosk_mode": false,
    "kiosk_screen": "status/dashboard",
    "roles": ["basic"],
    "exceptions": {},
    "preferences": {
      "href": "/api/user/v1/users/larry/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:
Delete an Existing Resource

This command removes a user from the system:

```
curl --user ${USER}:${PASSWORD} -s -k -i -X DELETE https://tanana:215/api/appliance/v1/users/jschwartz
```

HTTP/1.1 204 No Content
Date: Tue, 23 Jul 2013 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:

```python
>>> 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", "letmein")
>>> 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:

>>> response = opener.open(request)
>>> response.getcode()
200
>>> body = json.loads(response.read())
>>> print json.dumps(body, sort_keys=True, indent=4)
{
  "routes": [
    {
      "destination": "0.0.0.0",
      "family": "IPv4",
      "gateway": "10.80.231.1",
      "href": "/api/network/v1/routes/ixgbe0,0.0.0.0,10.80.231.1",
      "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)
Python RESTful Client

>>> 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'  
                                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": true,
            "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:

```python
>>> request = urllib2.Request('https://zfssa:215/api/alert/v1/actions/actions-001')
>>> request.get_method = lambda: 'DELETE'
>>> response = opener.open(request)
>>> response.getcode()
204
```