CONTENTS

Preface

Chapter 1 Overview
About the System Management API ............................................................. 12
System management interfaces .................................................................. 12
Authentication ............................................................................................ 12
Error responses ........................................................................................... 13
Resource hierarchy ...................................................................................... 15

Part 1 REST API Reference

Chapter 2 REST API: Retrieving Network Details
Get node information for an RMG ............................................................ 20
Get all networks for a node ....................................................................... 21
Get details for a specific network on a node ............................................. 23

Chapter 3 REST API: Retrieving RMG and Node Details
Get node info ............................................................................................ 26
Get nodes list ............................................................................................ 27
Get RMG info ........................................................................................... 29
Get RMG list ............................................................................................. 30

Chapter 4 REST API: Managing Subtenants
Abort subtenant deletion ......................................................................... 34
Create subtenant ....................................................................................... 36
Delete subtenant ....................................................................................... 38
Get subtenant details ............................................................................... 39
List subtenants ........................................................................................ 41
Remove subtenant .................................................................................... 42
Rename subtenants .................................................................................... 43

Chapter 5 REST API: Managing UIDs
Abort UID deletion ................................................................................... 48
Create UID and shared secret ................................................................. 50
Delete UID and shared secret .................................................................. 53
Modify the UID or shared secret ............................................................... 55
Get shared secret ....................................................................................... 56
List UIDs .................................................................................................. 59
Remove UID ............................................................................................. 60

Chapter 6 REST API: Managing Location Groups
Create a location group ............................................................................. 64
Delete a location group ............................................................................ 66
Get location group details ......................................................................... 67
Chapter 7  REST API: Managing Policy Selectors
Assign a policy selector ................................................................. 74
Create a policy selector ............................................................... 75
Delete a policy selector ............................................................... 78
Get policy selector ..................................................................... 79
List policy selectors .................................................................. 80
Update a policy selector ............................................................. 81

Chapter 8  REST API: Managing Policy Specifications
Create or update a policy specification ......................................... 86
Delete a policy specification ....................................................... 96
Get details for a policy specification .......................................... 97
List policy specifications ........................................................... 98
Example: How to create a policy specification ......................... 100

Chapter 9  REST API: System Monitoring
Web service statistics .................................................................. 104
Atmos storage consumption ..................................................... 105

Part 2  POX API Reference

Chapter 10  POX API: Authenticating Administrators
Authenticating as a SysAdmin .................................................... 112
Authenticating as a TenantAdmin or a SubtenantAdmin ............. 114

Chapter 11  POX API: Managing Tenants
Assign a TenantAdmin ............................................................... 118
Create a tenant ........................................................................... 119
Get tenant details ...................................................................... 120
List tenants ................................................................................ 123
Remove a TenantAdmin ........................................................... 125
Examples: Obtaining tenant information .................................. 126

Chapter 12  POX API: Managing Subtenants
Assign a SubtenantAdmin .......................................................... 132
Create a subtenant .................................................................... 133
Read subtenant.......................................................................... 134
List subtenants .......................................................................... 136
Modify a subtenant .................................................................... 138
Remove a SubtenantAdmin ....................................................... 139
Example: How to obtain subtenant information ....................... 140

Chapter 13  POX API: Managing NFS Shares
Assign NFS access node to a tenant .......................................... 144
Create NFS share ....................................................................... 145
Contents

Chapter 14  POX API: Listing File Systems
Listing file systems ................................................................. 158
Translate node name to UUID .................................................. 159

Chapter 15  POX API: Managing UIDs and Shared Secrets
Create UID .................................................................................. 162
Disable UID ............................................................................... 163
Enable UID .................................................................................. 164
Get UID .................................................................................... 166
List UIDs ................................................................................... 167
Reset shared secrets ................................................................. 168

Chapter 16  POX API: Managing Policy Selectors
Assign a policy selector (full replacement) ................................... 172
Assign a policy selector (incremental add) .................................. 173
Create a policy selector ............................................................. 175
Delete a policy selector (delete all) ............................................. 178
Delete a policy selector (incremental delete) .............................. 179
Get policy selector details ....................................................... 181
List policy selectors ................................................................. 182
List policy selectors per subtenant or list subtenants per policy selector ... 184
Modify a policy selector ........................................................... 185
Remove a policy selector .......................................................... 188

Chapter 17  POX API: Managing Policy Specifications
Create or update a policy specification ....................................... 192
Delete a policy specification ..................................................... 202
Get a policy specification .......................................................... 203
List policy specifications ........................................................... 205
Examples: Create a policy specification ................................... 207

Index
PREFACE

As part of an effort to improve its product lines, EMC periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Contact your EMC representative if a product does not function properly or does not function as described in this document.

Note: This document was accurate at publication time. New versions of this document might be released on the EMC online support website. Check the EMC online support website to ensure that you are using the latest version of this document.

Audience

This document is part of the Atmos documentation set, and is intended for use by developers who want to programmatically configure and maintain Atmos.

Related documentation

The EMC Atmos documentation set includes the following titles:

- EMC Atmos Release Notes
- EMC Atmos Administrator’s Guide
- EMC Atmos Programmer’s Guide
- EMC Atmos System Management API Guide
- EMC Atmos Security Configuration Guide
- EMC Atmos CAS Programmer’s Guide
- EMC Atmos CAS API Reference Guide
- EMC Atmos Installable File System (IFS) Installation and Upgrade Guide
- EMC Atmos online help
- EMC Atmos Series Open Source License and Copyright Information
- EMC Atmos Series Open Source License and Copyright Information for GPLv3

Conventions used in this document

EMC uses the following conventions for special notices:

Note: A note presents information that is important, but not hazard-related.

IMPORTANT
An important notice contains information essential to software or hardware operation.
Preface

Typographical conventions

EMC uses the following type style conventions in this document:

**Normal** Used in running (nonprocedural) text for:
- Names of interface elements, such as names of windows, dialog boxes, buttons, fields, and menus
- Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, functions, and utilities
- URLs, pathnames, filenames, directory names, computer names, links, groups, service keys, file systems, and notifications

**Bold** Used in running (nonprocedural) text for names of commands, daemons, options, programs, processes, services, applications, utilities, kernels, notifications, system calls, and man pages

Used in procedures for:
- Names of interface elements, such as names of windows, dialog boxes, buttons, fields, and menus
- What the user specifically selects, clicks, presses, or types

**Italic** Used in all text (including procedures) for:
- Full titles of publications referenced in text
- Emphasis, for example, a new term
- Variables

**Courier** Used for:
- System output, such as an error message or script
- URLs, complete paths, filenames, prompts, and syntax when shown outside of running text

**Courier bold** Used for specific user input, such as commands

**Courier italic** Used in procedures for:
- Variables on the command line
- User input variables

< > Angle brackets enclose parameter or variable values supplied by the user

[] Square brackets enclose optional values

| Vertical bar indicates alternate selections — the bar means “or”

{} Braces enclose content that the user must specify, such as x or y or z

... Ellipses indicate nonessential information omitted from the example

Where to get help

EMC support, product, and licensing information can be obtained as follows:

**Product information** - For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

https://support.emc.com

**Technical support** - Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.
Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to:

techpubcomments@emc.com
Preface
This chapter introduces the Atmos System Management API.

- **About the System Management API** ................................................................. 12
- **System management interfaces** ..................................................................... 12
- **Authentication** ............................................................................................... 12
- **Error responses** .............................................................................................. 13
- **POX error handling** ....................................................................................... 14

**Note:** All administrative update operations are prohibited during upgrade except for REST calls to create subtenant and create UID.
Overview

About the System Management API

Atmos provides a system-management API that lets you implement a subset of system management functions in client applications. It includes operations that let you:

- Retrieve RMG and node information.
- Create and manage tenants and subtenants.
- Create and manage UIDs.
- Create and manage policies.
- Retrieve web service statistics.
- View storage consumption.
- Define CIFS or NFS access nodes, and create and manage CIFS and NFS shares.

In addition to the system management API, Atmos includes:

- A storage API for creating, reading, updating, and deleting objects. For more information, see the EMC Atmos Programmer’s Guide.
- An API for storing and accessing fixed content (information in its final form). For more information, see the EMC Atmos CAS Programmer’s Guide and the EMC Atmos CAS API Reference Guide.

System management interfaces

The Atmos system management API is exposed through two interfaces:

- REST — A robust, standards-based API. This is the recommended interface, and is intended to replace the POX-based API over time. Using the REST interface has the following advantages:
  - It is standards-based.
  - It provides well-defined resources.
  - It includes enhanced error checking, and always returns an error message.
- POX (Plain Old XML) — An API that returns XML. This interface was developed for internal use to implement the system management GUI.

Note: The use of the POX-based API for external client applications is being phased out — use the REST API when possible.

It is a best practice to use HTTPS when performing system management functions.

Authentication

All system management operations require that the user making the request be authenticated, and each operation requires that the authenticating user be in a specific role. The REST and POX APIs use different authentication models.
REST Authentication

To authenticate via the REST API you must include a set of authentication headers with each request. The authentication headers correspond to the administrative roles.

**Table 1  Authentication Headers for Administrative Roles**

<table>
<thead>
<tr>
<th>Administrative Role</th>
<th>Authentication Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysAdmin</td>
<td>x-atmos-systemadmin: Specify the SysAdmin username. (Required.)</td>
</tr>
<tr>
<td></td>
<td>x-atmos-systemadminpassword: Specify the SysAdmin password. (Required.)</td>
</tr>
<tr>
<td></td>
<td>x-atmos-authtype: Specify password.</td>
</tr>
<tr>
<td>TenantAdmin</td>
<td>x-atmos-tenantadmin: Specify a valid username of a user in the TenantAdmin role.</td>
</tr>
<tr>
<td></td>
<td>x-atmos-tenantadminpassword: Specify the password for the x-atmos-tenantadmin.</td>
</tr>
<tr>
<td></td>
<td>x-atmos-authtype: Specify password.</td>
</tr>
<tr>
<td>SubtenantAdmin</td>
<td>x-atmos-subtenantadmin: Specify a valid username of a user in the SubtenantAdmin role.</td>
</tr>
<tr>
<td></td>
<td>x-atmos-subtenantadminpassword: Specify the password for the x-atmos-subtenantadmin.</td>
</tr>
<tr>
<td></td>
<td>x-atmos-authtype: Specify password.</td>
</tr>
</tbody>
</table>

Each request is a session. Sessions do not span requests.

POX Authentication

The POX API authentication model follows the typical GUI pattern where the user logs in and performs a set of actions during a session.

To authenticate via the POX API, use the login method appropriate to the role (SysAdmin, TenantAdmin, or SubtenantAdmin) for the management actions you want to perform. The server returns a session ID (gui_session_id) in the HTTP Set-Cookie response header. For each subsequent request, you must pass in the session ID in the HTTP Cookie request header.

The API does not include a logout operation. An inactive session expires after 30 minutes.

Error responses

When the server cannot fulfill a request, it generates and returns an error. The POX and REST API perform error handling in different ways.

EMC recommends that you follow the standard programming practice of checking the return codes when programming with the Atmos API. Always check the return status of the API calls and handle any errors appropriately. This is especially crucial for operations that change the state of the data, such as write and append. If the write operation fails with an error, it should be re-tried. A failed write operation can result in inconsistent data.
Overview

REST error handling

All REST requests return a status code in the header. For HTTP status codes that indicate failure, the response body includes an error message. For more information about HTTP status codes that indicate an error condition, see Table 2, “HTTP status codes returned for errors.”

Failed response header

HTTP/1.1 #{statusCode}
Date: date
Content-Type: type
Content-Length: length
Connection: close
Server: mongrel
x-atmos-sysmgmt-version: #{version}
<?xml version='1.0' encoding='UTF-8'?>

Failed response body

<errorResponse>
<errorCode>#{errorCode}</errorCode>
<errorMessage>#{error message}</errorMessage>
</errorResponse>

HTTP status codes

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad request.</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized. The client did not provide the correct tenant Admin name and password.</td>
</tr>
<tr>
<td>404</td>
<td>Name not found.</td>
</tr>
<tr>
<td>500</td>
<td>Server internal error. The error message in the response body provides the details of the error.</td>
</tr>
<tr>
<td>503</td>
<td>Service unavailable.</td>
</tr>
</tbody>
</table>

POX error handling

The POX API returns an error message when a request fails. The structure of the message varies depending on the operation. For example, you might receive a message like the following:

<cleared>Error: No such user.</cleared>

or a message like this:

<deleted>Delete Node CIFS entry error.</deleted>

See the description of the operation for more information about error responses.
Resource hierarchy

The system management API allows you to perform create, read, update, delete, and list operations. It uses the standard HTTP methods to perform these operations as described in Table 3.

Table 3 HTTP Methods

<table>
<thead>
<tr>
<th>For this operation</th>
<th>Use this HTTP method...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>POST</td>
</tr>
<tr>
<td>Read/List</td>
<td>GET</td>
</tr>
<tr>
<td>Update</td>
<td>PUT</td>
</tr>
<tr>
<td>Delete</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

For every type of resource (RMG, tenant, policy, and so on), you can explore the set of objects that are available by using the List operation. Once you have the list of existing objects, you can use the resource’s name to obtain further details about it, to update or delete it.

Note: In POST operations, you can send URL-encoded data and data that is not URL-encoded. Because the API does not enforce one format or the other, you must be consistent within your own application environment.

URI pattern

Atmos uses a URI pattern to determine whether to invoke the POX or the REST API.

REST API resources use this URL pattern:

/sysmgmt/<path-to-resource>

POX API resources use a roles-based URL pattern, for example:

/maui_admin/<path-to-resource>
/maui_tenant/<path-to-resource>
/maui_sub_tenant/<path-to-resource>

Resource access

Atmos requires that system management resources be requested using HTTPS on port 443. For example:

https://<ipaddress>:443/sysmgmt/rmgs

The web service handles system management resource requests sequentially and serially. It supports, at most, two concurrent management operations. You will see degraded performance in both the Atmos system management GUI and your client programs when more than two concurrent requests are made.

Because the system management API works with resources that can be widely distributed, the results of a successful operation might take time to propagate through the system. Make sure your application design anticipates this delay. For example, you might write an
application that manages UIDs. After you delete a UID, you might expect any requests to retrieve that UID would fail immediately after you delete it. However, in some cases it might take as long as 20 seconds for the delete to complete.
PART 1
REST API Reference

This section includes the REST API reference.

- “REST API: Retrieving Network Details” on page 19
  Describes the REST API for viewing network details for RMGs and nodes.

- “REST API: Retrieving RMG and Node Details” on page 25
  Describes the REST API for retrieving RMGs and nodes.

- “REST API: Managing Subtenants” on page 33
  Describes the REST API for managing subtenants.

- “REST API: Managing UIDs” on page 47
  Describes the REST API for managing UIDs.

- “REST API: Managing Location Groups” on page 63
  Describes the REST API for managing location groups.

- “REST API: Managing Policy Selectors” on page 73
  Describes the REST API for managing policy selectors.

- “REST API: Managing Policy Specifications” on page 85
  Describes the REST API for managing policy specifications.

- “REST API: System Monitoring” on page 103
  Describes the REST API for retrieving storage consumption and statistics.
CHAPTER 2
REST API: Retrieving Network Details

This chapter describes the REST APIs for viewing details of the Atmos network. It includes the following sections:

- Get node information for an RMG ................................................................. 20
- Get all networks for a node .............................................................................. 21
- Get details for a specific network on a node .................................................. 23
Get node information for an RMG

Returns an XML document listing the following details for each node in the specified RMG.

- State (up or down).
- IP address of the management network.
- Node name.
- Node location.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs/<rmg-name>/nodes

Request Parameters

- `x-atmos-systemadmin`: Specify the SysAdmin username. (Required.)
- `x-atmos-systemadminpassword`: Specify the SysAdmin password. (Required.)
- `x-atmos-authType`: Specify password.

Request header

GET /sysmgmt/rmgs/London/nodes HTTP/1.1
accept: */*
Date: Thu, 01 Nov 2012 14:51:23 GMT
x-atmos-systemadmin:sysadmin
x-atmos-systemadminpassword:password
x-atmos-authType:password

Request body

None

Response header

HTTP/1.1 200 OK
Date: Thu, 01 Nov 2012 14:51:23 GMT
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
ETag: "c1bd133d543f2aa19748d91ad36d8230"
Cache-Control: private, max-age=0, must-revalidate
Status: 200
X-Runtime: 8748
x-atmos-sysmgmt-version: 1.0.0
Content-Length: 426
Set-Cookie: _gui_session_id=BAh7DDoUbG9naW5fcsGFzcl; path=/; HttpOnly
Connection: close
Get all networks for a node

Returns an XML document listing the IP address, netmask, interface, and VLAN configured on the specified node for each of the following networks:

- IPMI
- internal
- bmc
- data
- DHCP
- management

This operation does not return the access IP addresses that may be assigned to the node.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs/<rmg-name>/nodes/<node-name>/network

Request parameters

- x-atmos-systemadmin: Specify the SysAdmin username. (Required.)
- x-atmos-systemadminpassword: Specify the SysAdmin password. (Required.)
- x-atmos-authtype: Specify password.
REST API: Retrieving Network Details

Request header

GET /sysmgmt/rmgs/London/nodes/London-is1-003/network HTTP/1.1
Host: 10.5.116.244
x-atmos-systemadmin:sysadmin
x-atmos-systemadminpassword:password
x-atmos-authType:password

Request body

None

Response header

HTTP/1.1 200 OK
Date: Thu, 01 Nov 2012 15:14:53 GMT
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
ETag: "67ea362d62901adeead30cb9160b8a8"
Cache-Control: private, max-age=0, must-revalidate
Status: 200
X-Runtime: 415
x-atmos-sysmgmt-version: 1.0.0
Content-Length: 904
Set-Cookie: _gui_session_id=BAh7DDoUbG9naW5fcGFzc193 path=/; HttpOnly
Connection: close

Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<networks>
    <network name="ipmi">
        <ip>192.168.13.13</ip>
        <netmask>255.255.255.0</netmask>
        <intf>eth0</intf>
        <vlan></vlan>
    </network>
    <network name="int">
        <ip>192.168.11.13</ip>
        <netmask>255.255.255.0</netmask>
        <intf>eth0</intf>
        <vlan></vlan>
    </network>
    <network name="bmc">
        <ip>192.168.13.141</ip>
        <netmask>255.255.255.0</netmask>
        <intf>bmc</intf>
        <vlan></vlan>
    </network>
    <network name="data">
        <ip>10.5.116.184</ip>
        <netmask>255.255.255.0</netmask>
        <intf>eth1</intf>
        <vlan></vlan>
    </network>
    <network name="dhcp">
        <ip>192.168.12.13</ip>
        <netmask>255.255.255.0</netmask>
        <intf>eth0</intf>
        <vlan></vlan>
    </network>
    <network name="mgmt">
        <ip>10.5.116.184</ip>
```
Get details for a specific network on a node

Returns an XML document with the following information for each network that can be configured for the specified node:

- IP address
- netmask
- interface
- VLAN

The network-name parameter can be:

- mgmt
- data
- ipmi
- dhcp
- bmc
- int

You cannot use this operation to obtain information about the access network.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs/<rmg-name>/nodes/<node-name>/network/<network-name>

Request parameters

- \texttt{x-atmos-systemadmin}: Specify the SysAdmin username. (Required.)
- \texttt{x-atmos-systemadminpassword}: Specify the SysAdmin password. (Required.)
- \texttt{x-atmos-authtype}: Specify password.

Request header

GET /sysmgmt/rmgs/London/nodes/London-is1-003/network/data HTTP/1.1
Host: 10.5.116.244
\texttt{x-atmos-systemadmin:sysadmin}
REST API: Retrieving Network Details

x-atmos-systemadminpassword: password
x-atmos-authType: password

Request body
None

Response header
HTTP/1.1 200 OK
Date: Thu, 01 Nov 2012 15:24:23 GMT
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
ETag: "99ecc2cbbf1f1f45578eba2425e4aaf8d"
Cache-Control: private, max-age=0, must-revalidate
Status: 200
X-Runtime: 280
x-atmos-sysmgmt-version: 1.0.0
Content-Length: 161
Set-Cookie: gui_session_id=BAh7DDoUbG9naW5fcGFzc193b; path=/; HttpOnly
Connection: close

Response body

<?xml version="1.0" encoding="UTF-8"?>
<network name="data">
  <ip>10.5.116.184</ip>
  <mask>255.255.255.0</mask>
  <intf>eth1</intf>
  <vlan></vlan>
</network>
CHAPTER 3
REST API: Retrieving RMG and Node Details

This chapter describes the REST API for viewing RMGs and nodes.

- Get node info .......................................................................................................... 26
- Get nodes list.......................................................................................................... 27
- Get RMG info........................................................................................................... 29
- Get RMG list ............................................................................................................ 30
REST API: Retrieving RMG and Node Details

Get node info

Returns an XML document that provides details about the specified node. The details include both configuration (such as machine type, atmos version, and number of CPUs) as well as metrics for the node (such as number of processes, swap memory free and used).

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs/<rmgName>/nodes/<nodeName>

Request parameters

- x-atmos-systemadmin: Specify the SysAdmin username. (Required.)
- x-atmos-systemadminpassword: Specify the SysAdmin password. (Required.)
- x-atmos-authtype: Specify password.

Request header

GET /sysmgmt/rmgs/Boston01/nodes/IS01-001 HTTP/1.1
accept: */*
date: Thu, 12 Nov 2009 10:24:07 GMT
x-atmos-systemadminpassword: password
x-atmos-systemadmin: SysAdmin1
x-atmos-authtype: password

Successful Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 12 Nov 2009 10:24:12 GMT
Set-Cookie: _gui_session_id=d29496c84a915be48bafed9e051ae17; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 638
Successful Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<node name="IS01-001" up="true" location="BostonFl1">
  <machineType>x86_64</machineType>
  <operationSystem>Linux</operationSystem>
  <atmosVersion>1.2.5</atmosVersion>
  <numberOfcpu>1</numberOfcpu>
  <cpuSpeed>1995</cpuSpeed>
  <totalMemory>1027372.000</totalMemory>
  <freeMemory>38784.000</freeMemory>
  <totalDiskSpace>6.231</totalDiskSpace>
  <numberOfprocesses>531</numberOfprocesses>
  <cpuSystemUsage>19.9</cpuSystemUsage>
  <cpuUserUsage>15.9</cpuUserUsage>
  <cpuIdle>64.2</cpuIdle>
  <totalSwapMemory>2104504.000</totalSwapMemory>
  <freeSwapMemory>2103008.000</freeSwapMemory>
</node>
```

Table 4  Get Node Info Return Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cpuSpeed&gt;</td>
<td>Integer in MHz</td>
</tr>
<tr>
<td>&lt;totalMemory</td>
<td>Decimal in Bytes</td>
</tr>
<tr>
<td>&lt;freeMemory&gt;</td>
<td>Bytes</td>
</tr>
<tr>
<td>&lt;totalDiskSpace&gt;</td>
<td>Decimal in GB</td>
</tr>
<tr>
<td>&lt;cpuSystemUsage&gt;</td>
<td>Percentage</td>
</tr>
<tr>
<td>&lt;cpuUserUsage&gt;</td>
<td>Percentage</td>
</tr>
<tr>
<td>&lt;cpuIdle&gt;</td>
<td>Percentage</td>
</tr>
<tr>
<td>&lt;totalSwapMemory&gt;</td>
<td>Decimal in Bytes</td>
</tr>
<tr>
<td>&lt;freeSwapMemory&gt;</td>
<td>Decimal in Bytes</td>
</tr>
</tbody>
</table>

Get nodes list

Returns the list of nodes for the specified RMG. For each node, it lists the node location, and whether it is up or down.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs/<rmgName>/nodes
REST API: Retrieving RMG and Node Details

Request parameters

- **x-atmos-systemadmin**: Specify the SysAdmin username. (Required.)
- **x-atmos-systemadminpassword**: Specify the SysAdmin password. (Required.)
- **x-atmos-authtype**: Specify password.

Request header

```plaintext
GET /sysmgmt/rmgs/Boston01/nodes HTTP/1.1
accept: */*
date: Thu, 12 Nov 2009 10:21:56 GMT
x-atmos-systemadminpassword: password
x-atmos-systemadmin: SysAdmin1
x-atmos-authtype: password
```

Request body

None

Successful Response header

```plaintext
HTTP/1.1 200 OK
Connection: close
Date: Thu, 12 Nov 2009 10:22:01 GMT
Set-Cookie: _gui_session_id=63da47f6abe6c4eb21e12f2328baca43; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 326
```

Successful Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<nodeList>
  <node name="IS01-001" up="true" location="BostonFl1"></node>
  <node name="IS01-002" up="true" location="BostonFl1"></node>
  <node name="IS01-003" up="true" location="BostonFl1"></node>
  <node name="IS01-004" up="true" location="BostonFl1"></node>
</nodeList>
```

Table 5  Get Nodes List Return Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>Container element for an individual node.</td>
</tr>
<tr>
<td>name</td>
<td>Shows the node’s name.</td>
</tr>
<tr>
<td>location</td>
<td>Shows the node’s location.</td>
</tr>
<tr>
<td>up</td>
<td>Returns true if the node is running, or false if the node is down.</td>
</tr>
</tbody>
</table>
Get RMG info

Returns information about a specific RMG.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs/<rmgName>

Request parameters

- `x-atmos-systemadmin`: Specify the SysAdmin username. (Required.)
- `x-atmos-systemadminpassword`: Specify the SysAdmin password. (Required.)
- `x-atmos-authtype`: Specify password.

Request header

Header: GET /sysmgmt/rmgs/Boston01 HTTP/1.1
accept: */*
date: Thu, 12 Nov 2009 10:01:29 GMT
x-atmos-systemadminpassword: password
x-atmos-systemadmin: SysAdmin1
x-atmos-authtype: password

Request body

None

Successful Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 12 Nov 2009 10:01:33 GMT
Set-Cookie: _gui_session_id=82c33d236af1dc95ed6b5335dc714ec1; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 287

Successful Response body

A successful response returns the following XML document:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
```
REST API: Retrieving RMG and Node Details

Table 6  Get RMG Info Return Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmg</td>
<td>Container element for individual RMG.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the RMG's name</td>
</tr>
<tr>
<td>localTime</td>
<td>Specifies the local time for the RMG.</td>
</tr>
<tr>
<td>nodesUp/nodesDown</td>
<td>Specifies the number of nodes that are running, and the number that are not running.</td>
</tr>
<tr>
<td>avgLoad15, avgLoad5, avgLoad1</td>
<td>Measures how busy the Atmos services on the RMG are for one, five, and 15 minute intervals.</td>
</tr>
</tbody>
</table>

Get RMG list

Returns an XML document that provides the complete list of RMGs managed by the authenticated user in the SysAdmin role. The XML document provides the following details on each RMG: local time, number of CPUs, number of nodes that are up and the number that are down, and the average load for 1, 5, and 15 minute intervals.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/rmgs

Request parameters

- x-atmos-systemadmin: Specify the SysAdmin username. (Required.)
- x-atmos-systemadminpassword: Specify the SysAdmin password. (Required.)
- x-atmos-authtype: Specify password.

Request header

GET /sysmgmt/rmgs HTTP/1.1
accept: */*
date: Thu, 12 Nov 2009 09:59:54 GMT
x-atmos-systemadminpassword: password
x-atmos-systemadmin: SysAdmin1
x-atmos-authtype: password

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 12 Nov 2009 10:00:00 GMT
Set-Cookie: _gui_session_id=afa8a3848c457bff0f8cdd6ebff2b53; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 328

Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rmgList>
  <rmg name="Boston01">
    <name>Boston01</name>
    <localTime>Thu Nov 12 09:59:59 +0000 2009</localTime>
    <nodesUp>4</nodesUp>
    <nodesDown>0</nodesDown>
    <avgLoad15>110.25</avgLoad15>
    <avgLoad5>111.25</avgLoad5>
    <avgLoad1>104.0</avgLoad1>
  </rmg>
</rmgList>
```

Table 7  Get RMG List Return Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmg</td>
<td>Container element for individual RMG.</td>
</tr>
<tr>
<td>name</td>
<td>Shows the RMG's name.</td>
</tr>
<tr>
<td>localTime</td>
<td>Shows the RMG's local time in UTC format.</td>
</tr>
<tr>
<td>nodesUp/nodesDown</td>
<td>Shows the number of nodes in the RMG that are running (up) and that are not running (down).</td>
</tr>
<tr>
<td>avgLoad15, avgLoad5, avgLoad1</td>
<td>Shows how busy the Atmos services on the RMG are for one, five, and 15 minute intervals.</td>
</tr>
</tbody>
</table>
REST API: Retrieving RMG and Node Details
CHAPTER 4
REST API: Managing Subtenants

This chapter describes the REST API for subtenants.

- Abort subtenant deletion ................................................................. 34
- Create subtenant ....................................................................... 36
- Delete subtenant ....................................................................... 38
- Get subtenant details ................................................................. 39
- List subtenants .......................................................................... 41
- Rename subtenants ................................................................. 42
Abort subtenant deletion

Cancels a running subtenant delete operation. The subtenant must be in the deleting state.

When a subtenant delete request is aborted, the UIDs are re-enabled and I/O can resume on objects that have not yet been deleted. Objects that have already been deleted cannot be recovered.

Required role

TenantAdmin

HTTP method

POST

URI

/sysmgmt/tenants/<tenantName>/subtenants/<subtenantName>/abort_deletion

Request parameters

HTTP header

- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.

Request headers

POST /sysmgmt/tenants/t1/subtenants/t1/abort_deletion HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 12:46:19 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 12:50:36 GMT
Set-Cookie: _gui_session_id=506a22799a2324f9551ef573c53ba467;
          path=/
Status: 200 OK
x-atoms-sysmgmt-version: 1.0.0
X-Runtime: 1217ms
ETag: "d434b983e7d60ef5c0566fb6b91476a3"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 63

Response body

<?xml version="1.0" encoding="UTF-8"?>
<aborted>true</aborted>
Create subtenant

Creates an Atmos subtenant or compliant subtenant within the tenant of the requesting tenant administrator. Returns a system-generated subtenant ID.

Upgrade considerations

You can perform this operation during upgrade from Atmos 2.0 to a higher version. However, this operation requires access to internal resources which might become unavailable (for up to 60 seconds) during the upgrade of the first RMG. If the create subtenant request occurs during this 60 second period, the request might fail with one of the following error messages:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<errorResponse>
  <errorCode>503</errorCode>
  <errorMessag>Service Unavailable</errorMessage>
</errorResponse>
```
or

```xml
<?xml version="1.0" encoding="UTF-8"?>
<errorResponse>
  <errorCode>400</errorCode>
  <errorMessag>Error: Failed to initialize subtenant.</errorMessage>
</errorResponse>
```

Make sure to code your application to handle these errors by either reporting the problem to the end user, or by retrying the request after a 60 second wait time.

See Table 8, “Subtenant Create Error Codes,” for the full set of errors returned by this operation.

Required role

TenantAdmin

HTTP method

POST

URI

/sysmgmt/tenants/<tenantName>/subtenants

Request parameters

HTTP Header:

- x-atmos-authsource — Specify local.
- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.
- x-atmos-secCompliant — (Optional.) Set to true to create a compliant subtenant that meets the SEC 17a-4f requirements.
• **x-atmos-subtenantname** — Specify the name of the new subtenant.

  The name must be unique within the tenant. It can be up to 255 characters, and can include: the alpha characters a to z (both upper and lower case), digits 0 to 9, or the following special characters: `- _ @ $ ^ ; [ ] ( ) { }.

  The name cannot be the same as the UUID pattern (32 characters long with only the characters a through f (both upper and lower case) and the digits 0-9.

**Request header**

```plaintext
POST /sysmgmt/tenants/t1/subtenants HTTP/1.1
x-atmos-authsource: local
accept: */*
date: Mon, 01 Mar 2010 12:47:21 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145
x-atmos-subtenantname: t1subtenant
```

**Request body**

```plaintext
None
```

**Response header**

```plaintext
HTTP/1.1 201 Created
Connection: close
Date: Mon, 01 Mar 2010 12:47:26 GMT
Set-Cookie: _gui_session_id=5cb85fd0a2ca7d91c9182e503c7dc5b; path=/
Status: 201 Created
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 970ms
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 99
```

**Response body**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<subtenantId>2e705c1a08e5412fb7231055d56733e0</subtenantId>
```
Delete subtenant

Deletes the named subtenant, the UIDs owned by the subtenant, and the objects owned by the UIDs.

When the delete request is submitted, Atmos disables the UIDs for the subtenant and any subsequent I/O requests result in an error. Once all of the data is removed and the capacity is zero, Atmos automatically removes the subtenant.

Capacity is not automatically calculated when the object deletion is completed. Use the TenantAdmin dashboard to recalculate the capacity.

Subtenant deletion requests can be aborted. See “Abort subtenant deletion” on page 34.

Users can view the status of Subtenants and UIDs using the TenantAdmin dashboard. Subtenants and UID status are: Operational, deleting, deleted.

You cannot delete the default subtenant.

It might take 6 months for data not under retention to be deleted. Objects under retention cannot be deleted until the retention period expires.

Required role

TenantAdmin
HTTP method

DELETE

URI

/sysmgmt/tenants/<tenantName>/subtenants/<subtenantName>

Request parameters

HTTP Header:

- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.

Request header

DELETE /sysmgmt/tenants/t1/subtenants/t1 HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 12:46:19 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 12:50:36 GMT
Set-Cookie: _gui_session_id=506a22799a2324f551ef573c53ba467; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 1217ms
ETag: "d434b983e7d60ef5c0566fb6b91476a3"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 63

Response body

<?xml version="1.0" encoding="UTF-8"?>
<deleted>true</deleted>

Get subtenant details

Returns an XML document that contains details about the specified subtenant. The details include: subtenant id, subtenant name, authentication source, status, a list of subtenant admins, capacity, default policy, a list of UIDs, and the a flag that indicates whether the subtenant is SEC compliant.
REST API: Managing Subtenants

The UID list includes the associated shared secret, status, and email

Note: Capacity always returns 0.

Required role

TenantAdmin

HTTP method

GET

URI

/sysmgmt/tenants/<tenantName>/subtenants/<subtenantName>

Request parameters

HTTP Header:
- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.

Request header

GET /sysmgmt/tenants/t1/subtenants/t1 HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 12:44:02 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 12:44:07 GMT
Set-Cookie: _gui_session_id=6dd0cec0433f53619e79c9ad300f37be; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 158ms
ETag: "c50975a52e2dd2501f9de1eaf70b3e2f"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 569
Response body

```xml
<subtenant>
  <id>88a41255117342e4b5f5c0f5c9f4a9df73a</id>
  <name>st_sec</name>
  <authenticationSource>Local</authenticationSource>
  <status>Initialized</status>
  <subtenantAdminList>
  </subtenantAdminList>
  <capacity>0</capacity>
  <defaultPolicySpec>defaultCompliant</defaultPolicySpec>
  <uidSecretList>
  </uidSecretList>
  <secCompliant>true</secCompliant>
</subtenant>
```

List subtenants

Returns an XML document that contains an entry for each subtenant contained by the named tenant. Each entry includes the following elements: name, subtenant id, authentication source, status, and a list of subtenantadmins.

Required role

TenantAdmin

HTTP method

GET

URI

/sysmgmt/tenants/<tenantName>/subtenants

Request parameters

HTTP Header:
- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the x-atmos-tenantadmin.
- `x-atmos-authtype` — Specify password.

Request header

```plaintext
GET /sysmgmt/tenants/t2/subtenants HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 12:40:12 GMT
x-atmos-tenantadmin: t2admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145
```
REST API: Managing Subtenants

Request body

- `showdeleted=true|false` — Optional. When true, lists the subtenants in the deleting or deleted state in the response body. When false, lists only the subtenants in the operational state.

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 12:40:17 GMT
Set-Cookie: _gui_session_id=0ebfeb56a16402dccc138a57002e09635; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 39ms
ETag: "b2fbd5c0dd40009d388a1068c4f3e6e"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 306

Response body

```xml
<?xml version="1.0" encoding="UTF-8"?><subtenantList>
  <subtenant>
    <name>t2</name>
    <id>95923accd86741d0ae2aa4947604eeac</id>
    <authenticationSource>Local</authenticationSource>
    <status>Initialized</status>
    <subtenantAdminList/>
  </subtenant>
</subtenantList>
```

Rename subtenants

Renames an existing subtenant.

Required role

TenantAdmin

HTTP method

PUT

URI

```
/sysmgmt/tenants/<tenantName>/subtenants/<subtenantName>
```

Request parameters

HTTP Header:

- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the x-atmos-tenantadmin.
- **x-atmos-authtype** — Specify password.
- **x-atmos-action** — Specify rename.
- **x-atmos-subtenantname** — Specify the name of the new subtenant.

  The name must be unique within the tenant. It can be up to 255 characters, and can include: the alpha characters a to z (both upper and lower case), digits 0 to 9, or the following special characters: - _ @ $ ^ , ; | [ ] ( ) { }.

  The name cannot be the same as the UUID pattern (32 characters long with only the characters a through f (both upper and lower case) and the digits 0-9).

**Request header**

```
PUT /sysmgmt/tenants/t1/subtenants/t1subtenant HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 12:58:01 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145
x-atmos-action: rename
x-atmos-subtenantname: t1subtennew
```

**Request body**

```
None
```

**Response header**

```
HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 12:58:06 GMT
Set-Cookie: _gui_session_id=0f5224779464f6335898fa21cb96be45; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 206ms
ETag: "036f9f5675132a6f33d4ccb5d6ee7aba"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 63
```

**Response body**

```
<?xml version="1.0" encoding="UTF-8"?&gt;
 &lt;updated&gt;true&lt;/updated&gt;
```
CHAPTER 5
REST API: Managing UIDs

This chapter describes the REST API for UIDs and shared secrets.

- Abort UID deletion .................................................................................................. 48
- Create UID and shared secret .................................................................................. 50
- Delete UID and shared secret .................................................................................. 53
- Modify the UID or shared secret .............................................................................. 55
- Get shared secret .................................................................................................... 56
- List UIDs ................................................................................................................. 58
Abort UID deletion

Cancels a running delete UID request. The UID must be in the deleting state.

When a UID delete request is aborted, the UID is re-enabled and I/O can resume on objects that have not yet been deleted. Objects that have already been deleted cannot be recovered.

Required role

TenantAdmin or SubtenantAdmin

HTTP method

POST

URI

/sysmgmt/tenants/<tenantName>/subtenants/<subtenantID>/uids/<existingUID>/abort_deletion

Request parameters

HTTP Header:

- The authentication headers described in “REST Authentication” on page 13.

Request headers

POST /sysmgmt/tenants/t1/subtenants/t1subtennew/uids/U6/abort_deletion
HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 13:55:48 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

or

POST /sysmgmt/tenants/t1/subtenants/t1subtennew/uids/U6/abort_deletion
HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 13:55:48 GMT
x-atmos-subtenantadmin: st1admin
x-atmos-subtenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 12:50:36 GMT
Set-Cookie:
REST API: Managing UIDs

_response_body_

```xml
<?xml version="1.0" encoding="UTF-8"?>
<aborted>true</aborted>
```
Create UID and shared secret

Creates a UID for the named subtenant, and returns the UID’s system-generated shared secret.

---

**Note:** There is a slight delay before the UID is available on all Atmos nodes.

**Upgrade considerations**

You can perform this operation during upgrade from Atmos 2.0 to a higher version. However, this operation requires access to internal resources which might become unavailable (for up to 60 seconds) during the upgrade of the first RMG. If the create UID request occurs during this 60 second period, the request might fail with this error message:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<errorResponse>
  <errorCode>503</errorCode>
  <errorMessag>Service Unavailable</errorMessage>
</errorResponse>
```

Make sure to code your application to handle this error by either reporting the problem to the end user, or by retrying the request after a 60 second wait time. See Table 9, “UID Create Error Codes,” for the full set of errors returned by this operation.

**Required role**

TenantAdmin or SubtenantAdmin

**HTTP method**

POST

**URI**

```
/sysmgmt/tenants/<tenantName>/subtenants/<subtenantName>/uids
```

**Request parameters**

HTTP Header:

- The authentication headers described in “REST Authentication” on page 13.
- **x-atmos-uid** — Specify the UID to create. When specifying a UID, use these rules for acceptable characters.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Acceptable characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characters</td>
<td>Alphanumeric string (A-Z, a-z, 0-9)</td>
</tr>
<tr>
<td>Special characters</td>
<td>at sign (@), dash (-), period (.), underscore (_)</td>
</tr>
<tr>
<td>Minimum characters</td>
<td>1</td>
</tr>
<tr>
<td>Maximum characters</td>
<td>255</td>
</tr>
<tr>
<td>Case-sensitive?</td>
<td>Yes</td>
</tr>
</tbody>
</table>
If unsupported characters are used, Atmos cannot retrieve the corresponding shared secret for the UID. This limitation does not apply to UIDs stored in the local lockbox.

If the UID will be used with Atmos Cloud Delivery Platform (ACDP), do not use the pipe symbol (|) or you will be unable to gather usage information for that UID as ACDP interprets this character as a delimiter.

- **x-atmos-email** — Specify a valid email address to associate with the UID.
- **x-atmos-expire** — Specify the number of hours after which the shared secret should expire.

### Request header

```plaintext
POST /sysmgmt/tenants/t1/subtenants/<subtenantName>/uids HTTP/1.1
accept: */*
date: Mon, 01 May 2015 13:06:17 GMT
x-atmos-uid: U6
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
x-atmos-email@a@b.com
x-atmos-expire: 9
host: 10.4.5.145
```

### Request body

None

### Response header

```plaintext
HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 13:06:21 GMT
Set-Cookie: _gui_session_id=263a324d452c9bcd224ffd773c79a977; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 725ms
ETag: "f08c7ce17f6e04ba990bc4fe06e43562"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 97
```

### Response body

```xml
<?xml version="1.0" encoding="UTF-8" rigorously="false">
  <sharedSecret>jzpKH{5ee28jM{Tiigk{Ta5vdu8=</sharedSecret>
</xml>
```
UID Create Error Codes

<table>
<thead>
<tr>
<th>HTTP Error Code</th>
<th>Error Message</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Error: UID &lt;xxx&gt; already exists.</td>
<td>The UID already exists</td>
</tr>
<tr>
<td>500</td>
<td>Error: UID name uses invalid characters.</td>
<td>Invalid UID Name</td>
</tr>
<tr>
<td>500</td>
<td>Error: Failed to store UID shared secret.</td>
<td>LDAP or Lock box failed</td>
</tr>
<tr>
<td>500</td>
<td>Error: UID name can not be longer than 255 characters.</td>
<td>The UID name is too long</td>
</tr>
<tr>
<td>401</td>
<td>Authentication failed.</td>
<td>The tenant does not exist.</td>
</tr>
<tr>
<td>401</td>
<td>Authentication failed.</td>
<td>The TenantAdmin credentials were invalid.</td>
</tr>
<tr>
<td>404</td>
<td>Error: Subtenant &lt;xxx&gt; not found</td>
<td>The subtenant does not exist.</td>
</tr>
<tr>
<td>400</td>
<td>Missing required parameter(s)</td>
<td>The Empty subtenant name/admin name/password name</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable: Not support creating UID in lock box during upgrading</td>
<td>Not support Lockbox UID creating during upgrade</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>Database not available</td>
</tr>
</tbody>
</table>
Delete UID and shared secret

Deletes the UID and the objects owned by the UID.

When the delete request is submitted, Atmos disables the UID so that subsequent I/O requests result in an error. Once all of the data is removed and the capacity is zero, Atmos automatically removes the UID.

Capacity is not automatically calculated when the object UID deletion requests can be aborted. See “Abort UID deletion” on page 48.

You can view the status of UIDs using the TenantAdmin or SubtenantAdmin dashboard. UID status are: Operational, deleting, deleted.

Required role

TenantAdmin or SubtenantAdmin

HTTP method

DELETE

URI

/sysmgmt/tenants/<tenantName>/subtenants/<subtenant-name>/uids/<existingUID>

Request parameters

HTTP Header:

- The authentication headers described in “REST Authentication” on page 13.

Request header

DELETE /sysmgmt/tenants/t1/subtenants/t1subtennew/uids/U6 HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 13:55:48 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 13:55:53 GMT
Set-Cookie: _gui_session_id=f258b8ed8d477437778d64537be9dc76; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 420ms
ETag: "d434b983e7d60ef5c0566f6b6b91476a3"
REST API: Managing UIDs

Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 63

Response body

<?xml version="1.0" encoding="UTF-8"?>
<deleted>true</deleted>
Modify the UID or shared secret

Use to:

- Disable or enable the specified UID.
  A UID in the disabled state cannot be used in data requests.
- Regenerate (reset) the UID’s primary shared secret.
- Generate a secondary shared secret.
- Delete the secondary shared secret.

  When the secondary shared secret is removed, Atmos rejects any subsequent authentication requests that use it.
- Remove the primary shared secret and promote the secondary shared secret to primary.
- Reset an expired primary shared secret. If successful, a new shared secret is generated and access to the associated UID data is enabled.
- Update the expiration time for the primary shared secret associated with a UID.

Returns one of the following responses:

- `<disabled>true</disabled>`
- `<enabled>true</enabled>`
- `<reseted>true</reseted>` or `<generatedPrimarySS>true</generatedPrimarySS>`
- `<generatedSecondarySS>true</generatedSecondarySS>`
- `<deletedSecondarySS>true</deletedSecondarySS>`
- `<promotedSecondarySS>true</promotedSecondarySS>`
- `<resetExpireTime>true</resetExpireTime>`
- `<updatedExpireTime>true</updatedExpireTime>`

Required role

TenantAdmin or SubtenantAdmin

HTTP method

PUT

URI

/sysmgmt/tenants/<tenant-name>/subtenants/<subtenant-name>/uids/<UID>

Request parameters

- The authentication headers described in “REST Authentication” on page 13.
**REST API: Managing UIDs**

- **x-atmos-action** — The value can be: disable, enable, reset (deprecated), generatePrimarySS, generateSecondarySS, deleteSecondarySS, promoteSecondarySS, resetExpireTime, updateExpireTime.

### Request header

```plaintext
PUT /sysmgmt/tenants/t1/subtenants/t1subtennew/uids/U6 HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 13:08:34 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145
x-atmos-action: disable
```

### Request body

None

### Response header

```plaintext
HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 13:08:38 GMT
Set-Cookie: _gui_session_id=ecbc0b9d2e4b5c09e669975049d2b637; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 1517ms
ETag: "92ae90fb35d9b6cc00983b51453cc9a1"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 65
```

### Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<disabled>true</disabled>
```

If you attempt to delete the secondary shared secret or if you attempt to promote the secondary shared secret, but it does not exist, Atmos returns a 404 error; for example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<errorResponse>
<errorCode>404</errorCode>
<errorMessag>Error: Secondary shared secret of UID <xxx> not found</errorMessage>
</errorResponse>
```

### Get shared secret

Returns the shared secrets for the specified UID, including the key creation time and number of hours to expiration.

### Required role

TenantAdmin or SubtenantAdmin
HTTP method

GET

URI

/sysmgmt/tenants/<tenant-name>/subtenants/<subtenant-ID>/uids/<UID>

Request parameters

HTTP Header:

• The authentication headers described in “REST Authentication” on page 13.

Request header

GET /sysmgmt/tenants/t1/subtenants/t1subtennew/uids/U6 HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 13:56:53 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 13:56:58 GMT
Set-Cookie: _gui_session_id=937563b2248d77d8b84a4c596553e16d; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 67ms
ETag: "3550963653b8531516e68d67b1596d3"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 97

Response body

If the UID has one shared secret, the response body is similar to this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sharedSecret keyCreateTime="2015-01-16 05:15:37" keyExpireTime="9 hour(s)">8Q3uPkxfglwpwdhgMcJcOJpPY88Q=</sharedSecret>
```

If the UID has two shared secrets, the response body includes the secondary shared secret; for example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sharedSecret keyCreateTime="2015-01-16 05:15:37" keyExpireTime="9 hour(s)">8Q3uPkxfglwpwdhgMcJcOJpPY88Q=</sharedSecret>
<secondarySharedSecret keyCreateTime="2015-01-16 05:15:37" keyExpireTime="9 hour(s)">8Q3uPkxfglwpwdhgMcJcOJpPY88Q=</secondarySharedSecret>
```
List UIDs

Returns an XML document that lists all of the UIDs for the specified tenant.

Required role

TenantAdmin or SubtenantAdmin

HTTP method

GET

URI

/sysmgmt/tenants/t1/subtenants/<subtenant-ID>/uids

Request parameters

HTTP Header:

- The authentication headers described in “REST Authentication” on page 13.

Request header

GET /sysmgmt/tenants/t1/subtenants/<subtenant-name>/uids HTTP/1.1
accept: */*
date: Mon, 01 Mar 2010 13:58:13 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 10.4.5.145

Request body

- showdeleted=true|false — Optional. When true, lists UIDs with the deleting or deleted status in the response body. When false (default), lists the UIDs in the operational state.

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 01 Mar 2010 13:58:17 GMT
Set-Cookie: _gui_session_id=572cd31331a3151139ea5968cae369f7; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 182ms
ETag: "29d66732259f44b4a19c54fb806553e8"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 94
Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<uid_list>
  <uid>U5</uid>
  <uid>U6</uid>
</uid_list>
```
CHAPTER 6
REST API: Managing Location Groups

This chapter describes the REST Location Group API which includes the following operations:

- Create a location group ................................................................. 64
- Delete a location group ................................................................. 66
- Get location group details ............................................................ 67
- List location groups ................................................................. 69
- Modify a location group ................................................................. 70

**Note:** After every modify operation, Atmos distributes the updated group definition to the nodes in the system. This distribution can take up to 10 minutes to complete. Do not use the group in a policy specification until the distribution completes.
Create a location group

Creates a new location group. This operation requires that:

- You provide a minimum of two locations or the operation fails.
- The specified locations exist and are available to the node where you run the operation.

When the command completes, Atmos creates the group and distributes the group definition to the nodes in the system. This can take up to 10 minutes to complete. Do not use the group in a policy specification until the distribution completes.

Required role

SysAdmin

HTTP method

POST

URI

/sysmgmt/locationgroups

Request parameters

- The authentication parameters described in “REST Authentication” on page 13.
- `x-atmos-location-group-name:<location-group-name>` — Specify the name of the location group to create. Naming requirements are:
  - Minimum 2 characters and maximum 255 characters.
  - First character must be a letter.
  - It can only include letters, numbers, and underscore character (_).
  - Names are case-sensitive. The name a1 and A1 refer to different groups.
- `x-atmos-locations:<location, location, ...>` — Specify a comma-separated list of existing locations with a minimum of two locations. Locations names are case-sensitive.

Examples

Successful request

This example shows a successful request to create a location group named locgroup11. It consists of the locations Tokyo and London. This request was successful, so the response header includes a status of 201 created.

Request header

```
POST /sysmgmt/locationgroups HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
  OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
```
Failed request

This example shows an unsuccessful request to create a location group called locgrp11. The request is not successful because a location group of the same name already exists. Failed requests include both a Status: 400 bad request in the response header, and a response body with an error code and message.

Request header

POST /sysmgmt/locationgroups HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
  OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Mary
x-atmos-systemadminpassword:password
x-atmos-location-group-name:locgrp1
x-atmos-locations:Tokyo,London

Response header

HTTP/1.1 400 Bad Request
Date: Mon, 09 May 2011 13:51:42 GMT
Server: Mongrel 1.1.5
Status: 400 Bad Request
X-Runtime: 414ms
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 163
Set-Cookie: _gui_session_id=3d7bc0e5d944f8b685a5c656135c184c; path=/; secure
Connection: close

Response body

<?xml version="1.0" encoding="UTF-8"?>
<errorResponse>
  <errorCode>400</errorCode>
  <errorMessage>Location group already exists.</errorMessage>
</errorResponse>
Delete a location group

Deletes the location group specified on the URI.

Required role

SysAdmin

HTTP method

DELETE

URI

/sysmgmt/locationgroups/<location-group-name>

Request parameters

- The authentication parameters described in “REST Authentication” on page 13.

Example

Successful request

This example shows a successful request to delete the location group named locgroup11. Because the request was successful, the response header includes a status of 204 No Content.

Request header

DELETE /sysmgmt/locationgroups/locgrp11 HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
  OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Mary
x-atmos-systemadminpassword:password

Response header

HTTP/1.1 204 No Content
Date: Mon, 09 May 2011 15:32:07 GMT
Server: Mongrel 1.1.5
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 3829ms
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 0
Set-Cookie: _gui_session_id=8080b952158e728dab750a6df273e889; path=/; secure
Connection: close

Failed request

This example shows an unsuccessful request to delete the location group named locgroup11. The request was not successful so the response header status is 404 not found, and the response body includes an error code 404.
Get location group details

Returns the details for the location group specified on the URI.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/locationgroups/<location-group-name>

Request parameters

- The authentication parameters described in “REST Authentication” on page 13.
Example

Successful request

This example shows a successful request to get details for the location group named LocGrp2. Because the request was successful, the response header includes a status of 200 OK.

Request header

GET /sysmgmt/locationgroups/LocGrp2 HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
  OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Mary
x-atmos-systemadminpassword:password

Response header

HTTP/1.1 200 OK
Date: Fri, 06 May 2011 19:05:28 GMT
Server: Mongrel 1.1.5
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 2311ms
ETag: "79c5ae981f9f19eb570438584a5bdc67"
Cache-Control: private, max-age=0, must-revalidate
Content-Type: application/xml; charset=utf-8
Content-Length: 142
Set-Cookie: _gui_session_id=f12c78e7fb8e2c78fc409af16d3c386f; path=/; secure
Connection: close

Response body

A successful request for a single location group returns an XML document with the locationGroup root element and one <name> and <locations> child element, for example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<locationGroup>
  <name>LocGrp2</name>
  <locations>London,New_York,Tokyo</locations>
</locationGroup>
```

Failed request

This example shows a failed request to get details for the location group named locgrp11. Because the request was unsuccessful, the response header includes a status of 404 Not Found, and a response body with an error code.

Request header

GET /sysmgmt/locationgroups/locgrp11 HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
  OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Anne
x-atmos-systemadminpassword:password

Response header

HTTP/1.1 404 Not Found
List location groups

Returns an XML document that lists the currently defined location group names and their corresponding locations for the authenticated SysAdmin.

Required role

SysAdmin

HTTP method

GET

URI

/sysmgmt/locationgroups

Request parameters

- The authentication parameters described in "REST Authentication" on page 13.

Examples

Successful request

This example shows a successful request to get the list of location groups for the SysAdmin Mary. Because the request was successful, the response header includes a status of 200 OK.

Request header

GET /sysmgmt/locationgroups HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
   OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Mary
x-atmos-systemadminpassword:password
REST API: Managing Location Groups

Response header

HTTP/1.1 200 OK
Date: Fri, 06 May 2011 18:56:51 GMT
Server: Mongrel 1.1.5
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 476ms
ETag: "526a63da220c3b3c0414e0f98c13d796"
Cache-Control: private, max-age=0, must-revalidate
Content-Type: application/xml; charset=utf-8
Content-Length: 598
Set-Cookie: _gui_session_id=6f65c181c3660223ae8df1d00ce507a4; path=/; secure
Connection: close

Response body

<?xml version="1.0" encoding="UTF-8"?>
<locationGroups>
  <locationGroup>
    <name>LogGrp3</name>
    <locations>New_York,Tokyo</locations>
  </locationGroup>
  <locationGroup>
    <name>locgrp1</name>
    <locations>Tokyo,London</locations>
  </locationGroup>
  <locationGroup>
    <name>LocGrp2</name>
    <locations>London,New_York,Tokyo</locations>
  </locationGroup>
  <locationGroup>
    <name>LocGrp3</name>
    <locations>London,New_York,Tokyo</locations>
  </locationGroup>
  <locationGroup>
    <name>LocGrp4</name>
    <locations>London,New_York</locations>
  </locationGroup>
</locationGroups>

Modify a location group

Lets you modify the contents of a location group. You can use this operation to:

- Change the name of an existing location group. If the location group is used in a policy specification, you cannot change its name.
- Change the set of locations in the location group by adding or removing a location.
- Change both the name and set of locations in one operation.

Required role

SysAdmin

HTTP method

PUT
URI

/sysmgmt/locationgroups/<location-group-name>

Request parameters

- `<location-group-name>` — Specify the name of the location group you want to make changes to.
- The authentication parameters described in “REST Authentication” on page 13.

Specify one or both of the following depending on what you want to change:

- `x-atmos-location-group-name:<location-group-name>` — Specify the new name for the location group.
- `x-atmos-locations:<locations>` — Specify the comma-separated list of new locations.

Examples

Successful request

This example shows a successful request to change the location group name from newlocgrp1 to locgrp11, and to also define a new set of locations. Because the request was successful, the response header includes a status of 200 OK.

Request header

PUT/sysmgmt/locationgroups/newlocgrp1 HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Anne
x-atmos-systemadminpassword:password
x-atmos-location-group-name:locgrp11
x-atmos-locations:New_York,London

Response header

HTTP/1.1 200 OK
Date: Mon, 09 May 2011 18:02:58 GMT
Server: Mongrel 1.1.5
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 7320ms
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 0
Set-Cookie: _gui_session_id=946df44d9e064144cc10740161e7fe66; path=/; secure
Connection: close

Failed request

A request can fail for several reasons. When it does fail, you get a status of 400 or 404 depending on the reason for the failure. In this example, the request fails because the location group name does not exist.
REST API: Managing Location Groups

Request header

PUT /sysmgmt/locationgroups/locgr111 HTTP/1.1
User-Agent: curl/7.19.4 (x86_64-unknown-linux-gnu) libcurl/7.19.4
  OpenSSL/0.9.8g zlib/1.2.3
Host: localhost
Accept: */*
x-atmos-systemadmin:Mary
x-atmos-systemadminpassword:password
x-atmos-location-group-name:locgrp11
x-atmos-locations:London,Tokyo,New_York

Response header

HTTP/1.1 404 Not Found
Date: Mon, 09 May 2011 17:56:00 GMT
Server: Mongrel 1.1.5
Status: 404 Not Found
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 375ms
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 163
Set-Cookie: _gui_session_id=236ab7b6910729cfd14c5af3d327bf87; path=/; secure
Connection: close

Response body

<?xml version="1.0" encoding="UTF-8"?>
<errorResponse>
  <errorCode>404</errorCode>
  <errorMessage>Location group does not exist.</errorMessage>
</errorResponse>
CHAPTER 7
REST API: Managing Policy Selectors

This chapter describes the Policy Selector REST API.

- Assign a policy selector................................................................. 74
- Create a policy selector ............................................................... 75
- Delete a policy selector ............................................................... 78
- Get policy selector ................................................................. 79
- List policy selectors ................................................................. 80
- Update a policy selector ......................................................... 81
## Assign a policy selector

Assigns the specified policy selector to the specified subtenant.

If the subtenant is a compliant subtenant (that is, it conforms to the SEC 17a-4f requirements), the policy selector you assign must also be compliant.

**Note:** You can assign additional policy selectors to a subtenant by re-issuing the assign API. Re-issue the API for each additional policy selector that you want to assign to the subtenant.

### Required role

TenantAdmin

### HTTP method

POST

### URI

`/sysmgmt/<tenantName>/policyselectors/assign`

### Request parameters

**HTTP Header:**

- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the x-atmos-tenantadmin.
- `x-atmos-authtype` — Specify `password`.

**HTTP Body:**

- `selector` — The policy selector name to assign.
- `policy_type` — Set to 0.
- `subtenants[]` — The name of the subtenant you are assigning the policy to.

### Request header

```plaintext
POST /sysmgmt/t1/policyselectors/assign HTTP/1.1
accept: */*
content-type: application/x-www-form-urlencoded
date: Thu, 11 Mar 2010 07:30:43 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
content-length: 46
host: 168.159.121.241
```

### Request body

```plaintext
selector=PS1&policy_type=0&subtenants[]=t1
```
Create a policy selector

Creates a policy selector within the specified tenant. You must assign a policy selector to a subtenant before it can be used.

Required role

TenantAdmin

HTTP method

POST

URI

/sysmgmt/<tenantName>/policyselectors

Request parameters

HTTP Header:

- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the `x-atmos-tenantadmin`.
- `x-atmos-authtype` — Specify password.

Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 11 Mar 2010 07:30:47 GMT
Set-Cookie: _gui_session_id=5b451a31926af440ea4027b6a4789b7e; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 4031ms
ETag: "724319415dc762386c0f5c6f15f97300"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 65

Response body

<?xml version="1.0" encoding="UTF-8"?>
<assign>assigned</assign>
Table 10  HTTP Body elements

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>perform_action</td>
<td>Specify create.</td>
</tr>
<tr>
<td></td>
<td>Example: perform_action=create</td>
</tr>
<tr>
<td>entry_index</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td></td>
<td>Example: entry_index=</td>
</tr>
<tr>
<td>entry[edit_mode]</td>
<td>Specify the type of metadata the policy selector will act on. Valid values:</td>
</tr>
<tr>
<td></td>
<td>0 (user metadata), 1 (system metadata), 2 (advanced).</td>
</tr>
<tr>
<td></td>
<td>Example: entry[edit_mode]=1</td>
</tr>
<tr>
<td>entry[spec_name]</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[spec_name]=</td>
</tr>
<tr>
<td>other[sec_compliant]</td>
<td>Specify true if the policy selector must conform to the SEC 17a-4f</td>
</tr>
<tr>
<td></td>
<td>requirements.</td>
</tr>
<tr>
<td>entry[event]</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[event]=</td>
</tr>
<tr>
<td>entry[id]</td>
<td>Specify the name of the selector to create. The policy selector name can be</td>
</tr>
<tr>
<td></td>
<td>up to 255 alpha characters long. It can contain the letters a to z (both</td>
</tr>
<tr>
<td></td>
<td>upper and lower case), any digits 0 to 9, or the following special</td>
</tr>
<tr>
<td></td>
<td>characters: - _ @ ^ _ ` ] ( ) {}</td>
</tr>
<tr>
<td></td>
<td>Note: The POX API does not validate this parameter. Follow these</td>
</tr>
<tr>
<td></td>
<td>naming rules to ensure data consistency.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[id]=ps2</td>
</tr>
<tr>
<td>other[specname]</td>
<td>Specify the name of an existing policy specification. This binds the</td>
</tr>
<tr>
<td></td>
<td>policy selector to the policy specification.</td>
</tr>
<tr>
<td></td>
<td>Example: other[specname]=default</td>
</tr>
<tr>
<td>entry[user_field_name]</td>
<td>Specify a user-metadata tag name. Use with user_operator and</td>
</tr>
<tr>
<td></td>
<td>user_value. The value of edit_mode must be 0.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[user_field_name]=</td>
</tr>
<tr>
<td>entry[user_operator]</td>
<td>Specify a value comparison operator. Valid values: EQUALS, ENDS WITH,</td>
</tr>
<tr>
<td></td>
<td>STARTS WITH, CONTAINS, and the following symbols: &lt;, =, &gt;, &lt;=, &gt;=.</td>
</tr>
<tr>
<td></td>
<td>Use with user_operator and user_value. The value of</td>
</tr>
<tr>
<td></td>
<td>edit_mode must be 0.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[user_operator]=EQUALS</td>
</tr>
<tr>
<td>entry[user_value]</td>
<td>Specify a user metadata value. The value can be text, a number or a</td>
</tr>
<tr>
<td></td>
<td>date that matches the value specified in the user_field_name parameter.</td>
</tr>
<tr>
<td></td>
<td>Use with user_operator and user_field_name. The value of</td>
</tr>
<tr>
<td></td>
<td>edit_mode must be 0.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[user_value]=</td>
</tr>
<tr>
<td>entry[user_event]</td>
<td>Specify a user metadata trigger event to activate this policy selector.</td>
</tr>
<tr>
<td></td>
<td>Valid values: ON_CREATE, ON_UMD_UPDATE, ON_SMD_UPDATE. The value of</td>
</tr>
<tr>
<td></td>
<td>edit_mode must be 0.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[user_event]=ON_CREATE</td>
</tr>
</tbody>
</table>
Create a policy selector

REST API: Managing Policy Selectors

Table 10  HTTP Body elements

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry[sys_field_name]</td>
<td>Specify a system metadata tag name. Valid values are: atime, mtime, ctime, itime, uid, gid, size, objname. Use with sys_operator, sys_value, and sys_event. The value of edit_mode must be 1.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> size is NOT available for the 'ON_CREATE' event.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[sys_field_name]=objname</td>
</tr>
<tr>
<td>entry[sys_operator]</td>
<td>Specify a system-metadata-value comparison operator. Valid values: EQUALS, ENDS WITH, STARTS WITH, CONTAINS, and the symbols &lt;=, &gt;=. Use with sys_field_name, sys_value, and sys_event. The value of edit_mode must be 1.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[sys_operator]=ENDS WITH</td>
</tr>
<tr>
<td>entry[sys_value]</td>
<td>Specify a system metadata value. The value can be text, a number or a date that matches the value specified in the sys_field_name parameter. Use with sys_field_name, sys_operator, and sys_event. The value of edit_mode must be 1.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[sys_value]=basic_2_replica_2s_sub</td>
</tr>
<tr>
<td>entry[sys_event]</td>
<td>Specify a system metadata trigger event for policy activation. Valid values: ON_CREATE, ON_UMD_UPDATE, ON_SMD_UPDATE. Use with sys_field_name, sys_operator, and sys_value. The value of edit_mode must be 1.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[sys_event]=ON_CREATE</td>
</tr>
<tr>
<td>entry[xquery]</td>
<td>Specify an Xquery string to match objects against. Use with adv_event. The value of edit_mode must be 2. Example: entry[xquery]=&quot;for+%24c+in+collection%28%29+return+%24c&quot;</td>
</tr>
<tr>
<td></td>
<td>The example represents the escaped string for 'for $c in collection() return $c'</td>
</tr>
<tr>
<td>entry[adv_event]</td>
<td>Specify an Xquery trigger event for policy activation. Valid values: ON_CREATE (non-system-metadata modes), ON_UMD_UPDATE, ON_SMD_UPDATE. Use with xquery. The value of edit_mode must be 2. Example: entry[adv_event]=ON_CREATE</td>
</tr>
<tr>
<td>operation_type</td>
<td>Values are: sync (the default) or async.</td>
</tr>
</tbody>
</table>

Request header

POST /sysmgmt/<tenantName>/policyselectors HTTP/1.1
accept: */*
content-type: application/x-www-form-urlencoded
date: Thu, 11 Mar 2010 07:04:30 GMT
x-atmos-tenantadmin: tadmin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
content-length: 455
host: 168.159.121.241:3000
REST API: Managing Policy Selectors

Request body


Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 11 Mar 2010 07:04:34 GMT
Set-Cookie: _gui_session_id=21a2ec2f1cb8441faa16e6a507713b0b; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 16029ms
ETag: "1197f36445213ae6d91d56d8b8c1866"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 64

Response body

<?xml version="1.0" encoding="UTF-8"?>
<create>created</create>

Delete a policy selector

Deletes the specified policy selector.

Required role

TenantAdmin

HTTP method

DELETE

URI

/sysmgmt/<tenantName>/policyselectors/<policyselectorName>

Request parameters

HTTP Header:

• x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
• x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
• x-atmos-authtype — Specify password.

Request header

DELETE /sysmgmt/t1/policyselectors/PS1 HTTP/1.1
Get policy selector

Retrieves details about the specified policy selector, such as policy selector name, status, ID, edit mode, metadata tag, match operator, metadata value, and event.

Required role

TenantAdmin

HTTP method

GET

URI

/sysmgmt/<tenantName>/policyselectors/<policySelectorName>

Request parameters

HTTP Header:
- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.
REST API: Managing Policy Selectors

Request header

GET /sysmgmt/t1/policyselectors/PolicySelector1 HTTP/1.1 accept: */* date: Tue, 02 Mar 2010 09:49:41 GMT x-atmos-tenant-admin: t1admin x-atmos-tenant-admin-password: password x-atmos-auth-type: password host: 168.159.121.223

Request body

None

Response header

HTTP/1.1 200 OK Connection: close Date: Tue, 02 Mar 2010 09:49:46 GMT Set-Cookie: _gui_session_id=a89188ffe9d613be22163528b0085c35; path=/ Status: 200 OK x-atmos-sysmgmt-version: 1.0.0 X-Runtime: 2793ms ETag: "abf7470b2df41053fefa1964bdf0b3d" Cache-Control: private, max-age=0, must-revalidate Server: Mongrel 1.1.5 Content-Type: application/xml; charset=utf-8 Content-Length: 350

Response body

<?xml version="1.0" encoding="UTF-8"?>
<policy_selector>
  <name>PolicySelector1</name>
  <status>Completed</status>
  <policy_id></policy_id>
  <edit_mode>systemmd</edit_mode>
  <metadata_tag>objname</metadata_tag>
  <match_operator>EQUALS</match_operator>
  <metadata_value>test</metadata_value>
  <on_event>ON_CREATE</on_event>
</policy_selector>

List policy selectors

Retrieves a list of all policy selectors defined for the specified tenant.

Required role

TenantAdmin

HTTP method

GET

URI

/sysmgmt/<tenantName>/policyselectors
REST API: Managing Policy Selectors

Request parameters

HTTP Header:

- **x-atmos-tenantadmin** — Specify a valid username of a user in the TenantAdmin role.
- **x-atmos-tenantadminpassword** — Specify the password for the x-atmos-tenantadmin.
- **x-atmos-authtype** — Specify password.

Request header

```
GET /sysmgmt/t1/policyselectors HTTP/1.1
accept: */*
date: Tue, 02 Mar 2010 09:48:06 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 168.159.121.223
```

Request body

None

Response header

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 02 Mar 2010 09:48:10 GMT
Set-Cookie: _gui_session_id=15aa8cec370ff9b58e46f104d22e6179; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 159ms
ETag: "4e3ab07f017198befb35b6609962091a"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 269
```

Response body

```
<?xml version="1.0" encoding="UTF-8"?>
<policy_selector_list>
  <policy_selector>
    <name>PolicySelector1</name>
    <expression>objname equals test</expression>
    <on_event>ON_CREATE</on_event>
    <spec>default</spec>
  </policy_selector>
</policy_selector_list>
```

Update a policy selector

Replaces the existing policy selector definition.

Required role

TenantAdmin
HTTP method

PUT

URI

/sysmgmt/<tenantName>/policyselectors/<policySelectorName>

Request parameters

HTTP Header:

- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.

HTTP Body:

Table 11  HTTP Body

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>perform_action</td>
<td>Specify modify. Example: perform_action=modify</td>
</tr>
<tr>
<td>entry_index</td>
<td>Not used. Supply the parameter with no value. Example: entry_index=</td>
</tr>
<tr>
<td>entry[edit_mode]</td>
<td>Specify what type of data the policy selector you are creating. Valid values: 0 (user metadata), 1 (system metadata), 2 (advanced). Example: entry[edit_mode]=1</td>
</tr>
<tr>
<td>entry[spec_name]</td>
<td>Not used. Supply the parameter with no value. Example: entry[spec_name]=</td>
</tr>
<tr>
<td>other[sec_compliant]</td>
<td>Specify true if the policy selector must conform to the SEC 17a-4f requirements.</td>
</tr>
<tr>
<td>entry[event]</td>
<td>Not used. Supply the parameter with no value. Example: entry[event]=</td>
</tr>
<tr>
<td>entry[id]</td>
<td>Specify the name of the selector to create or modify. Example: entry[id]=ps2</td>
</tr>
<tr>
<td>other[specname]</td>
<td>Specify the name of an existing policy specification. This binds the policy selector to the policy specification. Example: other[specname]=default</td>
</tr>
<tr>
<td>entry[user_field_name]</td>
<td>Specify a user-metadata tag name. Use with user_operator and user_value. The value of edit_mode must be 0. Example: entry[user_field_name]=</td>
</tr>
<tr>
<td>entry[user_operator]</td>
<td>Specify a value comparison operator. Valid values: EQUALS, ENDS WITH, STARTS WITH, CONTAINS, and the following symbols: &lt;, &gt;, &lt;=, &gt;= Use with user_operator and user_value. The value of edit_mode must be 0. Example: entry[user_operator]=EQUALS</td>
</tr>
</tbody>
</table>
Request header

PUT /sysmgmt/t1/policyselectors/def HTTP/1.1
REST API: Managing Policy Selectors

Request body


Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 11 Mar 2010 07:34:04 GMT
Set-Cookie: _gui_session_id=c21db1b99e0af5386ecf1804cae26d67; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 19606ms
ETag: "9a530389fadff917b4b8744d8b647142"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 64

Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<updated>updated</updated>
```
CHAPTER 8
REST API: Managing Policy Specifications

This chapter describes Policy Specification REST API.

- Create or update a policy specification ............................................................... 86
- Delete a policy specification ............................................................................... 96
- Get details for a policy specification ............................................................... 97
- List policy specifications .................................................................................. 98
- Example: How to create a policy specification ................................................ 100
Create or update a policy specification

Policy updates are deprecated in Atmos Version 2.1.5.
Creates a new policy specification, or performs a full replacement of an existing policy specification.
The create and update operations require the same set of parameters.
See “Example: How to create a policy specification” on page 100

Required role
TenantAdmin

HTTP method
POST

URI
To create a policy specification, use this URI:
/sysmgmt/<tenantName>/policies

Request parameters
HTTP Header:
- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.
The HTTP body is comprised of the set of parameters defined in:
- Table 12, “General parameters.”
- Table 13, “Metadata storage parameters.”
• **Table 14, “Replica definition parameters.”**

**Table 12  General parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>perform_action</td>
<td>Specify <code>create</code> or <code>modify</code>. The modify operation is deprecated beginning with Atmos version 2.1.5.</td>
</tr>
<tr>
<td>other[specname]</td>
<td>Specify the name of the policy to create. For create operations, this name can be up to 255 characters long. It can contain the letters a to z (both upper and lower case), any digits 0 to 9, or the following special characters: _ @ $ ^ , ;</td>
</tr>
<tr>
<td>other[sec_compliant]</td>
<td>Specify true if the policy specification must conform to the SEC 17a-4f requirements.</td>
</tr>
<tr>
<td>entry[spec_name]</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td>new_spec</td>
<td>Specify <code>true</code>.</td>
</tr>
<tr>
<td>replica_id</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
</tbody>
</table>

**Table 13  Metadata storage parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metadata[x][location_modifier]</td>
<td>Specifies the matching criteria for the location_place. [x] indicates replica number. For metadata the value is always 1. Valid values are: <code>sameAs</code> and <code>otherThan</code>. You cannot combine the location_modifier otherThan with the location_place of ANY.</td>
</tr>
</tbody>
</table>
| metadata[x][location_place] | Specifies the location where Atmos should create new files and directories. [x] indicates replica number. For metadata the value is always 1. Valid values:  
  • **ANY** (the default location): For the Web-service object interface, this is a location at or close to the client; for the Web-service namespace interface or file-system interface, this is the location of the parent directory. This value is case-sensitive.  
  • `$client`: designates the location of the client where an operation is executed; A policy using the `$client` location descriptor stores the replica on a storage service at the same location as the client that initiated the request.  
  • `rmg locations`: A specific RMG location. Restrictions: You cannot combine the location_modifier otherThan with the location_place of ANY. |

**Note:** Contact EMC Customer Support before using ANY.
Use the parameters in Table 14, “Replica definition parameters” to define the replica locations and their storage service attributes as follows:

- **Customize:** If enabled, you must define:
  - A replica location — The location can be a single location or a location expression. For more information about replica location expressions, see “Understanding replica location expressions” on page 94.
  - The storage services to apply to the replica.

- **Enable stripe:** Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Striping allows you to put blocks of data of the same replica on different nodes.

- **Federate:** If you enable federation, you must also specify the `service_name` parameter. You can only specify the federation names that were defined by the SysAdmin.

- **GeoParity:** Provides data redundancy without the overhead of replication. It divides an object into rows, each of which contains \( m \) data fragments and \( k \) code fragments. Each fragment is stored on a different disk. Using this method, each row of the object can be reconstructed from any \( m \) of the total set of fragments, tolerating up to \( k \) unavailable fragments.

  It has an enhanced data protection mechanism, concurrent write and read operations of an object may result in an I/O error until the write operation has completed.

  A fragment distribution across multiple sites has a WAN latency dependency on performance, therefore should be designed in consultation with EMC.

  Use the `spec[x][enable_erasure]` parameter to specify GeoParity.

  **Note:** Contact EMC Customer Support before using the GeoParity 9/3 configuration.

  For Atmos Virtual Edition (AVE) configurations, use protected storage (RAID or Mirroring).

  **Note:** For GeoParity use with AVE, contact EMC personnel.

  **Note:** You can define any number of replicas, this is an array, and every element the array represents a single replica. The \([x]\) indicates the replica number. Replicas start at 1.
Policies with sec_compliant set to true require a minimum of 2 sync or 1 EC sync replica.

**Table 14 Replica definition parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec[x][type]</td>
<td>Specifies the replica type. Valid values: <code>sync</code>, <code>async</code>. If there is more than one replica, then the value for at least one must be <code>sync</code>. <strong>Example:</strong> <code>spec[1][type]=sync</code></td>
</tr>
</tbody>
</table>
| spec[x][location1] [modifier] | Specify the location comparison. Valid values:  
  • `sameAs`  
  • `otherThan`. **Example:** `spec[1][location1][modifier]=sameAs`  
  **Note:** The `spec[x][location_modifer]` parameter is supported for backwards compatibility. |
| spec[x][location1] [place] or spec[x][location1] [group] | Specify the placement location. Use the [place] parameter to specify a single location, or [group] to specify a location group.  
  Valid values for [place] include:  
  • `ANY` (no replica customization)  
  **Note:** Contact EMC Customer Support before using ANY.  
  • A specific RMG location (case-sensitive)  
  • `$client`  
  • `$clientCreateLoc`  
  Valid values for [group] include:  
  • `ANY`  
  • A specific location group (case-sensitive)  
  • `$clientLocationGroup`  
  • `$createClientLocationGroup`  
  This parameter provides backward compatibility for: `spec[1][location_place]=$client`  
  **Example:** `spec[1][location1][place]=$client` |
| spec[x][location2][operator] | (Optional). Specify the logical operator when building a location expression. Values are:  
  • `AND`  
  • `OR`  
  Rules for logical expressions:  
  • Location expressions that use the `sameAs` modifier can be operated on by AND and OR.  
  • Location expressions that use the `otherThan` modifier can be operated on by AND only.  
  • Location expressions that use `sameAs` and `otherThan` can be operated on by AND only.  
  **Example:** `spec[1][location2][operator]=or` |
REST API: Managing Policy Specifications

**Table 14** Replica definition parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| spec[x][location2][modifier]| (Optional). Specify the location modifier when defining the second location in a location expression. Values are:  
• sameAs  
• otherThan.  
**Example:**  
spec[1][location2] [modifier]=sameAs |
| spec[x][location2][place] or spec[x][location2][group] | (Optional). Specify the location when defining the second location in a location expression. Values for [place] are:  
• ANY  
| Note: Contact EMC Customer Support before using ANY.  
• A specific RMG location (case-sensitive)  
• $client  
• $clientCreateLoc  
Values for [group] are:  
• A specific location group (case-sensitive)  
• $clientLocationGroup  
• $createClientLocationGroup  
**Example:**  
spec[1][location2] [place]=NewYork |
| spec[x][enable_customize]    | (Optional.) Use to specify the location and storage server attributes parameters:  
**Example:** spec[1][enable_customize]=on |
| spec[x][ssattrs_placement]  | Specify the storage server placement method that defines how the storage service allocates new objects to the physical disks under its management. Options are:  
**GREEN** — Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Only one or a few disks are active; the rest are spun down for energy conservation. When the active disks fill up, the spun-down disks are woken up and used.  
**OPTIMAL** or **BALANCED** — When either option is selected, Atmos writes objects to disks using a weighted random process so that disks with more free space have a higher chance of being used. For example, if disk 1 has 40% free space and disk 2 has 80% free space, disk 2 is twice as likely to be used than disk 1. This ensures that disk usage is balanced across all disks.  
**Example:** spec[1][ssattrs_placement]=OPTIMAL |
**Table 14  Replica definition parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| spec[x][ssattrs_actions] | Specify the data-at-rest services that are applied to replicas when they are stored. Valid values are:  
ANY — Pick any storage server, regardless of how it is configured (e.g., whether it is configured for compression).  
NONE — No services; normal writes and reads. This is the default.  
COMPRESSION — Data is compressed on write and decompressed on read. This conserves disk space but introduces processing overhead on the server, so typically it has a negative impact on performance. Compression can be used successfully for objects that are written once and accessed rarely.  
DEDUPLICATION — Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Multiple copies of the data are removed, and only one copy of the data is maintained. This is intended to conserve storage capacity. |
| spec[x][enable_stripe] | Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Optional. Use to enable striping. If used, you must also specify the stripe_size integer, stripe_number integer, stripe_units.  
If you enable stripe, you can also enable customize, but you cannot enable flexout.  
**Example:** spec[1][enable_stripe]=on |
| spec[x][stripe_number] | Use when striping is enabled. Specifies the number of nodes to stripe across.  
**Example:** spec[1][stripe_number]= |
| spec[x][stripe_size] | Use when striping is enabled. Specify the amount of data written to each node.  
**Example:** spec[1][stripe_size]= |
| spec[x][stripe_units] | Use when striping is enabled. Specify the stripe units. Valid values: B (no striping), KB, MB, GB, TB.  
**Example:** spec[1][stripe_units]=B |
| spec[x][enable_flexout] | (Optional). Specify this parameter to enable federation.  
When you enable flexout, you must specify the service_name parameter. When enabled, you cannot specify striping, customization or GeoParity.  
**Example:** spec[1][enable_flexout]=on |
| spec[x][service_name] | Use when federation is enabled. Specify the service_name's valid value is the [Cloud Federations name list] in the system configuration.  
**Example:** spec[1][service_name]= |
| spec[x][enable_erasure] | Optional. Specify to enable GeoParity. Defaults to off if not supplied.  
**Example:** spec[1][enable_erasure]=on |
| spec[x][alg] | Specifies the GeoParity algorithm. Value is always CRS.  
**Example:** spec[1][alg]=CRS |
Table 14  Replica definition parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| `spec[x][fragmentation]`        | Use to specify the GeoParity fragmentation scheme. Values are: 
  - fragmentation1 — for 9:3 (a storage overhead of 33%). Tolerates up to 3 simultaneous node or drive failures. 
  - fragmentation2 — for 10:6 (a storage overhead of 60%). Tolerates up to 6 simultaneous node or drive failures. 
  
  **Note:** Contact EMC Customer Support before using the GeoParity 9:3 configuration. 

  **Example:** `spec[1][fragmentation]=fragmentation2` |
| `other[read_access]`            | Specify the replicas to use for read access. Valid values: 
  - geographic: Chooses the closest replica in geographic terms. This is the default. 
  - random: Picks a replica at random. 

  **Example:** `other[read_access]=geographic` |
| `other[enable_retention]`       | (Optional). Specify to enable retention. For sec_compliant policy specifications, retention is enabled by default; otherwise it defaults to off when not supplied. Retention is a period of time during which the object cannot be modified or deleted. Atmos evaluates the retention period based on the object create time. 
  For sec_compliant policy specifications, you cannot specify a start delay window. For non-compliant policies, if you enable retention, you must also specify a Start Delay Window using the `other[retention_delay_xxx]` parameters. 
  The Start Delay Window specifies the amount of time to wait before applying the retention period. It is possible to apply a new retention period (after the first one has elapsed), but the time evaluation continues to be based on the object's create time. 
  For non-compliant policy specifications, when `enable_retention` is set to on, you must supply at least one of the `other[retention_delay_xxx]` parameters, and one of the `other[retention_xxx]` parameters. 
  If you specify both retention and deletion, the deletion length must be longer than retention length. 

  **Example:** `other[enable_retention]=on` |
| `other[retention_delay_year]`   | The start delay year. 

  **Example:** `other[retention_delay_year]=2` |
| `other[retention_delay_month]`  | The start delay month. 

  **Example:** `other[retention_delay_month]=3` |
| `other[retention_delay_day]`    | The start delay day. 

  **Example:** `other[retention_delay_day]=3` |
| `other[retention_delay_hour]`   | The start delay hour. 

  **Example:** `other[retention_delay_hour]=5` |
| `other[retention_delay_minute]` | The start delay minute. 

  **Example:** `other[retention_delay_minute]=1` |
**Examples**

**Create**

**Request header**

POST /sysmgmt/t1/policies HTTP/1.1
accept: */*
content-type: application/x-www-form-urlencoded
date: Mon, 08 Mar 2010 19:52:35 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
content-length: 941
host: 10.4.8.175

---

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>other[retention_delay)_second]</code></td>
<td>The start delay second. Example: <code>other[retention_delay_second]=3</code></td>
</tr>
<tr>
<td><code>other[retention_year]</code></td>
<td>Specify the retention year. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[retention_month]</code></td>
<td>Specify the retention Month. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[retention_day]</code></td>
<td>Specify the retention day. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[retention_hour]</code></td>
<td>Specify the retention hour. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[retention_minute]</code></td>
<td>Specify the retention minute. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[retention_second]</code></td>
<td>Specify the retention second. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[enable_deletion]</code></td>
<td>(Optional). Specify to enable a retention period. Defaults to off when not specified. Deletion specifies the time after an object was created that it should be deleted. This time must be longer than the retention time since it is not possible to delete an object that is being retained. Deletion applies only to files, not directories. If you specify both retention and deletion, the deletion length must be longer than retention length. Example: <code>other[enable_deletion]=on</code></td>
</tr>
<tr>
<td><code>other[deletion_year]</code></td>
<td>Specify the deletion year. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[deletion_month]</code></td>
<td>Specify the deletion month. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[deletion_day]</code></td>
<td>Specify the deletion day. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
<tr>
<td><code>other[deletion_hour]</code></td>
<td>Specify the deletion hour. The valid values are natural numbers like 0, 1, 2, and so on. There is no upper limit.</td>
</tr>
</tbody>
</table>
Request body

```plaintext
perform_action=create&other[specname]=def&entry[spec_name]=
&new_spec=true &replica_id=&metadata[1][location_modifier]=sameAs
&metadata[1][location_place]=ANY&spec[1][type]=sync&spec[1][location1]
[modifier]=sameAs&spec[1][location1][place]=ANY
&spec[1][ssattrs_placement]=OPTIMAL&spec[1][ssattrs_actions]=ANY
&spec[2][type]=sync&spec[2][location1][modifier]=sameAs
&spec[2][location1][place]=ANY&spec[2][ssattrs_placement]=OPTIMAL
&spec[2][ssattrs_actions]=ANY&operation_type=sync
```

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 08 Mar 2010 19:52:40 GMT
Set-Cookie: _gui_session_id=370d6a8676f4df43797e4e5f04c51f; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 16303ms
ETag: "22216908522de735aa0977926afdd3ae"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 63

Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<created>true</created>
```

Understanding replica location expressions

You define a replica location using a location modifier (such as sameAs or otherThan), and
a location (which can be a single location or a location group).

To specify a single location, use the following parameters:

```plaintext
spec[x][location1][modifier] = <modifier>
spec[x][location1][place] = <location>
```

To specify one location group, use the following parameters:

```plaintext
spec[x][location1][modifier] = <modifier>
spec[x][location1][group] = <location-group-name>
```

To create a location expression, specify the first location by using:

```plaintext
spec[x][location1][modifier] = <sameAs|otherThan>
spec[x][location1][place] = <location>
```

or

```plaintext
spec[x][location1][modifier] = <sameAs|otherThan>
spec[x][location1][group] = <location-group-name>
```

followed by the second location:

```plaintext
spec[x][location2][operator] = <AND | OR>
spec[x][location2][modifier] = <sameAs|otherThan>
spec[x][location2][place] = <location>
```

or

```plaintext
spec[x][location2][operator] = <AND | OR>
spec[x][location2][modifier] = <sameAs|otherThan>
spec[x][location2][group] = <location-group-name>
```
For example,

```
spec[1][location1][modifier]=sameas
spec[1][location1][place]=London
spec[1][location2][operator]=or
spec[1][location2][modifier]=sameas
spec[1][location2][place]=Tokyo
```

**Rules**

Keep these rules in mind when you define location expressions:

- Expressions that include the `sameAs` modifier can use both the AND and the OR operators; for example, you can define both of the following expressions:

  ```
sameAs Europe AND sameAs Asia
sameAs Europe OR sameAs Asia
```

- Expressions that include the `otherThan` modifier can use only the AND operator; for example:

  ```
otherThan London AND otherThan MexicoCity
```

- Expressions that include a combination of the modifiers, sameAs and otherThan, can include only the AND operator; for example:

  ```
sameAs London AND otherThan NewYork
```

- The first operand of a composite expression takes precedence over the second operand. This means that if you have:

  An operand that is a single location (such as Beijing) and an operand that is a location group (such as China), and the single location is in the location group, and the modifiers are otherThan and sameAs

  You should specify the expression like this:

  ```
otherThan <location> AND sameAs <location group>
```

For example, suppose you have a location group called China which includes Shanghai and Beijing. Make sure to specify:

```
otherThan Beijing AND sameAs China
```

and not:

```
sameAs China AND otherThan Beijing
```

When you specify `otherThan Beijing AND sameAs China`, Atmos distributes the replica only in Shanghai, but if you specified the reverse (sameAs China AND otherThan Beijing), Atmos would distribute the replica on any of the nodes in the China location group (including Beijing).

---

**Note:** If you create an expression that does not result in a location with valid resources, Atmos will not allow you to create the policy.
Delete a policy specification

Deletes the specified policy. Returns <deleted>true</deleted>.

You cannot delete a policy specification if it is used by a policy selector or a policy transition, or if it is assigned as the default policy for a subtenant.

Required role

TenantAdmin

HTTP method

DELETE

URI

/sysmgmt/<tenantName>/policies/<policyName>

Request parameters

HTTP Header:

- x-atmos-tenantadmin — Specify a valid username of a user in the TenantAdmin role.
- x-atmos-tenantadminpassword — Specify the password for the x-atmos-tenantadmin.
- x-atmos-authtype — Specify password.

Request header

DELETE /sysmgmt/t1/policies/Policy1 HTTP/1.1
accept: */*
date: Tue, 02 Mar 2010 09:23:59 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 168.159.121.223

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Tue, 02 Mar 2010 09:24:03 GMT
Set-Cookie: _gui_session_id=b2567202e721007ae2d112a1250eb6f3; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 2597ms
ETag: "d434b983e7d60ef5c0566fb6b91476a3"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 63
Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<deleted>true</deleted>
```

Get details for a policy specification

Returns an XML document that details the specified policy specification. Details include:
- policy name,
- metadata specification,
- replica details.

Required role

TenantAdmin

HTTP method

GET

URI

/sysmgmt/<tenantName>/policies/<policyName>

Request parameters

- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the x-atmos-tenantadmin.
- `x-atmos-authtype` — Specify password.

Request header

```
GET /sysmgmt/t1/policies/Policy5 HTTP/1.1
accept: */*
date: Tue, 02 Mar 2010 09:38:07 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 168.159.121.223
```

Request body

None

Response header

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 02 Mar 2010 09:38:11 GMT
Set-Cookie: _gui_session_id=9895fe61ae63bae6059e7cceb5bb866; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 1576ms
ETag: "597159fc8070fbc7801cfb20ae6113fc"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
```

Get details for a policy specification
REST API: Managing Policy Specifications

### Response body

```xml
<policy>
  <name>policy5</name>
  <status>Completed</status>
  <metadata_location_modifier>sameAs</metadata_location_modifier>
  <metadata_location_place>$client</metadata_location_place>
  <replica_list>
    <replica>
      <id>0</id>
      <type>sync</type>
      <enable_customize>on</enable_customize>
      <location>sameAs LT or sameAs New_York</location>
      <location_modifier>none</location_modifier>
      <location_place></location_place>
      <ssattrsPlacement></ssattrsPlacement>
      <ssattrsActions>NONE</ssattrsActions>
      <enable_stripping>on</enable_stripping>
      <stripe_number></stripe_number>
      <stripe_size></stripe_size>
      <stripe_units></stripe_units>
      <enable_flexout>on</enable_flexout>
      <cloud_service></cloud_service>
      <enable_erasure>off</enable_erasure>
      <erasure_alg></erasure_alg>
      <erasure_fragmentation>none</erasure_fragmentation>
    </replica>
    <replica>
      <read_access>geographic</read_access>
      <enable_retention>on</enable_retention>
      <retention_year></retention_year>
      <retention_month></retention_month>
      <retention_day>4</retention_day>
      <retention_hour>4</retention_hour>
      <retention_minute></retention_minute>
      <retention_second></retention_second>
      <enable_deletion>on</enable_deletion>
      <deletion_year>2</deletion_year>
      <deletion_month></deletion_month>
      <deletion_day></deletion_day>
      <deletion_hour></deletion_hour>
    </replica>
  </replica_list>
</policy>
```

### List policy specifications

Returns an XML document that contains all of the policies for the specified tenant. The listing includes each policy's name, list of replicas (including replica type, storage mechanism, and storage location).

**Required role**

TenantAdmin

**HTTP method**

GET
REST API: Managing Policy Specifications

URI

/sysmgmt/<tenantName>/policies

Request parameters

- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the x-atmos-tenantadmin.
- `x-atmos-authtype` — Specify password.

Request header

GET /sysmgmt/t1/policies HTTP/1.1
accept: */*
date: Tue, 02 Mar 2010 07:52:45 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 168.159.121.223

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Tue, 02 Mar 2010 07:52:50 GMT
Set-Cookie: _gui_session_id=f5bc5e4e8eb3df92635d20f8a8a19cd9; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 229ms
ETag: "93551db9a9f1a6c0c0f22c9e653f8e43"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 483

Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<policy_list>
  <policy>
    <name>default</name>
    <replica_list>
      <replica>
        <type>sync</type>
        <storage_mechanism></storage_mechanism>
        <location></location>
      </replica>
      <replica>
        <type>sync</type>
        <storage_mechanism></storage_mechanism>
        <locations></locations>
        <location></location>
      </replica>
    </replica_list>
  </policy>
</policy_list>
```
Example: How to create a policy specification

This section describes how to create a policy specification. To create a policy, you need to decide:

- Where to put metadata
- How many replicas you want, and for each replica, you have to decide:
  - Is it sync or async?
  - Where should it be placed?
  - Does it have a special characteristics such as striping, GeoParity, or federation?
  - What kind of read access should be used?
  - Do I need a retention or deletion rule?

The HTTP body of the create request includes parameters for values that define your response to each of the questions above. The following examples show how to supply these values. For a reference to all values, see “Create or update a policy specification” on page 86.

Example — Two Replicas, No Data Transformation, Retention Enabled

The following example shows how to build a policy that defines 2 replicas (1 async, 1 sync). Retention is enabled, but no other data transformation properties are defined.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value/Description</th>
<th>Parameter to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>ExampleCustom2</td>
<td>perform_action=create other[specname]=ExampleCustom2 entry[spec_name]=&amp; new_spec=true replica_id=</td>
</tr>
<tr>
<td>Metadata</td>
<td>sameAs, $client</td>
<td>metadata[1][location_modifier]=sameAs metadata[1][location_place]=$client</td>
</tr>
<tr>
<td>Replica 1</td>
<td>Replica Type: async</td>
<td>spec[1][type]=async</td>
</tr>
<tr>
<td></td>
<td>Enable customize: no But you must still supply the default values for replica location and Storage Server attribute parameters.</td>
<td>spec[1][location1][modifier]=sameAs spec[1][location1][place]=ANY spec[1][ssattrs_placement]=OPTIMAL spec[1][ssattrs_actions]=ANY</td>
</tr>
</tbody>
</table>
Once you have the specification described, you can build the HTTP request as described next:

1. Use the HTTP POST verb with the following URI:

   `/sysmgmt/<tenantName>/policies`

2. Pass the following parameters in the HTTP header:

   - `x-atmos-tenantadmin` — Specify a valid username for someone in the TenantAdmin role.
   - `x-atmos-tenantadminpassword` — Specify the password for the x-atmos-tenantadmin.
   - `x-atmos-authtype` — Specify `password`.

3. Build the HTTP body specifying the different policy specification attributes like this:

   ```
   perform_action=create&other[specname]=ExampleCustom2&entry[spec_name]=
   &&new_spec=true&replica_id=&metadata[1][location]=&metadata[1][location_place]=
   &$client&spec[1][type]=async&spec[1][location Modifier]=sameAs&spec[1][location1][place]=
   &ANY&spec[1][ssattrs_placement]=OPTIMAL&spec[1][ssattrs actions]=ANY&other[read access]=
   geographic&other[enable_retention]=on&other[retention_delay_month]=1&other[retention_year]=5
   &other[enable_deletion]=on&other[deletion_year]=&other[deletion_month]=6
   &other[deletion_day]=&other[deletion_hour]=
   &operation_type=sync
   ```

### Table: Specifying Different Policy Specification Attributes

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value/Description</th>
<th>Parameter to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replica 2</td>
<td>Replica Type: sync</td>
<td><code>spec[2][type]=sync</code></td>
</tr>
<tr>
<td></td>
<td>Enable customize: no</td>
<td><code>spec[2][location][modifier]=sameAs</code></td>
</tr>
<tr>
<td></td>
<td>You must still supply the default values for replica location and Storage Server attribute parameters.</td>
<td><code>spec[2][location1][place]=ANY</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>spec[2][ssattrs_placement]=OPTIMAL</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>spec[2][ssattrs_actions]=ANY</code></td>
</tr>
<tr>
<td>Replica selection for read access</td>
<td>geographic</td>
<td><code>other[read_access]=geographic</code></td>
</tr>
<tr>
<td>Retention</td>
<td>Retention is enabled.</td>
<td><code>other[enable_retention]=on</code></td>
</tr>
<tr>
<td></td>
<td>A start delay window is required.</td>
<td><code>other[retention_delay_month]=1</code></td>
</tr>
<tr>
<td></td>
<td>Retention values are set.</td>
<td><code>other[retention_year]=5</code></td>
</tr>
<tr>
<td>Deletion</td>
<td>Deletion is enabled.</td>
<td><code>other[enable_deletion]=on</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>other[deletion_year]=</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>other[deletion_month]=6</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>other[deletion_day]=</code></td>
</tr>
<tr>
<td>Operation type</td>
<td>sync</td>
<td><code>operation_type=sync</code></td>
</tr>
</tbody>
</table>
REST API: Managing Policy Specifications
CHAPTER 9
REST API: System Monitoring

This chapter describes the REST APIs for monitoring an Atmos™ system.

- Web service statistics
- Atmos storage consumption
REST API: System Monitoring

**Web service statistics**

This operation returns an XML document that contains basic node-level web service statistics.

**Required role**

SysAdmin

**HTTP method**

GET

**URI**

/sysmgmt/rmgs/<rmgName>/<nodes>/<nodeName>/wsstats

**Request parameters**

- **x-atmos-systemadmin**: Specify the SysAdmin username. (Required.)
- **x-atmos-systemadminpassword**: Specify the SysAdmin password. (Required.)
- **x-atmos-authtype**: Specify password.

**Request header**

GET /sysmgmt/rmgs/Boston01/nodes/IS01-001/wsstats HTTP/1.1
accept: */*
date: Fri, 13 Nov 2009 09:26:47 GMT
x-atmos-systemadminpassword: password
x-atmos-systemadmin: SysAdmin1
x-atmos-authtype: password

**Request body**

None

**Response header**

HTTP/1.1 200 OK
Connection: close
Date: Tue, 27 Oct 2009 03:33:05 GMT
Set-Cookie: _gui_session_id=96e1e72e9e7df2d7f151cfd4e65ce61; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 587
Response body

```xml
<?xml version='1.0' encoding='UTF-8'?>
<WSStat>
  <Node name='is01-001'/>
  <ReadsPerSec>0.00</ReadsPerSec>
  <WritesPerSec>0.00</WritesPerSec>
  <DeletesPerSec>0.00</DeletesPerSec>
  <TransPerSec>0.00</TransPerSec>
  <MeanReadLatencyMS>6.723</MeanReadLatencyMS>
  <MeanWriteLatencyMS>214.721</MeanWriteLatencyMS>
  <MeanDeleteLatencyMS>0.000</MeanDeleteLatencyMS>
  <MeanLatencyMS>110.722</MeanLatencyMS>
  <Reads>3</Reads>
  <Writes>3</Writes>
  <Deletes>0</Deletes>
  <Total>6</Total>
  <ReadLatencyMS>20</ReadLatencyMS>
  <WriteLatencyMS>644</WriteLatencyMS>
  <DeleteLatencyMS>0</DeleteLatencyMS>
  <UptimeMS>108302784</UptimeMS>
</WSStat>
```

- **Reads**, **Writes**, and **Deletes** are defined as follows:

<table>
<thead>
<tr>
<th></th>
<th>REST</th>
<th>SOAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads</td>
<td>GET and HEAD transactions</td>
<td>List, Get, and Read transactions.</td>
</tr>
<tr>
<td>Writes</td>
<td>PUT and POST transactions</td>
<td>Create, Set, Update, and Version transactions.</td>
</tr>
<tr>
<td>Deletes</td>
<td>DELETE transactions</td>
<td>Delete transactions</td>
</tr>
</tbody>
</table>

- **Total** is the sum of reads, writes, and deletes.

- The per-second (**PerSec**) rate for each of the three transaction types is calculated as the count divided by the total uptime since the last server restart.

- **Mean latency** is the sum of the individual latencies reported at logging time, divided by the total number of transactions. There also are latencies calculated by operation type in a similar manner.

- Separately, the total accumulated latency is reported for each operation (**ReadLatencyMS**, **WriteLatencyMS**, and **DeleteLatencyMS**) and for the uptime (**UptimeLatencyMS**), in milliseconds. This allows you to sample at any desired frequency and calculate latencies for that interval.

Atmos storage consumption

Provides information about the number of objects and the total object size (object size and metadata size) for a specified tenant, subtenant, or UID.

Required role

TenantAdmin or SubtenantAdmin
REST API: System Monitoring

HTTP method

GET

URI for Tenant Metrics

/sysmgmt/tenants/<tenantName>/scMetrics

URI for Subtenant Metrics

/sysmgmt/tenants/<tenantName>/<subTenantName>/scMetrics

URI for UID Metrics

/sysmgmt/tenants/<tenantName>/<subTenantName>/<UID>/scMetrics

Request parameters

- x-atmos-tenantadmin: Specify the TenantAdmin username. (Required.)
- x-atmos-tenantadminpassword: Specify the TenantAdmin's password. (Required.)
- x-atmos-authtype: Specify password.

Request header for Tenant Metrics

GET /sysmgmt/tenants/Tenant1/scMetrics HTTP/1.1
accept: */*
date: Thu, 12 Nov 2009 10:04:11 GMT
x-atmos-tenantadmin: Tenant1Admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Thu, 12 Nov 2009 10:04:15 GMT
Set-Cookie: _gui_session_id=2e336848d437fdaf9704b3157aeaff3; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
Cache-Control: no-cache
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 203
### Response body for Tenant Metrics

```xml
<?xml version='1.0' encoding='UTF-8'?>
<scMetrics>
  <objCount>5</objCount>
  <size>9845</size>
  <realsize>3306</realsize>
  <metadatasize>10434</metadatasize>
  <totalSize>13740</totalSize>
</scMetrics>
```

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objCount</td>
<td>The number of objects stored for the specified tenant, subtenant, or UID.</td>
</tr>
<tr>
<td>size</td>
<td>The object size (not including replicas or metadata) for the specified tenant, subtenant, or UID. The default size for a directory is 4k.</td>
</tr>
<tr>
<td>realsize</td>
<td>Derived from the size element multiplied by the number of replicas defined for the policy. Replica counts for directories are 0.</td>
</tr>
<tr>
<td>metadatasize</td>
<td>The sum of the size of the XML description for user metadata, system metadata, policy description, LSO layout, and so on.</td>
</tr>
<tr>
<td>totalsize</td>
<td>The sum of the realsize and the metadatasize elements.</td>
</tr>
</tbody>
</table>
This part includes the following chapters:

- **“POX API: Authenticating Administrators” on page 111**
  Describes the POX authentication methods.

- **“POX API: Managing Tenants” on page 117**
  Describes the POX API for managing tenants.

- **“POX API: Managing Subtenants” on page 131**
  Describes the POX API for managing subtenants.

- **“POX API: Managing NFS Shares” on page 143**
  Describes the POX API for adding NFS access nodes and for creating and managing NFS shares.

- **“POX API: Listing File Systems” on page 157**
  Describes the API that lists the file system capacity for each node.

- **“POX API: Managing UIDs and Shared Secrets” on page 161**
  Describes the POX API for managing UIDs.

- **“POX API: Managing Policy Selectors” on page 171**
  Describes the POX API for creating and managing policy selectors.

- **“POX API: Managing Policy Specifications” on page 191**
  Describes the POX API for creating and managing policy specifications.
POX API system management operations require an authenticated session. The user who authenticates must be in appropriate administrative role for the requested operation.

- Authenticating as a SysAdmin ................................................................. 112
- Authenticating as a TenantAdmin or a SubtenantAdmin ............................. 114
POX API: Authenticating Administrators

**Authenticating as a SysAdmin**

Authenticates a user in the Atmos SysAdmin role. Returns a session ID (_gui_session_id) in the HTTP response header **Set-Cookie**. All subsequent requests using this API must include this session ID in the HTTP request header **Cookie**.

There is no logout request. An inactive session expires after 30 minutes.

**HTTP method**

POST

**URI**

/mgmt_login/verify

**Request parameters**

HTTP Body:

- **auth_type**: Values are **local** or **remote**.
- **auth_addr**: When **auth_type = remote**, specify the IP address/host name of the remote location where the specified SysAdmin user is authenticating from.
- **username**: Specify the username of a SysAdmin user.
- **password**: Specify the password for the SysAdmin user.

**Request header**

POST /mgmt_login/verify HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Host: 10.5.116.110
Content-Length: 64

**Request body**

auth_type=local&auth_addr=&username=SysAdmin&password=\%231Passwd

**Response header**

HTTP/1.1 200 OK
Date: Fri, 11 Sep 2009 13:22:52 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 36
Set-Cookie: _gui_session_id=113fa69bb6173fa6016a75bbdc3fc06a; path=/
Connection: close

**Response body**

<authenticated>true</authenticated>

or
<authenticated>Authentication failed.</authenticated>
Authenticating as a TenantAdmin or a SubtenantAdmin

Authenticates an Atmos tenant administrator or subtenant administrator. Returns a session ID (_gui_session_id) in the HTTP response header Set-Cookie. All subsequent requests using this API must include this session ID in the HTTP request header Cookie.

There is no logout request. An inactive session expires after 30 minutes.

HTTP method

POST

URI

/user/verify

Request parameters

HTTP body:

- **tenant_name**: Specify the name of an existing Atmos tenant.
- **sub_tenant_name**: Specify the name of an existing Atmos subtenant when you want to authenticate as a subtenant administrator.
- **username**: Specify the TenantAdmin username, or the SubTenantAdmin username (to authenticate as a subtenant administrator).
- **password**: Specify the password associated with the username.

Request header

POST /user/verify HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Host: 10.5.116.110
Content-Length: 68

Request body

To authenticate as a tenant administrator:

tenant_name=maas_tenant&username=maas_tenant_admin&password=password

To authenticate as a subtenant administrator:

tenant_name=maas_tenant&sub_tenant_name=sub1&username=maas_tenant_admin&password=password
Response header

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 17:31:07 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 36
Set-Cookie: _gui_session_id=1f856413a34c20aba0ddb5dc0206fa35;
path=/
Connection: close

Response body

A successful login returns:

<authenticated>true</authenticated>

An unsuccessful login returns:

<authenticated>Authentication failed.</authenticated>
POX API: Authenticating Administrators
This chapter describes the POX API for managing tenants. It includes the following operations:

- Assign a TenantAdmin
- Create a tenant
- Get tenant details
- List tenants
- Remove a TenantAdmin
- Examples: Obtaining tenant information

These operations require an authenticated session. To create an authenticated session, see “Authenticating as a SysAdmin” on page 112.
POX API: Managing Tenants

Assign a TenantAdmin

Adds a user to the TenantAdmin role for the specified tenant. The TenantAdmin role is responsible for:

- Managing subtenants.
- Managing policies for the tenant to which they are assigned.
- Registering applications (by generating a UID/Shared Secret) and adding policies for the corresponding applications.

A tenant can have multiple TenantAdmins.
A TenantAdmin can be a TenantAdmin for only one tenant.
A user can be a TenantAdmin for multiple tenants.

Required role

SysAdmin

HTTP Method

GET

URI

/maui_admin/submit_tenant_admin_info

Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.
- **Accept**: Set to application/xml.

Querystring parameters:

- **operation**: Specify add.
- **flag_new**: Specify on (if user is new) or off (if user already exists).
- **tenant_name**: Specify the name of the tenant.
- **username**: Specify the name of the user to be assigned to the TenantAdmin role.
- **password**: Specify the password for the new user. This is only valid when **flag_new** is set to on.

Request header

GET

/maui_admin/submit_tenant_admin_info?operation=add&flag_new=off&tenant_name=TestTenant&username=TestAdmin&password= HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=5cfff6a5144b259dee629d1d204d3aceb
Host: 10.5.116.110
Response header

HTTP/1.1 200 OK
Date: Fri, 11 Sep 2009 13:22:38 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=5cff6a5144b259dee629d1d204d3aceb; path=/
Connection: close

Response body

A successful response looks like this: <cleared>true</cleared>

Create a tenant

Creates an Atmos tenant.

Required role

SysAdmin

HTTP method

POST

URI

/maui_admin/submit_add_tenant

Request parameters

HTTP header:

• Cookie: Set to the _gui_session_id returned by the authentication method.

• Accept: Set to application/xml.

HTTP body:

• auth_type: Specify local. Specifies how the tenant is authenticated.

• tenant_name: Specify a string representing the new tenant. It must be unique, and 30 characters or less.

• creation_type: Specify sync.

Request header

POST /maui_admin/add_tenant HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=fa6818a982d7e35a8ae6c9f8b639e82
Host: 10.5.116.110
Content-Length: 61
POX API: Managing Tenants

Request body

```
auth_type=local&tenant_name=testtenant&creation_type=sync
```

Response header

```
HTTP/1.1 200 OK
Date: Thu, 10 Sep 2009 13:29:11 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 56
Set-Cookie: _gui_session_id=fa6818a982d7e35a8aee6c9f8b639e82; path=/
Connection: close
```

Response body

```
<tenant_id>8bdb8ce67eaf4a4f946d24cce76c9974</tenant_id>
```

Get tenant details

Returns an XML document containing information about the tenant associated with the TenantAdmin making the request. The information includes:

- Tenant name
- List of TenantAdmins
- List of subtenants and subtenant admins
- List of Web Services access nodes
- List of CIFS and NFS access nodes
- Policies

Required role

TenantAdmin

HTTP Method

GET

URI

```
/tenant_admin/get_tenant_info
```

Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.
- **Accept**: Set to application/xml.
Request header

GET /tenant_admin/get_tenant_info HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=2948ab711e7c190d37ebc201d2bc2169
Host: 10.5.116.110

Response header

HTTP/1.1 200 OK
Date: Fri, 11 Sep 2009 11:58:23 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 1514
Set-Cookie: _gui_session_id=2948ab711e7c190d37ebc201d2bc2169;
    path=/
Connection: close
Response body

```xml
<tenant>
  <id>478e1c71b39843dabfada070b21c851a</id>
  <name>TestTenant</name>
  <status>Initialized</status>
  <authentication_source>Local</authentication_source>
  <tenant_admin_list>
    <tenant_admin>
      <id>478e1c71b39843dabfada070b21c851a</id>
      <name>TestAdmin</name>
      <authentication_source>Local</authentication_source>
    </tenant_admin>
  </tenant_admin_list>
  <access_node_list>
    <access_node>
      <id>564D3161-E062-BDD7-869F-296382B5F712</id>
      <name>Testtest1-002</name>
      <public_ip>10.5.116.111</public_ip>
      <status>Service up</status>
    </access_node>
    <access_node>
      <id>564DE0F3-A5BF-9DFD-8C5B-5EA182150521</id>
      <name>Testtest1-001</name>
      <public_ip>10.5.116.110</public_ip>
      <status>Service up</status>
    </access_node>
  </access_node_list>
  <nfs_cifs_node_list>
    <nfs_cifs_node>
      <id>564D4F54-A020-FAB6-51C5-95E59E8D971E</id>
      <name>Testtest2-002</name>
      <type>nfs</type>
      <public_ip>10.5.116.112</public_ip>
      <status>Service up</status>
    </nfs_cifs_node>
    <nfs_cifs_node>
      <id>564DC4E7-B0F3-D0E7-BD28-06FFDC4B4D71</id>
      <name>Testtest2-001</name>
      <type>cifs</type>
      <public_ip>10.5.116.113</public_ip>
      <status>Service up</status>
    </nfs_cifs_node>
  </nfs_cifs_node_list>
  <capacity>0</capacity>
  <sub_tenant_list>
    <sub_tenant>
      <id>478e1c71b39843dabfada070b21c851a</id>
      <name>TestTenant</name>
      <status>Initialized</status>
      <authentication_source>Local</authentication_source>
      <sub_tenant_admin_list>
        <sub_tenant_admin>
          <name>TestSubAdmin</name>
        </sub_tenant_admin>
      </sub_tenant_admin_list>
    </sub_tenant>
  </sub_tenant_list>
</tenant>
```
List tenants

Returns an XML document that contains general information about the tenants for the SysAdmin making the request. The XML document contains the following information:

- Tenant names and UUIDs
- Tenant status
- TenantAdmin lists
- Web service access nodes assigned to each tenant.

Returns an empty XML document if there are no tenants.
POX API: Managing Tenants

Required role

SysAdmin

HTTP Method

GET

URI

/maui_admin/list_tenant

Request parameters

HTTP header:

• **Cookie**: Set to the _gui_session_id returned by the authentication method.
• **Accept**: Set to application/xml.

Request header

GET /maui_admin/list_tenant HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=57a4b0e5d01fb7c7a2e18a3711def36a
Host: 10.5.116.110

Response header

HTTP/1.1 200 OK
Date: Fri, 11 Sep 2009 11:11:30 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 825
Set-Cookie: _gui_session_id=57a4b0e5d01fb7c7a2e18a3711def36a; path=/
Connection: close
Response body

```
<tenant_list>
  <tenant>
    <name>TestTenant2</name>
    <id>e60d9f18fc9a4f0d8bab77cf67e57e55</id>
    <status>Initialized</status>
    <authentication_source>Local</authentication_source>
    <tenant_admin_list>
      <tenant_admin>No Tenant Admin</tenant_admin>
    </tenant_admin_list>
    <access_node_list>
      <access_node>No access node</access_node>
    </access_node_list>
  </tenant>
  <tenant>
    <name>TestTenant3</name>
    <id>8bdb8ce67eaf4a4f946d24cce76c9974</id>
    <status>Initialized</status>
    <authentication_source>Local</authentication_source>
    <tenant_admin_list>
      <tenant_admin>No Tenant Admin</tenant_admin>
    </tenant_admin_list>
    <access_node_list>
      <access_node>No access node</access_node>
    </access_node_list>
  </tenant>
</tenant_list>
```

Remove a TenantAdmin

Removes the specified user from the TenantAdmin role for the specified tenant. If successful, returns `<cleared>true</cleared>`.

Required role

SysAdmin

HTP Method

GET

URI

/maui_admin/modify_tenant_admin

Request parameters

HTTP header:

- **Cookie**: Set to the `_gui_session_id` returned by the authentication method.
- **Accept**: Set to `application/xml`.

Querystring parameters:

- **operation**: Specify the operation type to run; for this request, the value is always `delete`.
- **tenant_name**: Specify the tenant name for whom the user is in the TenantAdmin role.
• **username**: Specify the name of the user to be removed as a TenantAdmin of the specified tenant.

### Request header

```
GET /maui_admin/modify_tenant_admin?operation=delete&tenant_name=TestTenant&username=TestAdmin HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=113fa69bb6173fa6016a75bbdc3fc06a
Host: 10.5.116.110
```

### Response header

```
HTTP/1.1 200 OK
Date: Fri, 11 Sep 2009 13:22:52 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=113fa69bb6173fa6016a75bbdc3fc06a; path=/
Connection: close
```

### Response body

```
<cleared>true</cleared>
```

### Examples: Obtaining tenant information

This section describes the set of commands you use for:

- “Obtaining the List of Tenants and Access Nodes” on page 126.
- “Obtaining Information about One Tenant” on page 127.

### Obtaining the List of Tenants and Access Nodes

To obtain the details about the tenants for the authenticated user in the SysAdmin role:

1. Authenticate as an Atmos SysAdmin. For more information, see Chapter 10, “POX API: Authenticating Administrators.”

2. Obtain and use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter. You set **Cookie** to the `_gui_session_id` returned by the authentication method.

3. Use the HTTP GET method with this URI:

   `/maui_admin/list_tenant`

The following body shows an example of what is returned:

```
<tenant_list>
  <tenant>
    <name>t1</name>
    <id>6387df4a05e14bde1a20e410d2ebcda</id>
    <status>Initialized</status>
    <authentication_source>Local</authentication_source>
  </tenant_admin_list>
```
Obtaining Information about One Tenant

To obtain the details about the tenants for the authenticated user in the SysAdmin role:

1. **Authenticate** as an Atmos SysAdmin. For more information, see Chapter 10, “POX API: Authenticating Administrators,“.

2. **Obtain** and use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter. You set **Cookie** to the _gui_session_id returned by the authentication method.

3. **Use** the HTTP GET method with this URI:

   /tenant_admin/get_tenant_info

The following body shows an example of what is returned:

```xml
<tenant>
  <id>6387df4a05e14bdeala20e410d2ebcda</id>
  <name>t1</name>
  <status>Initialized</status>
  <authentication_source>Local</authentication_source>
  <tenant_admin_list>
    <tenant_admin>
      <name>t1admin</name>
      <authentication_source>Local</authentication_source>
    </tenant_admin>
    <tenant_admin>
      <name>test</name>
      <authentication_source>Local</authentication_source>
    </tenant_admin>
  </tenant_admin_list>
  <policy_distribution_status>Completed</policy_distribution_status>
  <access_node_list>
    <access_node>IS01-002</access_node>
    <access_node>IS01-003</access_node>
    <access_node>IS01-001</access_node>
    <access_node>IS01-004</access_node>
  </access_node_list>
</tenant>
```

**Examples:** Obtaining tenant information 127
POX API: Managing Tenants

">&lt;multi_subtenant_access&gt;disable&lt;/multi_subtenant_access&gt;
&lt;/access_node&gt;
&lt;access_node&gt;
&lt;name&gt;IS01-003&lt;/name&gt;
&lt;id&gt;564D188C-F514-7901-39E3-63B2CDCA527A&lt;/id&gt;
&lt;public_ip&gt;10.5.116.246&lt;/public_ip&gt;
&lt;webservice&gt;enable&lt;/webservice&gt;
&lt;filesystem&gt;nfs&lt;/filesystem&gt;
&lt;multi_subtenant_access&gt;disable&lt;/multi_subtenant_access&gt;
&lt;/access_node&gt;
&lt;access_node&gt;
&lt;name&gt;IS01-001&lt;/name&gt;
&lt;id&gt;564D85B4-8FA2-34BF-24A4-9BFB9C17A02B&lt;/id&gt;
&lt;public_ip&gt;10.5.116.244&lt;/public_ip&gt;
&lt;webservice&gt;disable&lt;/webservice&gt;
&lt;filesystem&gt;nfs&lt;/filesystem&gt;
&lt;multi_subtenant_access&gt;disable&lt;/multi_subtenant_access&gt;
&lt;/access_node&gt;
&lt;access_node&gt;
&lt;name&gt;IS01-004&lt;/name&gt;
&lt;id&gt;564DF58B-B8B3-EC98-7F8B-E23B098FC976&lt;/id&gt;
&lt;public_ip&gt;10.5.116.247&lt;/public_ip&gt;
&lt;webservice&gt;enable&lt;/webservice&gt;
&lt;filesystem&gt;nfs&lt;/filesystem&gt;
&lt;multi_subtenant_access&gt;disable&lt;/multi_subtenant_access&gt;
&lt;/access_node&gt;
&lt;/access_node_list&gt;
&lt;capacity&gt;0&lt;/capacity&gt;
&lt;sub_tenant_list&gt;
&lt;sub_tenant&gt;
&lt;name&gt;t1&lt;/name&gt;
&lt;id&gt;6387df4a05e14bdea1a20e410d2ebcda&lt;/id&gt;
&lt;status&gt;Initialized&lt;/status&gt;
&lt;authentication_source&gt;Local&lt;/authentication_source&gt;
&lt;sub_tenant_admin_list&gt;
&lt;sub_tenant_admin&gt;t1admin&lt;/sub_tenant_admin&gt;
&lt;/sub_tenant_admin_list&gt;
&lt;/sub_tenant&gt;
&lt;sub_tenant&gt;
&lt;name&gt;t1admin&lt;/name&gt;
&lt;id&gt;ad33e4c978c74dfbb31d5c6363b13ec4&lt;/id&gt;
&lt;status&gt;Initialized&lt;/status&gt;
&lt;authentication_source&gt;Local&lt;/authentication_source&gt;
&lt;sub_tenant_admin_list&gt;
&lt;sub_tenant_admin&gt;No SubTenant Admin&lt;/sub_tenant_admin&gt;
&lt;/sub_tenant_admin_list&gt;
&lt;/sub_tenant&gt;
&lt;/sub_tenant_list&gt;
&lt;policy_list&gt;
&lt;policy&gt;
&lt;name&gt;default&lt;/name&gt;
&lt;expression&gt;&lt;/expression&gt;
&lt;policy_id&gt;0&lt;/policy_id&gt;
&lt;replica_list&gt;
&lt;replica&gt;
&lt;type&gt;sync&lt;/type&gt;
&lt;storage_mechanism&gt;&lt;/storage_mechanism&gt;
&lt;location&gt;&lt;/location&gt;
&lt;/replica&gt;
&lt;replica&gt;
&lt;type&gt;sync&lt;/type&gt;
&lt;storage_mechanism&gt;&lt;/storage_mechanism&gt;
&lt;location&gt;&lt;/location&gt;
&lt;/replica&gt;
&lt;/replica_list&gt;
&lt;retention&gt;&lt;/retention&gt;
<deletion/></deletion>
</policy>
</policy_list>
<policy_selector_list>
<policy_selector>
<name>policyselctor1</name>
<expression>atime &gt;= 2010-01-08 00:00:00</expression>
<on_event>ON_CREATE</on_event>
<spec>default</spec>
</policy_selector>
<policy_selector>
<name>policyselctor3</name>
<expression>itime equals 2010-01-25</expression>
<on_event>ON_CREATE</on_event>
<spec>Policy4</spec>
</policy_selector>
</policy_selector_list>
</tenant>
POX API: Managing Tenants
CHAPTER 12
POX API: Managing Subtenants

This chapter describes the API for managing subtenants, UIDs, and shared secrets. It includes the following topics:

- Assign a SubtenantAdmin ................................................................. 132
- Create a subtenant ........................................................................... 133
- Read subtenant .................................................................................. 134
- List subtenants .................................................................................. 136
- Modify a subtenant ........................................................................... 138
- Remove a SubtenantAdmin ................................................................. 139
- Example: How to obtain subtenant information ................................. 140

These operations require an authenticated session. To create an authenticated session, see “Authenticating as a SysAdmin” on page 112.
Assign a SubtenantAdmin

Assigns the specified user to the SubtenantAdmin role for the specified subtenant. SubTenantAdmins can:

- Assign users to a subtenant.
- Reorder policies (only available via the system management GUI).

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/submit_sub_tenant_admin_info

Request parameters

HTTP header:
- **Cookie**: Set to the _gui_session_id returned by the authentication method.
- **Accept**: Set to application/xml.

HTTP body:
- **operation**: Specify the operation type; for this request, the value is always add.
- **flag_new**: Specify if the user is new (value is on) or not (value is off).
- **username**: Specify the name of the user to be assigned as the SubTenantAdmin for the subtenant.
- **sub_tenant_name**: Specify the name of the subtenant.

Request header

POST /tenant_admin/submit_sub_tenant_admin_info HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=c718f1f27ad152016c024787d4e07a80
Host: 10.5.116.110
Content-Length: 77

Request body

operation=add&flag_new=off&username=TestUser&sub_tenant_name=TestTenant

Response header

HTTP/1.1 200 OK
Date: Mon, 14 Sep 2009 19:07:41 GMT
Response body

A successful request returns the following response:

```xml
<cleared>true</cleared>
```

Some error responses include:

```xml
<cleared>Error: No such user.</cleared>(the flag_new is off but username is non-exist)
```

```xml
or
```

```xml
<cleared>No such subtenant.</cleared>(the sub_tenant_name is non-exist)
```

Create a subtenant

Creates an Atmos subtenant within the tenant of the requesting tenant administrator.

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/add_sub_tenant

Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP body:

- **auth_type**: Specify local.
- **sub_tenant_name**: Specify a string representing the new subtenant.

The name must be unique within the tenant. It can be up to 255 characters, and can include: alpha characters a to z (both upper and lower case), digits 0 to 9, or the following special characters: - _ @ $ ^ , ; | [ ] ( ).

The name cannot be the same as the UUID pattern (32 characters long with only the characters a to f (both upper and lower case) and the digits 0-9).
Note: The POX API does not validate this parameter. Follow these naming rules to ensure data consistency.

- **creation_type**: Specify `sync`.

## Request header

```
POST /tenant_admin/add_sub_tenant HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=1f856413a34c20aba0dd5dc0206fa35
Host: 10.5.116.110
Content-Length: 57
```

## Request body

```
auth_type=local&sub_tenant_name=samplename&creation_type=sync
```

## Response header

```
HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 17:31:07 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 64
Set-Cookie: _gui_session_id=1f856413a34c20aba0dd5dc0206fa35; path=/
Connection: close
```

## Response body

```
A successful request returns:

<sub_tenant_id>747015be2d454ccebcc81853ac7c465a</sub_tenant_id>

Unsuccessful requests return error messages, for example:

<sub_tenant_id>Error: Subtenant samplename existing</sub_tenant_id>

or

<sub_tenant_id>Error: Subtenant samplename already exists.</sub_tenant_id>

## Read subtenant

Returns an XML document containing the following information about the specified subtenant:

- **ID**
- **Name**
- **Authentication source**
- **Status**
- **Subtenant administrators**
POX API: Managing Subtenants

- Capacity
- Default policy specification
- Shared secrets for the subtenant’s UIDs
- The export list

**Note:** Capacity always returns 0.

**Required role**

TenantAdmin

**HTTP method**

GET

**URI**

/tenant_admin/sub_tenant_info

**Request parameters**

HTTP header:
- **Cookie**: Set to the _gui_session_id returned by the authentication method.

Querystring parameter:
- **sub_tenant_name**: Specify the name of the subtenant whose details you want.

**Request header**

GET /tenant_admin/sub_tenant_info?sub_tenant_name=samplename HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=6807f31b8689a4bbf8a13110f7ba4f82
Host: 10.5.116.110

**Request body**

None

**Response header**

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 18:24:46 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 2086
Set-Cookie: _gui_session_id=6807f31b8689a4bbf8a13110f7ba4f82; path=/
Connection: close
Response body

```xml
<sub_tenant>
  <id>29077709482d40978697f40905342a4d</id>
  <name>st1</name>
  <authentication_source>Local</authentication_source>
  <status>Initialized</status>
  <sub_tenant_admin_list>
    <sub_tenant_admin>
      <name>st1admin</name>
      <authentication_source>Local</authentication_source>
    </sub_tenant_admin>
    <sub_tenant_admin>
      <name>t1admin</name>
      <authentication_source>Local</authentication_source>
    </sub_tenant_admin>
  </sub_tenant_admin_list>
  <capacity>0</capacity>
  <default_policy_spec>default</default_policy_spec>
  <uid_secret_list>
    <uid_secret>
      <uid>uid1</uid>
      <shared_secret>unmQ9cQsUb2jgPiY5ZPfcs4mtpU=</shared_secret>
      <status>enabled</status>
      <email>a@b.com</email>
    </uid_secret>
    <uid_secret>
      <uid>uid2</uid>
      <shared_secret>7YrrZ8snhwoXrkuT5aPzHHOnKlg=</shared_secret>
      <status>enabled</status>
      <email>a@b.com</email>
    </uid_secret>
  </uid_secret_list>
  <policy_selector_map_status>Completed</policy_selector_map_status>
  <policy_selector_list>
    <handler_list>
      <export_list>
        <sub_tenant>
        </sub_tenant>
      </export_list>
      <handler_list>
        <export_list>
          <sub_tenant>
          </sub_tenant>
        </export_list>
      </handler_list>
    </policy_selector_list>
  </policy_selector_list>
</sub_tenant>
```

List subtenants

Returns an XML document that contains general information about all subtenants associated with the requesting tenant administrator. The XML document contains the following information about one or more subtenants:

- Subtenant name
- ID
- Authentication source
- Status
- Subtenant administrator name

The HTTP request header must include the Cookie with the session ID (_gui_session_id) returned from a successful authentication.
List subtenants

POX API: Managing Subtenants

Required role

TenantAdmin

HTTP method

GET

URI

/tenant_admin/list_sub_tenant

Request parameters

HTTP header:

- **Cookie**: Set to the `_gui_session_id` returned by the authentication method.

Request header

GET /tenant_admin/list_sub_tenant HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=3cbab89e4a42a999f6f020a8e8e3a82a
Host: 10.5.116.110

Request body

None

Response header

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 17:37:29 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 2166
Set-Cookie: _gui_session_id=3cbab89e4a42a999f6f020a8e8e3a82a; path=/
Connection: close

Response body

A successful request returns:
Some examples of unsuccessful responses include:

<pre>&lt;cleared&gt;No such subtenant: TestTenant&lt;/cleared&gt;
</pre>
or

<pre>&lt;cleared&gt;No such user: st3admin.&lt;/cleared&gt;
</pre>

## Modify a subtenant

Lets you modify the default policy specification for the specified subtenant.

### Required role

TenantAdmin

### HTTP method

POST

### URI

/tenant_admin/submit_change_default_spec

### Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.
- **Accept**: Set to `application/xml`.
HTTP body:

- **default_spec_name**: Specify the name of the policy specification to set as the new default for the specified subtenant.
- **sub_tenant_name**: Specify the name of the subtenant to modify.

**Request header**

```plaintext
POST /tenant_admin/submit_change_default_spec HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=df11382b45e740710b4a163260fcc407
Host: 10.5.116.110
Content-Length: 56
```

**Request body**

```plaintext
default_spec_name=testSpec&sub_tenant_name=TestTenant
```

**Response header**

HTTP/1.1 200 OK
Date: Mon, 14 Sep 2009 18:26:26 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=df11382b45e740710b4a163260fcc407; path=/
Connection: close

**Response body**

A successful request returns:

```xml
<cleared>true</cleared>
```

**Remove a SubtenantAdmin**

Removes the specified user from the SubtenantAdmin role.

**Required role**

TenantAdmin

**HTTP method**

POST

**URI**

```plaintext
/tenant_admin/modify_sub_tenant_admin
```

**Request parameters**

HTTP header:
POX API: Managing Subtenants

- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP body:

- **operation**: Specify the operation type; for this request, the value is always `delete`.
- **sub_tenant_name**: Specify the name of the subtenant.
- **username**: Specify the name of the user to be removed from the SubTenantAdmin role for the specified subtenant.

Request header

```
POST /tenant_admin/modify_sub_tenant_admin HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=559f6115e5ed705e0b40157116d42301
Host: 10.5.116.110
Content-Length: 67
```

Request body

```
operation=delete&sub_tenant_name=TestTenant&username=TestTemp
```

Response header

```
HTTP/1.1 200 OK
Date: Mon, 14 Sep 2009 19:12:22 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=559f6115e5ed705e0b40157116d42301; path=/
Connection: close
```

Response body

```
A successful request returns:

<cleared>true</cleared>
```

Example: How to obtain subtenant information

This section describes the set of commands you use for:

- “Obtaining the List of Subtenants”
- “Obtaining Information about One Subtenant”

Obtaining the List of Subtenants

To obtain the list of subtenants for a TenantAdmin:

1. Authenticate as an Atmos TenantAdmin. For more information, see Chapter 10, “POX API: Authenticating Administrators.”
2. Obtain and use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter. You set Cookie to the _gui_session_id returned by the authentication method.

3. Use the HTTP GET method with this URI:

```
/tenant_admin/list_sub_tenant
```

The following body shows an example of what is returned:

```
<sub_tenant_list>
  <sub_tenant>
    <name>t1</name>
    <id>6387df4a05e14bdeala20e410d2ebcda</id>
    <authentication_source>Local</authentication_source>
    <status>Initialized</status>
    <sub_tenant_admin_list>
      <sub_tenant_admin>t1admin</sub_tenant_admin>
    </sub_tenant_admin_list>
  </sub_tenant>
  <sub_tenant>
    <name>t1admin</name>
    <id>ad33e4c978c74dfbb31d5c6363b13ec4</id>
    <authentication_source>Local</authentication_source>
    <status>Initialized</status>
    <sub_tenant_admin_list>
      <sub_tenant_admin>No Admin</sub_tenant_admin>
    </sub_tenant_admin_list>
  </sub_tenant>
</sub_tenant_list>
```

**Obtaining Information about One Subtenant**

1. Authenticate as an Atmos TenantAdmin. For more information, see Chapter 10, “POX API: Authenticating Administrators.”.

2. Obtain and use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter. You set Cookie to the _gui_session_id returned by the authentication method.

3. Use the HTTP GET method with this URI:

```
/tenant_admin/sub_tenant_info?sub_tenant_name=t1
```

The following body shows an example of what is returned:

```
<sub_tenant>
  <id>6387df4a05e14bdeala20e410d2ebcda</id>
  <name>t1</name>
  <authentication_source>Local</authentication_source>
  <status>Initialized</status>
  <sub_tenant_admin_list>
    <sub_tenant_admin>
      <name>t1admin</name>
      <authentication_source>Local</authentication_source>
    </sub_tenant_admin>
  </sub_tenant_admin_list>
  <capacity>0</capacity>
  <default_policy_spec>default</default_policy_spec>
  <uid_secret_list>
    <uid_secret>
      <uid>doctest</uid>
      <shared_secret>mcejGKBsf7dGy17vssZcSvR+Kgw=/</shared_secret>
      <status>enabled</status>
    </sub_tenant>
```
<uid_secret>
<uid>test1</uid>
<shared_secret>7Sp4bZxPp49GCtf3q5xHk18fNJM=</shared_secret>
<status>enabled</status>
<email>test1@emc.com</email>
</uid_secret>
</uid_secret_list>
<policy_selector_map_status>Completed</policy_selector_map_status>
<policy_selector_list>
<policy_selector>
<name>policyselector4</name>
<expression>itime equals 2010-01-25</expression>
<on_event>ON_CREATE</on_event>
<spec>Policy3</spec>
</policy_selector>
<policy_selector>
<name>policyselector3</name>
<expression>itime equals 2010-01-25</expression>
<on_event>ON_CREATE</on_event>
<spec>Policy4</spec>
</policy_selector>
</policy_selector_list>
<handler_list/>
<export_list/>
</sub_tenant>
CHAPTER 14
POX API: Managing NFS Shares

This chapter describes the POX API for configuring NFS access nodes and NFS shares.

- Assign NFS access node to a tenant ................................................................. 144
- Create NFS share ............................................................................................ 145
- Change NFS shares ......................................................................................... 148
- Delete NFS share ............................................................................................. 150
- List NFS shares ............................................................................................... 151
- Unassign NFS access node ............................................................................ 152
- Translate node name to UUID ......................................................................... 153
- Examples: Working with NFS shares ................................................................. 154

These API calls require an authenticated session. To create an authenticated session, see “Authenticating as a SysAdmin” on page 112.
**Assign NFS access node to a tenant**

Assigns a node as an NFS access node for the specified tenant. An NFS access node is an Atmos node that acts as an entry point for requests from NFS clients.

An access node can be assigned as both a Web-service access node and either an NFS or CIFS access node. An Atmos access node belongs to a single tenant.

To change the assignment of a node from CIFS to NFS, use the “Unassign NFS access node”, then reassign it using the “Assign CIFS access node to a tenant” call.

---

**Note:** Before you configure NFS or CIFS access to Atmos, synchronize your NFS or CIFS client machines with your Atmos NTP server time (see “Configuring NTP (Network Time Protocol)” in the *EMC Atmos System Administrator’s Guide*).

---

**Required role**

SysAdmin

**HTTP method**

POST

**URI**

/maui_admin/submit_assign_tenant_node

**Request parameters**

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP Body:

- **tenant_name**: Specify the name of the tenant the node will be assigned to.
- **nodes[node_name]=on**: Replace node_name with the name of the node. Only one node name can be specified per call.
- **fs[node_name]=nfs**: Specify the same node_name as in the nodes parameter. This means the node will be an NFS node.
- **multi_subtenant_access[node_name]=on**: (Optional.) Specify this parameter when you want all of a tenant’s subtenants to have access to the namespace through the specified node.

When not specified, only the default subtenant for the tenant has access. It also determines the mount path for any shares that you add.

If multi_subtenant_access is not enabled, all shares are created under the mauifs directory. If it is enabled, all shares are created under the mauifs/subtenantuuid directory. After you have created shares, do not change this value because if you do, any shares that you created will no longer be accessible.
Request header

POST /maui_admin/submit_assign_tenant_node HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=2267c906b87b35b46cef910be1790599
Host: 10.5.116.244
Content-Length: 142

Request body

tenant_name=t1&nodes[Shanghai-001]=on&fs[Shanghai-001]=nfs

Response header

HTTP/1.1 200 OK
Connection: close
Date: Wed, 20 Jan 2010 09:49:26 GMT
Set-Cookie: _gui_session_id=2267c906b87b35b46cef910be1790599;
path=/
Status: 200 OK
X-Runtime: 3358ms
ETag: "27430fa0a0a46a8af142853e782c1a44"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 24

Response body

A successful request returns:

<cleared>true</cleared>

A failed request returns a message like:

<cleared>Node Shanghai-001 has already be assigned.</cleared>

Create NFS share

Adds an NFS share to an existing NFS access node.

You can define the same share on multiple access nodes, but you should follow these guidelines if you do:

- Perform write operations on a single one.
- You can perform read operations to the same share on multiple access nodes.

Required role

TenantAdmin or SubTenantAdmin

HTTP method

POST
URI
/sub_tenant_admin/submit_add_node_nfs

Request parameters

HTTP header:
• **Cookie**: Set to the `_gui_session_id` returned by the authentication method.

HTTP Body:
• **node_uuid**: The UUID of the NFS access node where you want to add the NFS share. The node must first be specified as an NFS access node for the tenant the TenantAdmin is managing. For more information, see “Assign NFS access node to a tenant” For information about how to obtain the UUID for a node, see “Translate node name to UUID” on page 153.

• **subtenant_name**: Specify the name of the subtenant that can use the NFS share.

• **share_path**: Specify the relative path of the directory under `/mnt/mauifs/`. Cannot include these characters: & < > ' or `. This field cannot be greater than 256 characters.

**Note:** This field does not support Unicode characters.

The resulting mount path depends on whether `multi_subtenant_access` is enabled or not. For example, if you specify `share_path` as `nfsdir`, and `multi_subtenant_access` is disabled, the resulting mount path is:

```
/mnt/mauifs/nfsdir
```

If `multi_subtenant_access` is enabled, the resulting NFS mount path is:

```
/mnt/mauifs/subtenantuuid/nfsdir
```

• **host**: Specify the hosts that are allowed to access the share. If a host is not specified, it is not allowed access. Values are:

  • **Single machine** — Specify a fully qualified domain name, hostname (which can be resolved by the server), or IP address.

  • **Series of machines specified with wildcards** — Use the * or ? character to specify a string match. Do not use wildcards with IP addresses. When specifying wildcards in fully qualified domain names, dots (.) are not included in the wildcard. For example, `.*example.com` includes `one.example.com` but does not include `one.two.example.com`.

• **io**: Specifies whether the directory is read-only (ro) or has read/write (rw) permissions. The default is rw.

• **sync**: Specifies when the Atmos NFS server will respond to additional NFS client requests. Values are:

  • **yes** — The Atmos NFS server cannot reply to additional NFS client requests until it has received acknowledgement from the Atmos object store interface that it has received the current request.
Note: This does not mean that the request has been completed. Only that it has been received.

- no — The Atmos NFS server can reply to additional NFS client requests before it receives an acknowledgement that the current requested changes have been submitted to the Atmos object store interface.

- secure: Values are yes or no. If yes, all requests must originate from a port lower than 1024. If false, requests from any port number are accepted. The default is yes.

- anonuid: (Optional.) Specify a numeric value that represents the local file system UID of the anonymous account.

- anongid: (Optional.) Specify a numeric value that represents the group ID of the anonymous account.

Request header

POST /sub_tenant_admin/submit_add_node_nfs HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=48f4df73d20021e4716c4cd8fca9b0b7
Host: 10.5.116.244
Content-Length: 207

Request body

node_uuid=564D85B4-8FA2-34BF-24A4-9BFB9C17A02B&subtenant_name=t1&share_path=efgh&host=test1&io=rw&sync=no&secure=no&anonuid=&anongid=

Response header

HTTP/1.1 200 OK
Connection: close
Date: Wed, 20 Jan 2010 11:50:31 GMT
Set-Cookie: _gui_session_id=48f4df73d20021e4716c4cd8fca9b0b7;
path=/
Status: 200 OK
X-Runtime: 5062ms
ETag: "b98201b5ee2afcc512464a9c0842a7be"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 20

Response body

A successful response returns:

<added>true</added>

An unsuccessful response returns an error message, for example:

<added>Add node NFS entry error.</added>

or

<added>Share path is already existed.</added>
Change NFS shares

Replaces the existing configuration of an NFS share.

**Note:** This operation does not perform an update of the fields specified. It does a full replacement of the existing share configuration for the share path specified.

**Required role**

TenantAdmin or SubTenantAdmin

**HTTP method**

POST

**URI**

/sub_tenant_admin/submit_configure_node_nfs

**Request parameters**

HTTP header:

- **Cookie:** Set to the _gui_session_id returned by the authentication method.

HTTP Body:

- **node_uuid:** The UUID of the NFS access node where the NFS share you want to change resides. For information about how to obtain the UUID for a node, see “Translate node name to UUID” on page 153.

- **subtenant_name:** Specify the name of the subtenant that can use the NFS share.

- **share_path:** The relative path of the directory under /mnt/mauifs/ whose configuration you are changing. Cannot include these characters: & <> ' or " . This field cannot be greater than 256 characters. This field cannot contain Unicode characters.

- The resulting mount path depends on whether multi_subtenant_access is enabled or not. For example, if you specify share_path as nfsdir, and multi_subtenant_access is disabled, the resulting mount path is:

  /mnt/mauifs/nfsdir

- If multi_subtenant_access is enabled, the resulting NFS mount path is:

  /mnt/mauifs/subtenantuuid/nfsdir

- **host:** The hosts that access the share. Valid values are:

  - **Single machine** — A fully qualified domain name (which can be resolved by the server), a hostname (which can be resolved by the server), or IP address.

  - **Series of machines specified with wildcards** — Use the * or ? character to specify a string match. Do not use wildcards with IP addresses. When specifying wildcards in fully qualified domain names, dots (.) are not included in the wildcard. For example, *.example.com includes one.example.com but does not include one.two.example.com.
• **io:** Specify `ro` or `rw`. Specifies whether the directory is read-only (`ro`) or has read/write (`rw`) permissions. The default is `rw`.

• **sync:** Specifies when the Atmos NFS server will respond to additional NFS client requests. Values are:
  - **yes** — The Atmos NFS server cannot reply to additional NFS client requests until it has received acknowledgement from the Atmos object store interface that it has received the current request.
  - **no** — The Atmos NFS server can reply to additional NFS client requests before it receives an acknowledgement that the current requested changes have been submitted to the Atmos object store interface.

**Note:** This does not mean that the request has been completed. Only that it has been received.

• **secure:** Values are `yes` or `no`. If `yes`, all requests must originate from a port lower than 1024. If false, requests from any port number are accepted. The default is `yes`.

• **anonuid:** (Optional.) Specify a numeric value that represents the local file system UID of the anonymous account.

• **anongid:** (Optional.) Specify a numeric value that represents the group ID of the anonymous account.

---

**Request header**

```
POST /sub_tenant_admin/submit_configure_node_nfs HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=8cb62a5148719e998c2e28fa770c0c31
Host: 10.5.116.244
Content-Length: 204
```

**Request body**

```
node_uuid=564D85B4-8FA2-34BF-24A4-9BFB9C17A02B&subtenant_name=t1&share_path=abcd&host=test1&io=rw&sync=no&secure=no&anonuid=&anongid=
```

**Response header**

```
HTTP/1.1 200 OK
Connection: close
Date: Wed, 20 Jan 2010 11:39:55 GMT
Set-Cookie: _gui_session_id=8cb62a5148719e998c2e28fa770c0c31;
path=/
Status: 200 OK
X-Runtime: 6094ms
ETag: "27430fa0a0a46a8af142853e782c1a44"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 24
```

**Response body**

```
A successful request returns:
```

---

Change NFS shares 149
<cleared>true</cleared>

An unsuccessful response looks like this:
<cleared>Failed to modify node NFS entry.</cleared>

Delete NFS share

Deletes the specified NFS share path from the access node. This operation does not remove any data from the share.

Required role

TenantAdmin

HTTP method

POST

URI

/sub_tenant_admin/submit_delete_node_nfs

Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP Body:

- **node_uuid**: The UUID of the NFS access node where you want to delete the NFS share. For information about how to obtain the UUID for a node, see “Translate node name to UUID” on page 153.
- **subtenant_name**: Specify the name of the subtenant associated with the NFS share.
- **share_path**: The relative path of the directory under /mnt/mauifs/ to delete.

Request header

POST /sub_tenant_admin/submit_delete_node_nfs HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=097049b68b4134933598423bd1a49900
Host: 10.5.116.244
Content-Length: 80

Request body

node_uuid=564D85B4-8FA2-34BF-24A4-9BFB9C17A02B&subtenant_name=t1&share_path=abcd
Response header

HTTP/1.1 200 OK
Connection: close
Date: Wed, 20 Jan 2010 09:01:10 GMT
Set-Cookie: _gui_session_id=097049b68b4134933598423bd1a49900;
path=/
Status: 200 OK
X-Runtime: 2673ms
ETag: "0d7da07514bfa39e841b8ee8cba2b4b2"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 24

Response body

<deleted>true</deleted>

List NFS shares

Returns the list of NFS shares defined for the specified subtenant on the specified access node.

Required role

TenantAdmin

HTTP method

POST

URI

/sub_tenant_admin/node_nfs_list

Request parameters

HTTP header:

• **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP Body:

• **node_uuid**: The UUID of the NFS access node where the target NFS share resides. For information about how to obtain the UUID for a node, see “Translate node name to UUID” on page 153.

• **subtenant_name**: Specify the name of the subtenant that can use the NFS share. This is a required parameter.

Request header

POST /sub_tenant_admin/node_nfs_list HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=54e0c8db2fe57fe33d069098915cfa6
Request body

```
node_uuid=564D85B4-8FA2-34BF-24A4-9BFB9C17A02B&subtenant_name=t1
```

Response header

```
HTTP/1.1 200 OK
Connection: close
Date: Wed, 20 Jan 2010 12:37:27 GMT
Set-Cookie: _gui_session_id=54e0c8db2fe57fef33d069098915cfa6; path=/
Status: 200 OK
X-Runtime: 1805ms
ETag: "d5286a8ca5df4cb02d35e9c4965ad5bf"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 401
```

Response body

```
<result>
  <access_type>nfs</access_type>
  <nfs_node_list>
    <nfs_node>
      <share_path>/mnt/mauifs/test</share_path>
      <host>test1</host>
    </nfs_node>
    <nfs_node>
      <share_path>/mnt/mauifs/abcd</share_path>
      <host>host1</host>
    </nfs_node>
    <nfs_node>
      <share_path>/mnt/mauifs/efgh</share_path>
      <host>test1</host>
    </nfs_node>
  </nfs_node_list>
</result>
```

Unassign NFS access node

Removes the tenant assignment for this node. It does not remove any NFS assignments, but after the tenant assignment is removed, the tenant is no longer able to access those NFS shares.

Required role

SysAdmin

HTTP method

POST

URI

/maui_admin/submit_deassign_tenant_node
Request parameters

HTTP Header:
- **Cookie**: Set to the `_gui_session_id` returned by the authentication method.

HTTP Body:
- **node_name**: The name of the Atmos node whose tenant assignment you want to remove.

Request header

POST /maui_admin/submit_deassign_tenant_node HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=2267c906b87b35b46cef910be1790599
Host: 10.5.116.244
Content-Length: 59

Request body

`node_name=Shanghai-001`

Response header

HTTP/1.1 200 OK
Connection: close
Date: Wed, 20 Jan 2010 09:49:26 GMT
Set-Cookie: _gui_session_id=2267c906b87b35b46cef910be1790599;path=/
Status: 200 OK
X-Runtime: 3358ms
ETag: "27430fa0a046a8aaf142853e7821a44"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 24

Response body

A successful request returns the following:

`<cleared>true</cleared>`

A failed request returns a message like the following:

`<cleared>No such node: Beijing-002</cleared>`

**Translate node name to UUID**

Returns the UUID of the node name you pass in.

**Required role**

TenantAdmin
HTTP method

GET

URI

/util/translate

Request parameters

HTTP header:
- Cookie: Set to the _gui_session_id returned by the authentication method.

Querystring parameters:
- scope: Set the value to the string Node.
- input: Set the value to the string name.
- value: Specify the node name to convert to Atmos UUID.
- output: Set the value to the string uuid.

Request header

GET
/util/translate?scope=Node&input=name&value=Testtest1-001&output=uuid HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=1c05b0f4c4e08fa3d91f6b39035f86
Host: 10.5.116.110

Response header

HTTP/1.1 200 OK
Date: Fri, 24 Jul 2009 18:54:40 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 68
Set-Cookie: _gui_session_id=1c05b07ab4c4e08fa3d91f6b39035f86; path=/
Connection: close

Response body

A successful request returns the node ID.

<translated_id>564DE0F3-A5BF-9DFD-8C5B-5EA182150521</translated_id>

An unsuccessful request returns the following:

<translated_id><translated_id>

Examples: Working with NFS shares

This section describes the set of command operations you use for:
- “Adding an NFS access node”
• “Creating an NFS share”

Adding an NFS access node

Before you can export an NFS share from Atmos you must define one or more NFS access nodes. An access node can:

• Belong to only one tenant.
• Be assigned as both a Web-services access node and either an NFS or CIFS access node.

You must have tenant administrative permissions to define a node as an NFS access node. If the tenant has multiple subtenants, and all these subtenants need to access this node via NFS, then you should enable the multi_subtenant_access parameter. If you do not enable it, only the default subtenant can access the node. Follow these steps to add an NFS access node

1. Authenticate as a system administrator. For more information, see Chapter 10, “POX API: Authenticating Administrators.” Use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter.

2. If you do not have the UUID of the target access node, you can convert the node name to its UUID by making a request to translate it. For more information, see “Translate node name to UUID” on page 153.

3. Add the NFS access node. For more information, see “Assign NFS access node to a tenant” on page 144.

Creating an NFS share

Once you define an Atmos NFS access node, you can define one or more NFS shares for that node. Follow these steps, to create an NFS share:

1. Authenticate as a tenant administrator. For more information, see Chapter 10, “POX API: Authenticating Administrators,”. Use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter. You set Cookie to the _gui_session_id returned by the authentication method.

2. If you do not have the UUID of the target access node, you can convert the node name to its UUID by making a request to translate it. For more information, see “Translate node name to UUID” on page 153.

3. If one does not already exist, add a NFS node. For more information, see “Assign NFS access node to a tenant” on page 144.

4. Add the NFS share. For more information, see “Create NFS share” on page 145.
POX API: Managing NFS Shares
CHAPTER 15
POX API: Listing File Systems

This chapter describes the API that lists the file system capacity for each node.

- Listing file systems

This API requires an authenticated session. To create an authenticated session, see Chapter 10, “POX API: Authenticating Administrators”.

POX API: Listing File Systems

Listing file systems

Lists the total capacity, used capacity, and free capacity for each disk in the specified node.

Required role

SysAdmin

HTTP method

GET

URI

/mgmt/list_filesystem

Request parameters

HTTP header:

- **Cookie**: Set to the `_gui_session_id` returned by the authentication method.

Querystring parameters:

- **node_uuid**: Specify the UUID of the Atmos node whose file systems you want to list. For more information, see “Translate node name to UUID” on page 159.

Request header

GET

/mgmt/list_filesystem?node_uuid=44454C4C-4A00-1059-804C-B1C04F47463

Accept: application/xml

Cookie: _gui_session_id=b31789b92a88a9109f231bc9b3fa9e2d

Host: 127.0.0.1:80

Request body

None

Response header

HTTP/1.1 200 OK

Date: Wed, 18 Jun 2008 18:48:06 GMT

Server: Mongrel 1.1.3

Status: 200 OK

Cache-Control: no-cache

Content-Type: application/xml; charset=utf-8

Content-Length: 4239

Set-Cookie: _gui_session_id=b31789b92a88a9109f231bc9b3fa9e2d; path=/

Connection: close

Response body

<filesystem_list>
<filesystem>
<device_path>/dev/sde1</device_path>
<total_capacity>984.51 GB</total_capacity>
<used_capacity>33.55 MB</used_capacity>
<used_capacity_percent>0.0%</used_capacity_percent>
<free_capacity>984.47 GB</free_capacity>
<free_capacity_percent>100.0%</free_capacity_percent>
</filesystem>
<filesystem>
<device_path>/dev/sdf1</device_path>
<total_capacity>984.51 GB</total_capacity>
<used_capacity>33.55 MB</used_capacity>
<used_capacity_percent>0.0%</used_capacity_percent>
<free_capacity>984.47 GB</free_capacity>
<free_capacity_percent>100.0%</free_capacity_percent>
</filesystem>

Note: The used capacity value includes the amount used by the operating system.

Translate node name to UUID

Returns the UUID of the node name you pass in.

Required role

TenantAdmin

HTTP method

GET

URI

/util/translate

Request parameters

HTTP header:
- Cookie: Set to the _gui_session_id returned by the authentication method.

Querystring parameters:
- scope: Set the value to the string Node.
- input: Set the value to the string name.
- value: Specify the node name to convert to Atmos UUID.
- output: Set the value to the string uuid.

Request header

GET
/util/translate?scope=Node&input=name&value=Testtest1-001&output=uuid HTTP/1.1
Accept: application/xml
POX API: Listing File Systems

Cookie: _gui_session_id=1c05b07ab4c4e08fa3d91f6b39035f86
Host: 10.5.116.110

Response header

HTTP/1.1 200 OK
Date: Fri, 24 Jul 2009 18:54:40 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 68
Set-Cookie: _gui_session_id=1c05b07ab4c4e08fa3d91f6b39035f86; path=/
Connection: close

Response body

A successful request returns the node ID.

<translated_id>564DE0F3-A5BF-9DFD-8C5B-5EA182150521</translated_id>

An unsuccessful request returns the following:

<translated_id><translated_id>
CHAPTER 16
POX API: Managing UIDs and Shared Secrets

This chapter describes the API for managing UIDs and shared secrets.

- Create UID............................................................................................................. 162
- Disable UID ......................................................................................................... 163
- Enable UID ......................................................................................................... 164
- Get UID .............................................................................................................. 166
- List UIDs .......................................................................................................... 167
- Reset shared secrets ........................................................................................ 168

All of the API calls require an authenticated session. To create an authenticated session, see Chapter 10, “POX API: Authenticating Administrators.”
Create UID

Creates a UID with the name specified in the `app_name` HTTP body parameter, then generates and returns a shared secret for it. The HTTP request header must include the Cookie with the session ID (`_gui_session_id`) returned from a successful authentication.

**Note:** There is a slight delay before the UID is available on all Atmos nodes.

**Required role**

TenantAdmin

**HTTP method**

POST

**URI**

`/sub_tenant_admin/add_application`

**Request parameters**

HTTP header:

- **Cookie:** Set to the `_gui_session_id` returned by the authentication method.

HTTP body:

- **app_name:** Specify a string representing the UID. When specifying a UID, use these rules for acceptable characters.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Acceptable characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characters</td>
<td>Alphanumeric string (A-Z, a-z, 0-9)</td>
</tr>
<tr>
<td>Special characters</td>
<td>at sign (@), dash (-), period (.), underscore (_)</td>
</tr>
<tr>
<td>Minimum characters</td>
<td>1</td>
</tr>
<tr>
<td>Maximum characters</td>
<td>255</td>
</tr>
<tr>
<td>Case-sensitive?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

If unsupported characters are used, Atmos cannot retrieve the corresponding shared secret for the UID. This limitation does not apply to UIDs stored in the local lockbox.

If the UID will be used with Atmos Cloud Delivery Platform (ACDP), do not use the pipe symbol (`|`) or you will be unable to gather usage information for that UID as ACDP interprets this character as a delimiter.

**Note:** The POX API does not validate this parameter. Follow these naming rules to ensure data consistency.

- **sub_tenant_name:** Specify the name of the subtenant that owns the UID.
- **email:** Specify an email address.
Request header

POST /sub_tenant_admin/add_application HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=283ef56e4824548d162f9cfe273ae7ae
Host: 10.5.116.110
Content-Length: 56

Request body

app_name=samplenameUid0&sub_tenant_name=samplename&email=a@b.com

Response header

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 17:40:22
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 60
Set-Cookie: _gui_session_id=283ef56e4824548d162f9cfe273ae7ae; path=/
Connection: close

Response body

A successful request returns a shared secret:
<shared_secret>MSuDvtqmivSS4qRsvHyrAMuyY8Y=/</shared_secret>

Unsuccessful requests return error messages, for example:
<shared_secret>Error: UID TestUid0 already exists.</shared_secret>
or
<shared_secret>No such subtenant, or no permission to access requested subtenant.</shared_secret>

Disable UID

Changes the state of the specified UID to disabled. Disabled UIDs cannot be used in data requests.

Required role

TenantAdmin

HTTP method

POST

URI

/sub_tenant_admin/disable_application
POX API: Managing UIDs and Shared Secrets

Request parameters

HTTP header:
- **Cookie**: Set to the \_gui\_session\_id returned by the authentication method.

HTTP body:
- **app\_name**: Specify the UID to disable.
- **sub\_tenant\_name**: Specify the name of the subtenant that owns the UID.

Request header

```
POST /sub_tenant_admin/disable_application HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: \_gui\_session\_id=cf26c9cc550128f6dcd07bd1deaae4a2
Host: 10.5.116.110
Content-Length: 42
```

Request body

```
app\_name=TestUid6&sub\_tenant\_name=Test
```

Response header

```
HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 18:14:43 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=cf26c9cc550128f6dcd07bd1deaae4a2; path=/
Connection: close
```

Response body

A successful request returns:
```
<cleared>true</cleared>
```

If the UID does not exist, it returns:
```
<html>
<body>You are being
<a href="http://10.6.143.43/user/service_unavailable">redirected</a>.
</body>
</html>
```

If the subtenant does not exist, or a permissions error occurs, it returns:
```
<cleared>No such sub tenant or no permission to access requested sub tenant</cleared>
```

Enable UID

Changes the state of a UID from disabled to enabled.
Required role

TenantAdmin

HTTP method

POST

URI

/sub_tenant_admin/enable_application

Request parameters

HTTP header:

• **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP body:

• **app_name**: Specify the UID to enable.
• **sub_tenant_name**: Specify the name of the subtenant that owns the UID.

Request header

POST /sub_tenant_admin/enable_application HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=50a9726ce3b0bef91cfafbf1732fd589
Host: 10.5.116.110
Content-Length: 42

Request body

app_name=TestUid7&sub_tenant_name=Test

Response header

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 18:15:37 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=50a9726ce3b0bef91cfafbf1732fd589; path=/
Connection: close

Response body

A successful request returns:

<cleared>true</cleared>

An unsuccessful request returns a message like this:

<cleared>No such sub tenant or no permission to access requested sub tenant</cleared>
Get UID

Returns an XML document containing the shared secrets for the specified application name (UID) and subtenant name.

Required role

TenantAdmin

HTTP method

GET

URI

/sub_tenant_admin/get_uid

Request parameters

HTTP header:

• Cookie: Set to the _gui_session_id returned by the authentication method.

Querystring parameters:

• app_name: Specify the UID whose shared secret you want to return.
• sub_tenant_name: Specify the name of the subtenant that owns the UID.

Request header

GET /sub_tenant_admin/get_uid?app_name=TestUid0&sub_tenant_name=Test
  HTTP/1.1
  Accept: application/xml
  Cookie: _gui_session_id=035da1f5bddb4a8704734c6658eef70a
  Host: 10.6.143.43

Request body

None

Response header

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 17:42:21 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 165
Set-Cookie: _gui_session_id=035da1f5bddb4a8704734c6658eef70a; path=/
Connection: close

Response body

A successful request returns:
List UIDs

Returns an XML document that contains the following information about the UIDs for the specified tenant:

- The UID name.
- The UID shared secret.
- The UID status (enabled or disabled).
- The email address that is associated with the UID.

The HTTP request header must include the Cookie with the session ID (_gui_session_id) returned from a successful authentication.

Required role

TenantAdmin

HTTP method

GET

URI

/sub_tenant_admin/list_uid

Request parameters

HTTP header:

- Cookie: Set to the _gui_session_id returned by the authentication method.

Querystring parameters:

- sub_tenant_name: Specify the name of the subtenant whose UIDs you want to list.

Request header

GET /sub_tenant_admin/list_uid?sub_tenant_name=TestTenant HTTP/1.1
Accept: application/xml
POX API: Managing UIDs and Shared Secrets

Request body

None

Response header

HTTP/1.1 200 OK
Date: Mon, 14 Sep 2009 20:12:40 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 694
Set-Cookie: _gui_session_id=bdcdd918da8292c74272f91e806b4d7b; path=/
Connection: close

Response body

A successful request returns:

```xml
<uid_secret_list>
  <uid_secret>
    <uid>TestUidA</uid>
    <shared_secret>u3ZXmQztwkgbwD6fMIywx96cOE0=</shared_secret>
    <status>enabled</status>
    <email></email>
  </uid_secret>
  <uid_secret>
    <uid>TestUidB</uid>
    <shared_secret>5G3sIG7o1cdI0Oy+quML8d2n3o8=</shared_secret>
    <status>enabled</status>
    <email>a@b.com</email>
  </uid_secret>
  <uid_secret>
    <uid>user1</uid>
    <shared_secret></shared_secret>
    <status>enabled</status>
    <email></email>
  </uid_secret>
  <uid_secret>
    <uid>TestUid1</uid>
    <shared_secret>kJzLB9ha46mVfc3lhSnNs+dTYeU=</shared_secret>
    <status>enabled</status>
    <email></email>
  </uid_secret>
</uid_secret_list>
```

Reset shared secrets

Generates a new shared secret for the specified app_name (UID). If the specified UID does not exist, it is created.

Required role

TenantAdmin
HTTP method

POST

URI

/sub_tenant_admin/add_application

Request parameters

HTTP header:
- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP body:
- **app_name**: Specify the UID whose shared secret needs to be regenerated.
- **sub_tenant_name**: Specify the name of the subtenant that owns the UID.
- **regenerate**: Specify the value as **yes**.

Request header

POST /sub_tenant_admin/add_application HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=c4fc5be4097b2a9d1fe4ac7714adfba8
Host: 10.5.116.110
Content-Length: 57

Request body

app_name=TestUid7&sub_tenant_name=Test&regenerate=yes

Response header

HTTP/1.1 200 OK
Date: Mon, 06 Jul 2009 18:16:32 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 60
Set-Cookie: _gui_session_id=c4fc5be4097b2a9d1fe4ac7714adfba8; path=/
Connection: close

Response body

A successful request returns:

<shared_secret>7KVzlh1cyat9WnrTaFPyQxR0Zo=</shared_secret>

An unsuccessful request returns:

<shared_secret>No such sub tenant or no permission to access requested sub tenant</shared_secret>
CHAPTER 17
POX API: Managing Policy Selectors

This chapter describes the POX Policy Selector API.

- Assign a policy selector (full replacement) ............................................................. 172
- Assign a policy selector (incremental add) ............................................................ 173
- Create a policy selector ......................................................................................... 175
- Delete a policy selector (delete all) ......................................................................... 178
- Delete a policy selector (incremental delete) ......................................................... 179
- Get policy selector details ..................................................................................... 181
- List policy selectors ............................................................................................... 182
- List policy selectors per subtenant or list subtenants per policy selector ............... 184
- Modify a policy selector ........................................................................................ 185
- Remove a policy selector ....................................................................................... 188
Assign a policy selector (full replacement)

Assigns a policy selector to the specified subtenant(s). This operation replaces any existing assignments for the policy selector.

For example, suppose PolicySelector1 is assigned to subtenants t1, t2, and t3. If you call this operation to assign PolicySelector1 to subtenant t4, this operation removes the assignment of PolicySelector1 from subtenants t1, t2, and t3. Once it is complete, PolicySelector1 is assigned only to subtenant t4.

If the subtenant is a compliant subtenant (that is, it conforms to the SEC 17a-4f requirements), the policy selector you assign must also be compliant.

See “Assign a policy selector (incremental add)” on page 173 to add subtenants or policy selectors without replacing the existing settings.

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/submit_assign_policy

Request parameters

HTTP header:
- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP body:
- **selector**: Specify the policy selector name.
- **policy_type**: Set to 0.
- **operation_type**: Values are sync (the default) or async.
- **subtenants[]=subtenantname**: Specify a comma-separated list of subtenant names.

Restrictions

- Do not include spaces between the items in the comma-separated list for subtenants[] specification. If the list includes spaces, the operation ignores any names that follow the space. You can do this: subtenants[]=t1,subtenants[]=t2. But you cannot do this: subtenants[]=t1, subtenants[]=t2.

Request header

```text
POST /tenant_admin/submit_assign_policy HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=4bb29923e416a85e2204f3fccc3f73d
Host: 10.5.116.244
```
Assign a policy selector (incremental add)

Adds one or more subtenants to an existing policy selector assignment without removing any existing subtenant assignments.

To replace the set of subtenants assigned to a policy selector, see “Assign a policy selector (full replacement)” on page 172.

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/submit_incremental_assign_policy

Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.

HTTP body:

- **policy_type**: Specify 0.
- **operation_type**: Specify add.
POX API: Managing Policy Selectors

- **selector[]=<policy selector name>:** Specify a comma-separated list of policy selectors to assign.
- **subtenants[]= <subtenantname>:** Specify a comma-separated list of subtenant names.

**Restrictions**

- Do not include spaces between the items in the comma-separated list (for both selector[] and subtenants[] specifications). If the list includes spaces, the operation ignores any names that follow the space. You can do this: subtenants[] = t1, subtenants[] = t2. But you cannot do this: subtenants[] = t1, subtenants[] = t2.
- Do not assign multiple selectors to multiple subtenants or the request fails. You can assign:
  - Multiple selectors to one tenant.
  - One selector to multiple tenants.

**Request header**

POST /tenant_admin/submit_incremental_assign_policy HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=efdb3e094883505731bdaef1f928ff
Host: 10.5.116.244:3000
Content-Length: 251

**Request body**

This example assigns the policy selector, PolicySelector1, to subtenant t1, t2, t3, and t4.

policy_type=0&operation=add&selector[]=PolicySelector1&subtenants[]=t1 &subtenants[]=t2&subtenants[]=t3&subtenants[]=t4&operation_type=sync

This example assigns the policy selectors, PolicySelector1 and PolicySelector2, to subtenant t1.

policy_type=0&operation=add&selector[]=PolicySelector1&selector[]=Poli cySelector2&subtenants[]=t1&operation_type=sync

**Response header**

HTTP/1.1 200 OK
Connection: close
Date: Tue, 08 Jun 2010 08:34:58 GMT
Set-Cookie: _gui_session_id=93b654faf28d67fe6723e41a60781641; path=/
Status: 200 OK
X-Runtime: 12782ms
ETag: "27430faa0a46a8af142853e782c1a44"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 24

**Response body**

A successful request returns:
If policy selector or tenant does not exist, the operation returns messages like the following:

<cleared>Subtenant tx1 not exist.</cleared>
<cleared>Policy selector PolicySelectorx1 not exist.</cleared>

If you attempt to assign more than one policy selector to more than one tenant, the following error is returned.
<cleared>Cannot assign multiple selectors to multiple subtenants</cleared>

Create a policy selector

Creates a policy selector and binds it to an existing policy specification. You can define a policy selector to act on user metadata, system metadata, or an XQuery specification.

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/submit_create_selector

Request parameters

HTTP header:

• Cookie: Set to the _gui_session_id returned by the authentication method.

HTTP body:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>perform_action</td>
<td>Specify create (for create operations) or modify (for modify operations).</td>
</tr>
<tr>
<td></td>
<td>Example: perform_action=create</td>
</tr>
<tr>
<td>entry_index</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td></td>
<td>Example: entry_index=</td>
</tr>
<tr>
<td>entry[edit_mode]</td>
<td>Specify what type of metadata the policy selector you are creating will act on. Valid values: 0 (user metadata), 1 (system metadata), 2 (advanced).</td>
</tr>
<tr>
<td></td>
<td>Example: entry[edit_mode]=1</td>
</tr>
<tr>
<td>entry[spec_name]</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td></td>
<td>Example: entry[spec_name]=</td>
</tr>
<tr>
<td>other[sec_compliant]</td>
<td>Specify true if the policy selector must conform to the SEC 17a-4f requirements.</td>
</tr>
</tbody>
</table>
POX API: Managing Policy Selectors

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry[event]</td>
<td>Not used. Supply the parameter with no value. <strong>Example:</strong> entry[event]=</td>
</tr>
<tr>
<td>entry[id]</td>
<td>Specify the name of the selector to create. The policy selector name can be up to 255 alpha characters long. It can contain the letters a to z (both upper and lower case), any digits 0 to 9, or the following special characters: - _ @ $ ^ ;</td>
</tr>
<tr>
<td>other[specname]</td>
<td>Specify the name of an existing policy specification. This binds the policy selector to the policy specification. <strong>Example:</strong> other[specname]=default</td>
</tr>
<tr>
<td>entry[user_field_name]</td>
<td>Specify a user-metadata tag name. Use with user_operator and user_value. The value of edit_mode must be 0. <strong>Example:</strong> entry[user_field_name]=</td>
</tr>
<tr>
<td>entry[user_operator]</td>
<td>Specify a value comparison operator. Valid values: EQUALS, ENDS WITH, STARTS WITH, CONTAINS, and the following symbols: &lt;, =, &gt;, &lt;=, &gt;= Use with user_operator and user_value. The value of edit_mode must be 0. <strong>Example:</strong> entry[user_operator]=EQUALS</td>
</tr>
<tr>
<td>entry[user_value]</td>
<td>Specify a user metadata value. The value can be text, a number or a date that matches the value specified in the user_field_name parameter. Use with user_operator and user_field_name. The value of edit_mode must be 0. <strong>Example:</strong> entry[user_value]=</td>
</tr>
<tr>
<td>entry[user_event]</td>
<td>Specify a user metadata trigger event to activate this policy selector. Valid values: ON_CREATE, ON_UMD_UPDATE, ON_SMD_UPDATE. The value of edit_mode must be 0. <strong>Example:</strong> entry[user_event]=ON_CREATE</td>
</tr>
<tr>
<td>entry[sys_field_name]</td>
<td>Specify a system metadata tag name. Valid values are: atime, mtime, ctime, itime, uid, gid, size, objname. Use with sys_operator, sys_value, and sys_event. The value of edit_mode must be 1. <strong>Note:</strong> size is NOT available for the 'ON_CREATE' event. <strong>Example:</strong> entry[sys_field_name]=objname</td>
</tr>
<tr>
<td>entry[sys_operator]</td>
<td>Specify a system-metadata-value comparison operator. Valid values: EQUALS, ENDS WITH, STARTS WITH, CONTAINS, and the symbols &lt;, =, &gt;, &lt;=, &gt;= Use with sys_field_name, sys_value, and sys_event. The value of edit_mode must be 1. <strong>Example:</strong> entry[sys_operator]=ENDS WITH</td>
</tr>
<tr>
<td>entry[sys_value]</td>
<td>Specify a system metadata value. The value can be text, a number or a date that matches the value specified in the sys_field_name parameter. Use with sys_field_name, sys_operator, and sys_event. The value of edit_mode must be 1. <strong>Example:</strong> entry[sys_value]=basic_2_replica_2s_sub</td>
</tr>
<tr>
<td>entry[sys_event]</td>
<td>Specify a system metadata trigger event for policy activation. Valid values: ON_CREATE, ON_UMD_UPDATE, ON_SMD_UPDATE. Use with sys_field_name, sys_operator, and sys_value. The value of edit_mode must be 1. <strong>Example:</strong> entry[sys_event]=ON_CREATE</td>
</tr>
</tbody>
</table>
This example shows how to create a policy selector named `ps2` for the default policy. It operates on system metadata `ON_CREATE` event. When the `objname` ends with the extension `.basic_2_replica_2s_sub`.

**Request header**

```
POST /tenant_admin/submit_create_selector HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=f01b73beb4d2bc0b4171d76da7e260e9
Host: 10.5.116.244
Content-Length: 446
```

**Request body**

```
WITH entry[sys_value]=.basic_2_replica_2s_sub&entry[sys_event]=ON_CREATE&entry[xquery]='for $c in collection() return $c'
```

**Response header**

```
HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 06:47:59 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 20
Set-Cookie: _gui_session_id=f01b73beb4d2bc0b4171d76da7e260e9; path=/
Connection: close
```

**Response body**

```
A successful request returns:

<added>true</added>
```
Delete a policy selector (delete all)

Deletes the specified policy selector.

Required role

TenantAdmin

HTTP method

DELETE

URI

/sysmgmt/<tenantName>/policyselectors/<policyselectorName>

Request parameters

HTTP Header:

- `x-atmos-tenantadmin` — Specify a valid username of a user in the TenantAdmin role.
- `x-atmos-tenantadminpassword` — Specify the password for the `x-atmos-tenantadmin`.
- `x-atmos-authtype` — Specify `password`.

Request header

DELETE /sysmgmt/t1/policyselectors/PS1 HTTP/1.1
accept: */*
date: Tue, 02 Mar 2010 14:14:33 GMT
x-atmos-tenantadmin: t1admin
x-atmos-tenantadminpassword: password
x-atmos-authtype: password
host: 168.159.121.223

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Tue, 02 Mar 2010 14:14:37 GMT
Set-Cookie: _gui_session_id=83103d8c2cab67156540179cf3934e96; path=/
Status: 200 OK
x-atmos-sysmgmt-version: 1.0.0
X-Runtime: 2469ms
ETag: "4e7d6f65fa998ceadf75fab0b9447f71"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 64
Response body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<delete>deleted</delete>
```

Delete a policy selector (incremental delete)

Lets you remove the policy selector assignment as follows:

- Removes one policy selector assignment from multiple subtenants.
- Removes multiple policy selector assignments from one subtenant.

You cannot remove multiple policy selector assignments from multiple subtenants.

Required role

TenantAdmin

HTTP method

POST

URI

POST /tenant_admin/submit_incremental_assign_policy

Request parameters

HTTP Body:

- **policy_type**: Specify 0.
- **operation**: Specify `remove`.
- `selector[]=policy selector name`: Specify a comma-separated list of policy selectors to assign.
- `subtenants[]=subtenantname`: Specify a comma-separated list of subtenant names.
- **operation_type**: Specify `sync`.

Restrictions

- Do not include spaces between the items in the comma-separated list (for both `selector[]` and `subtenants[]` specifications). If the list includes spaces, the operation ignores any names that follow the space. You can do this: `subtenants[]=t1,subtenants[]=t2`. But you cannot do this: `subtenants[]=t1, subtenants[]=t2`.
- Do not assign multiple selectors to multiple subtenants or the request fails. You can assign:
  - Multiple selectors to one tenant.
  - One selector to multiple tenants.
POX API: Managing Policy Selectors

Request header

POST /tenant_admin/submit_incremental_assign_policy HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=cbeb3bb417f8edd602d908799e266ea
Host: 10.5.116.184:3000
Content-Length: 91

Request body

This example removes the policy selectors PS1 and PS2 from the subtenant t1s1:

policy_type=0&operation=remove&selector[]=PS1&selector[]=PS2&subtenants[]=t1s1&operation_type=sync

This example removes the policy selector PS1 from the subtenants t1s1 and t1s2:

policy_type=0&operation=remove&selector[]=PS1&subtenants[]=t1s1&subtenants[]=t1s2&operation_type=sync

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 14 Jun 2010 11:57:16 GMT
Set-Cookie: _gui_session_id=cbeb3bb417f8edd602d908799e266ea; path=/
Status: 200 OK
X-Runtime: 13790ms
ETag: "27430fa0a0a46a8af142853e782c1a44"
Cache-Control: private, max-age=0, must-revalidate
Server: Mongrel 1.1.5
Content-Type: application/xml; charset=utf-8
Content-Length: 24

Response body

A successful request returns:

<cleared>true</cleared>

If policy selector or tenant does not exist, the operation returns messages like the following:

<cleared>Subtenant tx1 not exist.</cleared>

or

<cleared>Policy selector PolicySelectorx1 not exist.</cleared>

If you attempt to remove more than one policy selector from more than one tenant, the following error is returned.

<cleared>Cannot assign multiple selectors to multiple subtenants</cleared>
Get policy selector details

Retrieves details about the specified policy selector. The HTTP request header must include the cookie with the session ID (_gui_session_id) returned from a successful authentication.

Details include:
- Policy name
- Status
- Edit mode
- Metadata tag
- Match operator
- Metadata value
- On event

Required role
TenantAdmin

HTTP method
GET

URI
/tenant_admin/get_policy_selector

Request parameters
HTTP header:
- Cookie: Set to the _gui_session_id returned by the authentication method.

Querystring parameters
- selector: Specify the selector name.

Request header
GET /tenant_admin/get_policy_selector?selector=policyselector1
Host: 10.5.116.244

Request body
None
Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 06:10:19 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 282
Set-Cookie: _gui_session_id=468042d911f268cebda90fba2ba10e; path=/
Connection: close

Response body

A successful request returns:

<policy_selector>
  <name>policyselector1</name>
  <status>Completed</status>
  <policy_id></policy_id>
  <edit_mode>systemmd</edit_mode>
  <metadata_tag>atime</metadata_tag>
  <match_operator>&gt;=</match_operator>
  <metadata_value>2010-01-08 00:00:00</metadata_value>
  <on_event>ON_CREATE</on_event>
</policy_selector>

List policy selectors

Retrieves general information about the policy selectors associated with a tenant or subtenant.

Required role

TenantAdmin

HTTP method

GET

URI

/tenant_admin/list_policy_selector

Request parameters

HTTP header:
- **Cookie**: Set to the _gui_session_id returned by the authentication method.

Querystring parameters:
- **sub_tenant_name**: (Optional.) specify the name of the subtenant if you want to narrow the result to only the policy selectors assigned to the subtenant. For example:

  GET /tenant_admin/list_policy_selector?sub_tenant_name=t1
List policy selectors

Request header

GET /tenant_admin/list_policy_selector HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=0537b8f8f361c03c47cc6975d04d4a9c
Host: 10.5.116.244

Request body

None

Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 06:16:11 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 992
Set-Cookie: _gui_session_id=0537b8f8f361c03c47cc6975d04d4a9c; path=/
Connection: close

Response body

A successful request returns:

<policy_selector_list>
  <policy_selector>
    <name>Cust1UIDSelector</name>
    <expression>uid equals Customer1UID</expression>
    <on_event>ON_CREATE</on_event>
    <spec>2LSand1RA</spec>
  </policy_selector>
  <policy_selector>
    <name>Cust2Selector</name>
    <expression>uid equals Customer2UID</expression>
    <on_event>ON_CREATE</on_event>
    <spec>1LSand1RA</spec>
  </policy_selector>
  <policy_selector>
    <name>DevSelector</name>
    <expression>uid equals DevUID</expression>
    <on_event>ON_CREATE</on_event>
    <spec>1LSand1LScomp</spec>
  </policy_selector>
  <policy_selector>
    <name>SampleSelector</name>
    <expression>atime equals 2009-09-16 00:00:00</expression>
    <on_event>ON_CREATE</on_event>
    <spec>SamplePolicy</spec>
  </policy_selector>
</policy_selector_list>
List policy selectors per subtenant or list subtenants per policy selector

Returns one of the following (per request):

- Given a subtenant name, returns the list of policy selectors that are currently assigned.
- Given a policy selector name, returns the list of subtenants to which it is currently assigned.

Required role

TenantAdmin

HTTP method

GET

URI

GET /tenant_admin/get_incremental_assign_policy

Request parameters

Supply one of the following Querystring parameters:

- `selector=`<selector-name>: Specify the name of the selector for which you want to list the subtenants.

or

- `subtenant=`<subtenant-name>: Specify the name of the subtenant that you want to get the list of policy selectors for.

Request header

GET /tenant_admin/get_incremental_assign_policy?selector=PS2 HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=e0aa6150b737744a737e0deaa87f852
Host: 10.5.116.184:3000

Request body

None

Response header

HTTP/1.1 200 OK
Connection: close
Date: Mon, 14 Jun 2010 12:40:54 GMT
Set-Cookie: _gui_session_id=e0aa6150b737744a737e0deaa87f852; path=/
Status: 200 OK
X-Runtime: 306ms
ETag: "72b640a6efc52596886b91fd48451b92"
Cache-Control: private, max-age=0, must-revalidate
Response body

If you request the list of subtenants for a policy selector, a successful result looks like this:

```xml
<result>
  <subtenants>
    <subtenant>t1</subtenant>
    <subtenant>t1s1</subtenant>
    <subtenant>t1s2</subtenant>
    <subtenant>t1s3</subtenant>
  </subtenants>
</result>
```

If you request the list of policies for a single tenant, a successful result looks like this:

```xml
<result>
  <selectors>
    <selector>PS1</selector>
    <selector>PS2</selector>
  </selectors>
</result>
```

If the selector or subtenant does not exist, messages like the following are returned:

```xml
<result>
  <error_message>Policy selector PSa not exist.</error_message>
</result>
```

or

```xml
<result>
  <error_message>Subtenant t1s1a not exist.</error_message>
</result>
```

Modify a policy selector

Modifies the specified policy selector. A modify operation replaces the entire existing policy selector.

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/submit_modify_selector

Request parameters

HTTP header:

- **Cookie**: Set to the _gui_session_id returned by the authentication method.
HTTP body:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>perform_action</td>
<td>Specify <code>create</code> (for create operations) or <code>modify</code> (for modify operations). Example: perform_action=modify</td>
</tr>
<tr>
<td>entry_index</td>
<td>Not used. Supply the parameter with no value. Example: entry_index=</td>
</tr>
<tr>
<td>entry[edit_mode]</td>
<td>Specify what type of data the policy selector you are creating. Valid values: 0 (user metadata), 1 (system metadata), 2 (advanced). Example: entry[edit_mode]=1</td>
</tr>
<tr>
<td>entry[spec_name]</td>
<td>Not used. Supply the parameter with no value. Example: entry[spec_name]=</td>
</tr>
<tr>
<td>other[sec_compliant]</td>
<td>Specify true if the policy selector must conform to the SEC 17a-4f requirements.</td>
</tr>
<tr>
<td>entry[event]</td>
<td>Not used. Supply the parameter with no value. Example: entry[event]=</td>
</tr>
<tr>
<td>entry[id]</td>
<td>Specify the name of the selector to create or modify. Example: entry[id]=ps2</td>
</tr>
<tr>
<td>other[specname]</td>
<td>Specify the name of an existing policy specification. This binds the policy selector to the policy specification. Example: other[specname]=default</td>
</tr>
<tr>
<td>entry[user_field_name]</td>
<td>Specify a user-metadata tag name. Use with user_operator and user_value. The value of edit_mode must be 0. Example: entry[user_field_name]=</td>
</tr>
<tr>
<td>entry[user_operator]</td>
<td>Specify a value comparison operator. Valid values: EQUALS, ENDS WITH, STARTS WITH, CONTAINS, and the following symbols: &lt;, =, &gt;, &lt;=, &gt;= Use with user_operator and user_value. The value of edit_mode must be 0. Example: entry[user_operator]=EQUALS</td>
</tr>
<tr>
<td>entry[user_value]</td>
<td>Specify a user metadata value. The value can be text, a number or a date that matches the value specified in the user_field_name parameter. Use with user_operator and user_field_name. The value of edit_mode must be 0. Example: entry[user_value]=</td>
</tr>
<tr>
<td>entry[user_event]</td>
<td>Specify a user metadata trigger event to activate this policy selector. Valid values: ON_CREATE, ON_UMD_UPDATE, ON_SMD_UPDATE. The value of edit_mode must be 0. Example: entry[user_event]=ON_CREATE</td>
</tr>
<tr>
<td>entry[sy_field_name]</td>
<td>Specify a system metadata tag name. Valid values are: atime, mtime, ctime, isize, uid, gid, size, objname. Use with sys_operator, sys_value, and sys_event. The value of edit_mode must be 1. <strong>Note:</strong> size is NOT available for the 'ON_CREATE' event. Example: entry[sy_field_name]=objname</td>
</tr>
</tbody>
</table>
Modify a policy selector

This example shows how to modify a policy selector named `ps2` for the default policy. It operates on system metadata `ON_CREATE` event. When the objname ends with the extension `.basic_2_replica_2s_sub`.

**Request header**

```
POST /tenant_admin/submit_modify_selector HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=d273fc48cce1ac20a7fdcfdf69434267
Host: 10.5.116.244
Content-Length: 451
```

**Request body**

```
perform_action=modify&entry_index=&entry[edit_mode]=1&entry[spec_name]=
  =&entry[event]=&entry[id]=ps2&other[specname]=default&entry[sys_field_name]=
  =&entry[sys_operator]=&entry[sys_value]=&entry[sys_event]=
  =
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| entry[sys_operator]        | Specify a system-metadata-value comparison operator. Valid values: EQUALS,
                           | ENDS, STARTS WITH, CONTAINS, and the symbols <, >, <=, >=. Use with sys_field_name,
                           | sys_value, and sys_event. The value of edit_mode must be 1. Example: entry[sys_operator]=ENDS WITH |
| entry[sys_value]           | Specify a system metadata value. The value can be text, a number or a date
                           | that matches the value specified in the sys_field_name parameter. Use with sys_field_name,
                           | sys_operator, and sys_event. The value of edit_mode must be 1. Example: entry[sys_value]=basic_2_replica_2s_sub |
| entry[sys_event]           | Specify a system metadata trigger event for policy activation. Valid values:
                           | ON_CREATE, ON_UMD_UPDATE, ON_SMD_UPDATE. Use with sys_field_name,
                           | sys_operator, and sys_value. The value of edit_mode must be 1. Example: entry[sys_event]=ON_CREATE |
| entry[xquery]              | Specify an Xquery string against which to match objects. Use with adv_event.
                           | The value of edit_mode must be 2. Example: entry[xquery]=‘for+%24c+in+collection%28%29+return+%24
                           | C’ The example represents the escaped string for 'for $c in collection() return $c' |
| entry[adv_event]           | Specify an Xquery trigger event for policy activation. Valid values:
                           | ON_CREATE (non-system-metadata modes), ON_UMD_UPDATE, ON_SMD_UPDATE.
                           | Use with xquery. The value of edit_mode must be 2. Example: entry[adv_event]=ON_CREATE |
| operation_type             | Values are sync (the default) or async.                                      |
Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 06:52:58 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=d273fc48ccelac20a7fdcfdf69434267; path=/
Connection: close

Response body

A successful request returns: <cleared>true</cleared>

Remove a policy selector

Removes the specified policy selector from all subtenants.

Required role

TenantAdmin

HTTP method

POST

URI

/tenant_admin/submit_assign_policy

Request parameters

HTTP header:

- Cookie: Set to the _gui_session_id returned by the authentication method.

HTTP body:

- selector: Specify the policy selector name.
- policy_type: Optional.

Request header

POST /tenant_admin/submit_assign_policy HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=4fe718ff82a9c1166c3386a90e4d311f
Host: 10.5.116.244
Content-Length: 37

Request body

selector=SampleSelector&policy_type=0
Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 06:25:05 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 24
Set-Cookie: _gui_session_id=4fe718ff82a9c1166c3386a90ebd311f; path=/
Connection: close

Response body

A successful request returns: <cleared>true</cleared>
CHAPTER 18
POX API: Managing Policy Specifications

This chapter describes the POX Policy Specification API.

- Create or update a policy specification ................................................................. 192
- Delete a policy specification ................................................................................. 202
- Get a policy specification .................................................................................... 203
- List policy specifications .................................................................................... 205
- Examples: Create a policy specification .............................................................. 207
Create or update a policy specification

Creates a new policy specification, or performs a full replacement of an existing one.
The modify operation is deprecated beginning with Atmos version 2.1.5.

**Note:** The create and update operations are combined because they require the same set of parameters.

Required role

TenantAdmin

HTTP method

POST

URI

To create a policy specification, use this URI:
	/tenant_admin/submit_create_spec

Request parameters

HTTP header:

- **Cookie:** Set to the `_gui_session_id` returned by the authentication method.

HTTP body:

The HTTP body is comprised of the set of parameters defined in:

- Table 16, “General parameters.”
- Table 17, “Metadata storage parameters.”
Table 16  General parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>perform_action</td>
<td>Specify create (for create operations)</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> perform_action=create</td>
</tr>
<tr>
<td>other[specname]</td>
<td>Specify the name of the policy to create. The name can be any string, but it cannot include special characters. For create operations, this name can be up to 255 alpha characters long. It can contain the letters a to z (both upper and lower case), any digits 0 to 9, or the following special characters: - _ @ $ ^ , ;</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The POX API does not validate this parameter. Follow these naming rules to ensure data consistency.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> other[specname]=samplepolicy</td>
</tr>
<tr>
<td>other[sec_compliant]</td>
<td>Specify true if the policy specification must conform to the SEC 17a-4f requirements.</td>
</tr>
<tr>
<td>entry[spec_name]</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
<tr>
<td>new_spec</td>
<td>Specify true for all create operations.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> new_spec=true</td>
</tr>
<tr>
<td>replica_id</td>
<td>Not used. Supply the parameter with no value.</td>
</tr>
</tbody>
</table>

Table 17  Metadata storage parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metadata[x][location_modifier]</td>
<td>Specifies the matching criteria for the location_place. [x] indicates replica number. For metadata the value is always 1. Valid values are: sameAs (no replica customization), and otherThan.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> metadata[1][location_modifier]=sameAs</td>
</tr>
<tr>
<td>metadata[x][location_place]</td>
<td>Specifies the location where Atmos should create new files and directories. [x] indicates replica number. For metadata, the value is always 1. Valid values:</td>
</tr>
<tr>
<td></td>
<td>- <strong>ANY</strong> (the default location): For the Web-service object interface, this is a location at or close to the client; for the Web-service namespace interface or file-system interface, this is the location of the parent directory</td>
</tr>
<tr>
<td></td>
<td>- <strong>$client</strong>: designates the location of the client where an operation is executed; A policy using the $client location descriptor stores the replica on a storage service at the same location as the client that initiated the request.</td>
</tr>
<tr>
<td></td>
<td>- [rmglocations]: A specific RMG location. This value is case-sensitive.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> metadata[1][location_place]=$client</td>
</tr>
<tr>
<td>Note:</td>
<td>Contact EMC Customer Support before using ANY.</td>
</tr>
</tbody>
</table>
Use the parameters in Table 18 to define the replica locations and their storage service attributes as follows:

- **Customize**: If enabled, you must define:
  - A replica location — The location can be a single location or a location expression. For more information about replica location expressions, see “Understanding replica locations” on page 200.
  - The storage services to apply to it. The `ssattrs_actions` and `ssattrs_placement` parameters are required when you enable customization.

- **Enable stripe**: Striping allows you to put blocks of data of the same replica on different nodes. If you enable striping, you must specify the `stripe_number`, `stripe_size`, and `stripe_units` parameters.

*Note:* Stripe configurations require an RPQ submission, contact EMC Customer Support before using this data layout.

- **Federate**: If you enable federation, you must also specify the `service_name` parameter. You can only specify the federation names that were defined by the SysAdmin.

- **GeoParity**: Provides data redundancy without the overhead of replication. GeoParity divides an object into rows, each of which contains \( m \) data fragments and \( k \) code fragments. Each fragment is stored on a different disk. Using this method, each row of the object can be reconstructed from any \( m \) of the total set of fragments, tolerating up to \( k \) unavailable fragments.

  GeoParity has an enhanced data protection mechanism, concurrent write and read operations of an object may result in an I/O error until the write operation has completed.

  A fragment distribution across multiple sites has a WAN latency dependency on performance, therefore should be designed in consultation with EMC.

*Note:* Contact EMC Customer Support before using the GeoParity 9/3 configuration.

For Atmos Virtual Edition (AVE) configurations, use protected storage (RAID or Mirroring).

*Note:* For GeoParity use with AVE, contact EMC personnel.

Policies with `sec_compliant` set to true require a minimum of 2 sync or 1 EC sync replica.
**Note:** You can define any number of data replicas, this is an array, and every element the array represents a single replica. The [x] indicates the replica number. Replicas start at 1.

### Table 18  Replica definition parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| spec[x][type] | Specify the replica type. Valid values: `sync`, `async`. If there is more than one replica, then the value for at least one must be `sync`.  
**Example:** `spec[1][type]=sync` |
| spec[x][location1][modifier] | Specify the location comparison. Valid values:  
- `sameAs`  
- `otherThan`.  
**Example:** `spec[1][location1][modifier]=sameAs`  
**Note:** The `spec[x][location][modifier]` parameter is supported for backwards compatibility. |
| spec[x][location1][place] | Specify the replica placement. Values to specify a single location are:  
- `ANY` (no replica customization)  
**Note:** Contact EMC Customer Support before using `ANY`.  
- A specific RMG location (case-sensitive)  
- `$client`  
- `$createClientLocationGroup`  
**Example:**  
`spec[1][location1][place]=$client`  
or  
`spec[1][location1][place]=NewYork`  
To specify a location group, use this format:  
`spec[1][location1][place]=location_group#<location-group-name>`  
Valid location group values are:  
- A specific location group (case-sensitive)  
- `$clientLocationGroup`  
- `$createClientLocationGroup`  
**Example:**  
`spec[1][location1][place]=location_group#$clientLocationGroup`  
or  
`spec[1][location1][place]=location_group#MidAtlantic` |
| spec[x][location2][operator] | (Optional). Specify the logical operator when building a location expression. Values are:  
- `AND`  
- `OR`  
Rules for logical expressions:  
- Location expressions that use the `sameAs` modifier can be operated on by `AND` and `OR`.  
- Location expressions that use the `otherThan` modifier can be operated on by `AND` only.  
- Location expressions that use `sameAs` and `otherThan` can be operated on by `AND` only.  
**Example:** `spec[1][location2][operator]=or` |
POX API: Managing Policy Specifications

### Table 18 Replica definition parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| spec[x][location2][modifier] | (Optional). Specify the location modifier when defining the second location in a location expression. Values are:  
  - sameAs  
  - otherThan.  
  **Example:** spec[1][location2][modifier]=sameAs |
| spec[x][location2][place] | (Optional). Specify the location place when defining the second location in a location expression.  
  Values to specify a single location are:  
  - ANY (no replica customization)  
  **Note:** Contact EMC Customer Support before using ANY.  
  - A specific RMG location (case-sensitive)  
  - $client  
  - $createLocationGroup  
  **Example:** spec[1][location2][place]=$client  
  or  
  spec[1][location2][place]=NewYork  
  To specify a location group, use this format: spec[1][location2][place]=location_group#<location-group-name>  
  Valid location group values are:  
  - A specific location group (case-sensitive)  
  - $clientLocationGroup  
  - $createClientLocationGroup  
  **Example:** spec[1][location2][place]=location_group#$clientLocationGroup  
  or  
  spec[1][location2][place]=location_group#MidAtlantic |
| spec[x][enable_customize] | (Optional.) Specify if you want to specify the replica location, and storage server attributes parameters:  
  **Example:** spec[1][enable_customize]=on |
| spec[x][ssattrs_placement] | Specify the storage server placement method that defines how the storage service allocates new objects to the physical disks under its management. Options are:  
  - GREEN — Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Only one or a few disks are active; the rest are spun down for energy conservation. When the active disks fill up, the spun-down disks are woken up and used.  
  - OPTIMAL or BALANCED — When either option is selected, Atmos writes objects to disks using a weighted random process so that disks with more free space have a higher chance of being used. For example, if disk 1 has 40% free space and disk 2 has 80% free space, disk 2 is twice as likely to be used than disk 1. This ensures that disk usage is balanced across all disks.  
  **Example:** spec[1][ssattrs_placement]=OPTIMAL |
**Table 18** Replica definition parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| spec[x][ssattrs_actions]  | Use when customization is on. Specify the data-at-rest services that are applied to replicas when they are stored. Valid values are: ANY — Pick any storage server, regardless of how it is configured (e.g., whether it is configured for compression).  
NONE — No services; normal writes and reads. This is the default.  
COMPRESSION — Data is compressed on write and decompressed on read. This conserves disk space but introduces processing overhead on the server, so typically it has a negative impact on performance. Compression can be used successfully for objects that are written once and accessed rarely.  
DEDUPLICATION — Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Multiple copies of the data are removed, and only one copy of the data is maintained. This is intended to conserve storage capacity. |
| spec[x][enable_stripe]     | Deprecated in Atmos Version 2.1.5. Existing customers using this feature will still be able to use it after upgrade. New and existing customers who are not using this feature will no longer be allowed to enable it. Optional. Specify if you want to enable striping. If it is, you must also specify the stripe_size integer, stripe_number integer, stripe_units.  
If you enable stripe, you can also enable customize, but you cannot enable flexout.  
**Example:** `spec[1][enable_stripe]=on` |
| spec[x][stripe_number]     | Use when striping is enabled. Specifies the number of nodes to stripe across.  
**Example:** `spec[1][stripe_number]=` |
| spec[x][stripe_size]       | Use when striping is enabled. Specify the amount of data written to each node.  
**Example:** `spec[1][stripe_size]=` |
| spec[x][stripe_units]      | Use when striping is enabled. Specify the stripe units. Valid values: B (no striping), KB, MB, GB, TB.  
**Example:** `spec[1][stripe_units]=B` |
| spec[x][enable_flexout]    | (Optional). Specify this parameter if you want to enable federation. When you enable flexout, you must specify the service_name parameter. If you enable flexout, you cannot also specify striping or customization.  
**Example:** `spec[1][enable_flexout]=on` |
| spec[x][service_name]      | Use when federation is enabled. Specify the service_name’s valid value is the [Cloud Federations name list] in the system configuration.  
**Example:** `spec[1][service_name]=` |
Use the parameters in the following table to define the read access, retention rules, and deletion rules. Retention and deletion are optional values and must be enabled before you can specify any values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec[x][enable_erasure]</td>
<td>Optional. Specify if you want to enable GeoParity.</td>
</tr>
<tr>
<td>Example: spec[1][enable_erasure]=on</td>
<td></td>
</tr>
<tr>
<td>spec[x][alg]</td>
<td>Specifies the GeoParity algorithm. Value is always CRS.</td>
</tr>
<tr>
<td>Example: spec[1][alg]=CRS</td>
<td></td>
</tr>
<tr>
<td>spec[x][fragmentation]</td>
<td>Use to specify the GeoParity fragmentation scheme. Values are:</td>
</tr>
<tr>
<td>Note: Contact EMC Customer Support before using the GeoParity 9:3 configuration.</td>
<td></td>
</tr>
<tr>
<td>Example: spec[1][fragmentation]=fragmentation2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>other[read_access]</td>
<td>Specify the replicas to use for read access.</td>
</tr>
<tr>
<td>geohraphic: Chooses the closest replica in geographic terms. This is the default.</td>
<td></td>
</tr>
<tr>
<td>random: Picks a replica at random.</td>
<td></td>
</tr>
<tr>
<td>Example: other[read_access]=geographic</td>
<td></td>
</tr>
<tr>
<td>other[enable_retention]</td>
<td>(Optional). Specify to enable retention.</td>
</tr>
<tr>
<td>For sec_compliant policy specifications, retention is enabled by default; otherwise it defaults to off when not supplied. Retention is a period of time during which the object cannot be modified or deleted. Atmos evaluates the retention period based on the object create time. For sec_compliant policy specifications, you cannot specify a start delay window. For non-compliant policies, if you enable retention, you must also specify a Start Delay Window using the other[retention_delay_xxx] parameters. The Start Delay Window specifies the amount of time to wait before applying the retention period. It is possible to apply a new retention period (after the first one has elapsed), but the time evaluation continues to be based on the object’s create time. For non-compliant policy specifications, when enable_retention is set to on, you must supply at least one of the other[retention_delay_xxx] parameters, and one of the other[retention_xxx] parameters. If you specify both retention and deletion, the deletion length must be longer than retention length.</td>
<td></td>
</tr>
<tr>
<td>Example: other[enable_retention]=on</td>
<td></td>
</tr>
<tr>
<td>other[retention_delay_year]</td>
<td>The start delay year.</td>
</tr>
<tr>
<td>Example: other[retention_delay_year]=2</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>other[retention_delay_month]</td>
<td>The start delay month. Example: other[retention_delay_month]=3</td>
</tr>
<tr>
<td>other[retention_delay_hour]</td>
<td>The start delay hour. Example: other[retention_delay_hour]=5</td>
</tr>
<tr>
<td>other[retention_delay_minute]</td>
<td>The start delay minute. Example: other[retention_delay_minute]=1</td>
</tr>
<tr>
<td>other[retention_delay_second]</td>
<td>The start delay second. Example: other[retention_delay_second]=3</td>
</tr>
<tr>
<td>other[retention_year]</td>
<td>Specify the retention year. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[retention_month]</td>
<td>Specify the retention Month. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[retention_day]</td>
<td>Specify the retention day. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[retention_hour]</td>
<td>Specify the retention hour. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[retention_minute]</td>
<td>Specify the retention minute. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[retention_second]</td>
<td>Specify the retention second. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[enable_deletion]</td>
<td>(Optional). Specify to enable deletion. Valid values are on or off. If you specify both retention and deletion, the deletion length must be longer than retention length. Deletion is the time after an object was created that it should be deleted. This time must be longer than the retention time since it is not possible to delete an object that is being retained. Deletion applies only to files, not directories. Example: other[enable_deletion]=on</td>
</tr>
<tr>
<td>other[deletion_year]</td>
<td>Specify the deletion year. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[deletion_month]</td>
<td>Specify the deletion month. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[deletion_day]</td>
<td>Specify the deletion day. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
<tr>
<td>other[deletion_hour]</td>
<td>Specify the deletion hour. Valid values are natural number like 0, 1, 2, ..., and so on. There is no upper limit.</td>
</tr>
</tbody>
</table>
Examples

Create

Request header

POST /tenant_admin/submit_create_spec HTTP/1.1
Accept: application/xml
Content-Type: application/x-www-form-urlencoded
Cookie: _gui_session_id=04a14f8c3dbd25426c5885e7808cfd84
Host: 10.5.116.244
Content-Length: 852

Request body

perform_action=create&other[specname]=def&entry[spec_name]=
&new_spec=true &replica_id=&metadata[1][location_modifier]=sameAs
&metadata[1][location_place]=ANY&spec[1][type]=sync&spec[1][location1]
[modifier]=sameAs&spec[1][location1][place]=ANY
&spec[1][ssattrs_placement]=OPTIMAL&spec[1][ssattrs_actions]=ANY
&spec[2][type]=sync&spec[2][location1][modifier]=sameAs
&spec[2][location1][place]=ANY&spec[2][ssattrs_placement]=OPTIMAL
&spec[2][ssattrs_actions]=ANY&operation_type=sync

Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 08:10:25 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 20
Set-Cookie: _gui_session_id=04a14f8c3dbd25426c5885e7808cfd84;
path=/
Connection: close

Response body

A successful request returns:

<added>true</added>

Understanding replica locations

You define a replica location using a location modifier (such as sameAs or otherThan), and
a location (which can be a single location or a location group).

To specify a single location, use the following parameters:

spec[x][location1][modifier] = <modifier>
spec[x][location1][place] = <location>

To specify one location group, use the following parameters:

spec[x][location1][modifier] = <modifier>
spec[x][location1][place] = location_group#<location_group_name>

To create a location expression, specify the first location by using:

spec[x][location1][modifier] = <sameAs|otherThan>
spec[x][location1][place] = <location>

or

spec[x][location1][modifier] = <sameAs|otherThan>
spec[x][location1][place] = location_group#<location-group-name>)

followed by the second location:

spec[x][location2][operator] = <AND | OR>
spec[x][location2][modifier] = <sameAs|otherThan>
spec[x][location2][place] = <location>

or for a location group:

spec[x][location2][operator] = <AND | OR>
spec[x][location2][modifier] = <sameAs|otherThan>
spec[x][location2][place] = location_group#<location_group_name>

For example,

spec[1][location1][modifier]=sameas
spec[1][location1][place]=London
spec[1][location2][operator]=or
spec[1][location2][modifier]=sameas
spec[1][location2][place]=Tokyo

Rules

Keep these rules in mind when you define location expressions:

- Expressions that include the sameAs modifier can use both the AND and the OR operators; for example, you can define both of the following expressions:
  
sameAs Europe AND sameAs Asia
  sameAs Europe OR sameAs Asia

- Expressions that include the otherThan modifier can use only the AND operator; for example:
  
otherThan London AND otherThan MexicoCity

- Expressions that include a combination of the modifiers, sameAs and otherThan, can include only the AND operator; for example:
  
sameAs London AND otherThan NewYork

- The first operand of a composite expression takes precedence over the second operand. This means that if you have:
  
An operand that is a single location (such as Beijing) and an operand that is a location group (such as China), and the single location is in the location group, and the modifiers are otherThan and sameAs
  
You should specify the expression like this:
  
otherThan <location> AND sameAs <location group>

For example, suppose you have a location group called China which includes Shanghai and Beijing. Make sure to specify:

otherThan Beijing AND sameAs China

and not:

sameAs China AND otherThan Beijing
When you specify otherThan Beijing AND sameAs China, Atmos distributes the replica only in Shanghai, but if you specified the reverse (sameAs China AND otherThan Beijing), Atmos would distribute the replica on any of the nodes in the China location group (including Beijing).

**Note:** If you create an expression that does not result in a location with valid resources, Atmos will not allow you to create the policy.

### Delete a policy specification

Deletes the specified policy. Returns a Boolean representing success (true) or failure (an error message) of the request. The HTTP request header must include the Cookie with the session ID (_gui_session_id) returned from a successful authentication.

You cannot delete a policy specification if it is used by a policy selector or a policy transition, or if it is assigned as the default policy for a subtenant.

**Required role**

TenantAdmin

**HTTP method**

GET

**URI**

/tenant_admin/submit_delete_spec

**Request parameters**

HTTP header:

- **Cookie:** Set to the _gui_session_id returned by the authentication method.

Querystring parameter:

- **spec_name:** Specify the policy name.
- **operation_type:** sync or async. Sync is the default.

**Request header**

GET

/tenant_admin/submit_delete_spec?spec_name=SamplePolicy&operation_type=sync HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=72435064e18f97f0f1827bce644c41e
Host: 10.5.116.244

**Request body**

None
Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 07:33:41 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 110
Set-Cookie: _gui_session_id=72435064e18f97f0f18273bce644c41e; path=/
Connection: close

Response body

A successful request returns:
<deleted>true</deleted>

An unsuccessful request returns an error message. The following is an example:
<deleted>Failed to delete placement policy. It's used by at least one policy selector or subtenant.</deleted>

Get a policy specification

Returns an XML document containing the following information about the specified policy:

- Policy name
- The metadata location modifier and metadata location
- The list of replicas and the replica attributes
- The replica location is returned in the combination of:
  - A <location/> element — Contains the full location description. Returned for a simple location (such as, NewYork), a location group, and a location expression.
  - A<location_modifier/> and <location_place/> — Populated for a simple location. Returned as an empty element if the location is a location group, or a location expression.

The HTTP request header must include the Cookie with the session ID (_gui_session_id) returned from a successful authentication.

Required role

TenantAdmin

HTTP method

GET

URI

/tenant_admin/get_policy_spec
POX API: Managing Policy Specifications

Request parameters

HTTP header:
- **Cookie**: Set to the \_gui\_session\_id returned by the authentication method.

Querystring parameter:
- **spec\_name**: Specify the policy name.

Request header

GET /tenant_admin/get_policy_spec?spec_name=test1 HTTP/1.1
Accept: application/xml
Cookie: \_gui\_session\_id=63e999e54b321eb737959faad3588948
Host: 10.5.116.244:1234
HTTP/1.1 200 OK
Date: Tue, 22 Mar 2011 08:33:36 GMT
Server: Mongrel 1.1.5
Status: 200 OK
X-Runtime: 701ms
ETag: "5a43608ea28d51ba0c48ad054c931421"
Cache-Control: private, max-age=0, must-revalidate
Content-Type: application/xml; charset=utf-8
Content-Length: 1056
Set-Cookie: \_gui\_session\_id=63e999e54b321eb737959faad3588948; path=/; secure
Connection: close

Request body

None

Response header

HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 07:15:17 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 1935
Set-Cookie: \_gui\_session\_id=f25456ba05eea359566ab263f91e89ef; path=/
Connection: close

Response body

A successful request returns:

```xml
<policy>
   <name>test1</name>
   <status>Completed</status>
   <metadata_location_modifier>otherThan</metadata_location_modifier>
   <metadata_location_place>$client</metadata_location_place>
   <replica_list>
      <replica>
         <id>0</id>
         <type>sync</type>
         <enable_customize>on</enable_customize>
         <location>sameAs $client</location>
         <location_modifier>sameAs</location_modifier>
         <location_place>$client</location_place>
      </replica>
   </replica_list>
</policy>
```
POX API: Managing Policy Specifications

List policy specifications

Returns an XML document that contains general information about all policy specifications for the tenant. The XML document contains the following information:

- Policy Name
- List of replicas including replica type, storage mechanism, and location.
- Each policy in the list has two elements that describe the location
  - The `<location/>` element — Returned when a simple location is defined. Returned as an empty string if the location contains a location group or a location expression, for example:
    `<location>westcaldwell</location>`
  - The `<locations/>` element — Returned when a simple location is defined and when a location group or location group defined by an expression. Used to describe the full location description, for example:
    `<locations>sameAs locgrp1 or sameAs locgrp2</locations>`

The HTTP request header must include the Cookie with the session ID (_gui_session_id) returned from a successful authentication.

Required role

TenantAdmin

HTTP method

GET

URI

/tenant_admin/list_policy

Request parameters

HTTP header:
POX API: Managing Policy Specifications

- **Cookie**: Set to the \_gui\_session\_id returned by the authentication method.

**Request header**

```
GET /tenant_admin/list_policy HTTP/1.1
Accept: application/xml
Cookie: _gui_session_id=7403fa595c3065898f20045dd8dd67dc
Host: 10.5.116.244
```

**Request body**

None

**Response header**

```
HTTP/1.1 200 OK
Date: Wed, 16 Sep 2009 07:23:04 GMT
Server: Mongrel 1.1.5
Status: 200 OK
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 2219
Set-Cookie: _gui_session_id=7403fa595c3065898f20045dd8dd67dc; path=/
Connection: close
```

**Response body**

A successful request returns an XML document like this:

```
<policy_list>
  <policy>
    <name>default</name>
    <replica_list>
      <replica>
        <type>sync</type>
        <storage_mechanism></storage_mechanism>
        <locations></locations>
        <location></location>
      </replica>
      <replica>
        <type>sync</type>
        <storage_mechanism></storage_mechanism>
        <locations></locations>
        <location></location>
      </replica>
    </replica_list>
    <retention>
      <delay></delay>
      <period></period>
    </retention>
    <deletion></deletion>
  </policy>
  <policy>
    <name>policy2</name>
    <replica_list>
      <replica>
        <type>sync</type>
        <storage_mechanism></storage_mechanism>
        <locations>sameAs Tokyo</locations>
        <location>Tokyo</location>
      </replica>
    </replica_list>
    <retention>
      <delay></delay>
      <period></period>
    </retention>
    <deletion></deletion>
  </policy>
</policy_list>
```
Examples: Create a policy specification

This section describes how to create a policy specification. To create a policy, you need to decide:

- How to handle metadata.
- How many replicas you want, and for each replica, you have to decide:
  - Is it sync or async.
• Does it have a special characteristics such as striping, GeoParity, or federation.
• What kind of read access should be used?
• Do you want to implement a retention or deletion rule?

The HTTP body of the create request includes parameters for values that define your response to each of the questions above. The following examples show how to supply these values. For a reference to all values, see “Create or update a policy specification” on page 192.

“Example — Two Replicas, No Data Transformation, Retention Enabled” on page 208

Example — Two Replicas, No Data Transformation, Retention Enabled

The following example shows how to build a policy that defines 2 replicas (1 async, 1 sync). Retention is enabled, but it has no other data transformation properties defined.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value/Description</th>
<th>Parameter to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Name</td>
<td>ExampleCustom2</td>
<td>perform_action=createother[specname]=ExampleCustom2entry[spec_name]=&amp;new_spec=true</td>
</tr>
<tr>
<td>Metadata</td>
<td>sameAs, $client</td>
<td>metadata[1][location_modifier]=sameAsmetadata[1][location_place]=$client</td>
</tr>
<tr>
<td>Replica 1</td>
<td>Replica Type: async</td>
<td>spec[1][type]=async</td>
</tr>
<tr>
<td></td>
<td>Enable customize: no</td>
<td>spec[1][location1][modifier]=sameAsspec[1][location1][place]=ANY</td>
</tr>
<tr>
<td></td>
<td>But you must still supply the default values for replica location and Storage</td>
<td>spec[1][ssattrs_placement]=OPTIMALspec[1][ssattrs_actions]=ANY</td>
</tr>
<tr>
<td></td>
<td>Server attribute parameters.</td>
<td></td>
</tr>
<tr>
<td>Replica 2</td>
<td>Replica Type: sync</td>
<td>spec[2][type]=sync</td>
</tr>
<tr>
<td></td>
<td>Enable customize: no</td>
<td>spec[2][location][modifier]=sameAs spec[2][location1][place]=ANY</td>
</tr>
<tr>
<td></td>
<td>You must still supply the default values for replica location and Storage Server</td>
<td>spec[2][ssattrs_placement]=OPTIMAL spec[2][ssattrs_actions]=ANY</td>
</tr>
<tr>
<td></td>
<td>attribute parameters.</td>
<td></td>
</tr>
<tr>
<td>Replica selection for</td>
<td>geographic</td>
<td>other[read_access]=geographic</td>
</tr>
<tr>
<td>read access</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Once you have the specification described, you can build the HTTP request as described next:

1. Authenticate as a tenant administrator. For more information, see Chapter 10, “POX API: Authenticating Administrators.”

2. Obtain and use the session ID for all subsequent calls by specifying it in the HTTP Header Cookie parameter. You set Cookie to the _gui_session_id returned by the authentication method.

3. Use the HTTP POST verb with the following URI:

   /tenant_admin/submit_create_spec

4. Build the HTTP body specifying the different policy specification attributes like this:


### Table: Policy Specification Attributes

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value/Description</th>
<th>Parameter to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retention</td>
<td>Retention is enabled.</td>
<td>other[enable retention]=on</td>
</tr>
<tr>
<td></td>
<td>A start delay window is required.</td>
<td>other[retention Delay year]=other[retention Delay month]=1 other[retention Delay day]=other[retention Delay hour]=other[retention Delay minute]=other[retention Delay second]=</td>
</tr>
<tr>
<td></td>
<td>Retention values are set.</td>
<td>other[retention Year]=5 other[retention Month]=other[retention Day]=other[retention Hour]=other[retention Minute]=other[retention Second]=</td>
</tr>
<tr>
<td>Deletion</td>
<td>Deletion is enabled.</td>
<td>other[enable Deletion]=on other[deletion Year]=other[deletion Month]=6 other[deletion Day]=other[deletion Hour]=</td>
</tr>
<tr>
<td>Operation type</td>
<td>sync</td>
<td>operation_type=sync</td>
</tr>
</tbody>
</table>
POX API: Managing Policy Specifications
INDEX

Symbols

_gui_session_id 13

A
access node
NFS 144
authentication 12
POX 13, 112, 114
REST 13

capacity
for file systems 158
CIFS
access nodes 120
create
NFS access node 144
NFS share 145
policy selector 75, 175
policy specification 64, 66, 67, 86, 192
policy specification example 100
subtenant 133
tenant 119
UID 50, 162

disable UID 163

D
delete
NFS shares 150
policy selector 78
policy specification 96, 202
Shared Secret 53
subtenant 38
UID 53
DeleteLatency 105

E
EMC online support website 7
enable UID 164
error handling
POX 14
REST 14

F
file system
capacity 158

H
handling errors 14
HTTP status codes 14
HTTPS 15

I
interface types 12

L
latency 105

M
metrics
subtenant 106
tenant 106
UID 106

N
NFS
access node assignment 144
access nodes 120
change shares 147
create share 145
delete share 150
list shares 151
remove tenant access node 152
node
delete latency 105
deletes 105
mean latency 105
readlatency 105
reads 105
uptime latency 105
UUID to name 153
web service statistics 20, 104
write latency 105
writes 105

P
policy selector
assign 74
create 75, 175
delete 78
get 79, 181
list 80, 182
modify 185
remove 188
update 81
policy specification
create 64, 66, 67, 69, 70, 86, 192
create example 100
create example with two replicas 100
delete 96, 202
get 203
list 98, 205
read 97

POST operation
Index

URL-encoding 15
POX
  authentication 13
  error handling 14
  interface 12
  session expiration 13

R
ReadLatency 105
REST
  authentication headers 13
  error handling 14
  interface 12
  resource hierarchy 15
  Status Codes 14
RMG
  info 29
  list 30
  role
    SysAdmin 13
    TenantAdmin 13

S
session
  expiration 13
Set-Cookie 13
Shared Secret
  read 56
  Reset 55
  reset 168
statistics
  web services 104
storage consumption 105
subtenant
  create 133
  get 134
  list 136
  modify 138
SubtenantAdmin role
  assign 132
  authenticate in POX 114
  remove 139
SysAdmin role 13
  authenticate in POX 112

T
tenant
  change access node assignment 152
  create 119
  get 120
  list 123
  NFS access node 144
tenant metrics
  metadatasize 107
  object count 107
  real size 107
  size 107
  total size 107
TenantAdmin