Moab Web Services 9.0.0

Reference Guide

October 2015

© 2015 Adaptive Computing Enterprises, Inc. All rights reserved.

Distribution of this document for commercial purposes in either hard or soft copy form is strictly prohibited without prior written consent from Adaptive Computing Enterprises, Inc.

Adaptive Computing, Cluster Resources, Moab, Moab Workload Manager, Moab Viewpoint, Moab Cluster Manager, Moab Cluster Suite, Moab Grid Scheduler, Moab Grid Suite, Moab Access Portal, and other Adaptive Computing products are either registered trademarks or trademarks of Adaptive Computing Enterprises, Inc. The Adaptive Computing logo and the Cluster Resources logo are trademarks of Adaptive Computing Enterprises, Inc. All other company and product names may be trademarks of their respective companies.

Adaptive Computing Enterprises, Inc.
1712 S. East Bay Blvd., Suite 300
Provo, UT 84606
+1 (801) 717-3700
www.adaptivecomputing.com

Scan to open online help

Contents

Welcome 1

Moab Web Services Overview 3

Chapter 1 Moab Web Services Setup 5

Configuring Moab Web Services 5

Setting Up MWS Security 22

Securing The Connection With Moab 23

Securing The Connection With MongoDB 23

Securing Client Connections To MWS 24

Securing The LDAP Connection 28

Securing The Connection With The Message Queue 29

Version And Build Information 30

Chapter 2 Access Control 33

Chapter 3 About The API 37

RESTful Web Services 37

Data Format 38

Global URL Parameters 39

Requesting Specific API Versions 42

Responses And Return Codes 43

Error Messages 46

Pre- And Post-Processing Hooks 48

Authentication 58

System Events 59

Chapter 4 Resources Introduction 63

Access Control Lists (ACLs) 64

Accounting 68

Accounting Accounts 68

Accounting Allocations 71

Accounting Charge Rates 75

Accounting Funds 79

Accounting Liens 89

Accounting Organizations 93

Accounting Quotes 96

Accounting Transactions 100

Accounting Usage Records 105

Accounting Users 119
Credentials 123
Diagnostics 139
Distinct 144
Events 146
Images 154
Insight Database 163
Job Arrays 171
Jobs 173
Job Templates 200
Metric Types 202
Nodes 204
Notification Conditions 212
Notifications 217
Permissions 225
Plugins 231
Plugin Types 239
Policies 244
Fairshare 251
Principals 260
Priority 266
Reports 269
Reservations 279
Resource Types 287
Roles 288
Standing Reservations 299

Chapter 5 Overview Of Reporting Framework 303
Example Report (CPU Utilization) 306

Chapter 6 About Moab Web Services Plugins 311
Plugin Overview 311
Plugin Introduction 312
Lifecycle States 314
Events 315
Custom Web Services 315
Utility Services 316
Data Consolidation 317
Routing 318
Plugin Developer's Guide 319
Requirements 320
Dynamic Methods 321
Logging 321
I18n Messaging 322
Configuration 324

Configuration Constraints 326
Individual Datastore 335
Exposing Web Services 337
Reporting State Data 339
Controlling Lifecycle 343
Accessing MWS REST Resources 344
Creating Events And Notifications 346
Handling Events 354
Handling Exceptions 356
Managing SSL Connections 356
Utilizing Services Or Custom "Helper" Classes 358
Packaging Plugins 363
Example Plugin Types 373
Moab Workload Manager Resource Manager Integration 374
Configuring Moab Workload Manager 375
Resource Manager Queries 376
Plugin Type Management 381
Listing Plugin Types 382
Displaying Plugin Types 382
Plugin Type Documentation 383
Add Or Update Plugin Types 383
Plugin Management 387
Listing Plugins 387
Creating A Plugin 388
Displaying A Plugin 389
Modifying A Plugin 389
Deleting A Plugin 390
Monitoring And Lifecycle Controls 390
Setting Default Plugin Configuration 393
Plugin Services 393
Job RM Service 394
Moab REST Service 394
Node RM Service 396
Plugin Control Service 396
Plugin Datastore Service 398
Plugin Event Service 401
SSL Service 403
Storage RM Service 403
Virtual Machine RM Service 403

Chapter 7 Plugin Types 405
Power Management Plugin 405
OpenStack Plugin 411
ViewpointQueryHelper Plugin 415

Chapter 8 References **427**
Client Code Samples 427
Javascript Code Samples 427
PHP Code Samples 429
Perl Code Samples 432
Python Code Examples 434
Curl Samples 435
Configuration 436
Resource Reference 446
Resources Reference 446
Fields: Access Control Lists (ACLs) 447
Accounting 455
Fields: Accounts 455
Fields: Allocations 459
Fields: Charge Rates 463
Fields: Fund Balances 465
Fields: Fund Statement Summary 473
Fields: Fund Statements 483
Fields: Funds 495
Fields: Liens 503
Fields: Organizations 507
Fields: Quotes 509
Fields: Transactions 514
Fields: Usage Records 518
Fields: Users 522
Fields: Credentials 524
Fields: Events 525
Fields: Images 531
Fields: Job Arrays 540
Fields: Jobs 604
Fields: Job Templates 666
Fields: Metric Types 696
Fields: Nodes 697
Fields: Notification Conditions 725
Fields: Notifications 729
Fields: Plugins 731
Fields: Plugin Types 736
Fields: Policies 740
Fields: Principals 767
Fields: Report Datapoints 775
Fields: Reports 777
Fields: Reservations 784
Fields: Resource Types 826
Fields: Roles 827

Fields: Report Samples 833
Fields: Standing Reservations 835
Fields: User's Permissions 895

Welcome

Welcome to the *Moab Web Services Reference Guide* for version 9.0.0.

> This Reference Guide assumes MWS has already been installed. See the *Moab HPC Suite Installation and Configuration Guide* for installation instructions, including troubleshooting the installation.

The following sections will help you quickly get started using MWS:

- Moab Web Services Overview: Gives an overview about what MWS is and how it works.
- Moab Web Services Setup: Contains instructions in order to get MWS configured and secured correctly.
- Access Control: Contains information describing how to manage access control in MWS.
- About the API: Describes how to use RESTful web services, explains the JSON data format used for all communications with MWS, describes global URL parameters used in MWS calls, and contains other helpful information for using the Moab Web Services API.
- Resources Introduction: Contains MWS resources and the HTTP methods defined on them.
- Overview of Reporting Framework: Provides an overview of the framework and the concepts related to it and works through an example report (CPU Utilization) with details regarding which web services to use and with what data.
- About Moab Web Services Plugins: Describes MWS plugins, their use, and their creation in Moab Web Services.
- References: Contains client code samples and information about configuration settings; also provides field information for each MWS resource object.

Moab Web Services Overview

Moab Web Services (MWS) is a component of Adaptive Computing Suites that enables programmatic interaction with Moab Workload Manager via a RESTful interface. MWS lets you create and interact with Moab objects and properties such as jobs, nodes, virtual machines, and reservations. MWS is the preferred method for those wishing to create custom user interfaces for Moab and is the primary method by which Moab Viewpoint communicates with Moab.

MWS communicates with the Moab Workload Manager (Moab) server using the same wire protocol as the Moab command-line interface. By publishing a standard interface into Moab's intelligence, MWS significantly reduces the amount of work required to integrate Moab into your solution.

This documentation is intended for developers performing such integrations. If you are a Moab administrator, and for conceptual information about Moab, see Moab Workload Manager Overview in the *Moab Workload Manager 9.0.0 Administrator Guide*.

Chapter 1 Moab Web Services Setup

This chapter explains what you need to know in order to get MWS configured, and secured correctly.

Before configuring MWS, confirm that all prerequisites were met and that MWS installed correctly. See the *Moab HPC Suite Installation and Configuration Guide* for prerequisites and installation instructions, including troubleshooting the installation.

In this chapter:

- Configuring Moab Web Services
- Setting up MWS Security
- Version and Build Information

Related Topics

- Moab Web Services Overview
- Access Control

Configuring Moab Web Services

This section describes the location of the MWS configuration files. It also shows some examples of how to configure logging.

To see a full reference to all configuration and logging parameters available in MWS, see Configuration.

This topic contains these sections:

- Home Directory on page 6
- Configuration Files on page 6
- Logging Configuration Using mws-config.groovy on page 6
- LDAP Configuration Using mws-config.groovy on page 12
- Insight Database Configuration Using mws-config.groovy on page 16
- PAM (Pluggable Authentication Module) Configuration Using mws-config.groovy on page 17
- OAuth Configuration Using mws-config.groovy on page 18

MWS does not support LDAP *and* PAM authentication at the same time.

Home Directory

The MWS home directory contains configuration files, log files, and files that serve features of MWS such as hooks and plugins. You should set the location of the MWS home directory using the **MWS_HOME** property. If you do not set **MWS_HOME** as a Java property or as an environment variable, then MWS will use `/opt/mws` as the default **MWS_HOME**.

Configuration Files

The primary configuration file is `MWS_HOME/etc/mws-config.groovy`. If this file is missing or contains errors, MWS will not start.

Configuration files can also be placed in the `MWS_HOME/etc/mws.d directory`. Any configuration files here get merged with `MWS_HOME/etc/mws-config.groovy`. In case of conflict, the configuration in `MWS_HOME/etc/mws.d` takes precedence.

If `MWS_HOME/etc/log4j.properties` exists, MWS will load it as well.

Logging Configuration Using `mws-config.groovy`

Shown below is an example that logs all error messages and fatal messages to `/opt/mws/log/mws.log` (For information about the format of the MWS logs, see "Standard Log Format" in the *Moab Workload Manager Administrator Guide*.). It also logs all stack traces to `/opt/mws/log/stacktrace.log`. Note that this example is not configured to log events; for details on logging events, see Configuring an Event Log on page 7.

```
Minimal logging configuration
-----------------------------

log4j = {
   appenders {
      rollingFile name: 'stacktrace',
         file: '/opt/mws/log/stacktrace.log',
         maxFileSize: '1GB'
      rollingFile name: 'rootLog',
         file: '/opt/mws/log/mws.log',
         threshold: org.apache.log4j.Level.ERROR,
         maxFileSize: '1GB'
   }
   root {
      debug 'rootLog'
   }
}
```

Alternatively, you may configure a console appender instead of a rolling file, as shown below.

```
Console logging configuration
-----------------------------

log4j = {
   appenders {
      rollingFile name: 'stacktrace',
         file: '/opt/mws/log/stacktrace.log',
         maxFileSize: '1GB'
      console name: 'consoleLog',
         threshold: org.apache.log4j.Level.ERROR
   }
   root {
      debug 'consoleLog'
   }
}
```

You may configure logging by using either `MWS_HOME/etc/mws-config.groovy` or `MWS_HOME/etc/log4j.properties`.

If you do not define any `log4j` configuration, MWS will write its log files to `java.io.tmpdir`. For Tomcat, `java.io.tmpdir` is generally set to `$CATALINA_BASE/temp` or `CATALINA_TMPDIR`.

Configuring an Event Log

Logging events to a flat file requires that you make a few changes to the configuration in the **log4j** section of the `mws-config.groovy` file so that events will be logged to the `events.log` file, and all other MWS logging information will be sent to the `mws.log` file.

Causing events.log to roll based on a time window

You can specify how often the `events.log` file rolls. The following example illustrates the configuration changes you will need make to `mws-config.groovy` to cause the `events.log` file to roll based on a time window. Note the following three examples:

- In this example, `mws-config.groovy` is configured so that `events.log` rolls daily at midnight.

```
Daily rolling events.log configuration in mws-config.groovy
-----------------------------------

log4j = {
  def eventAppender = new org.apache.log4j.rolling.RollingFileAppender(name:
'events', layout: pattern(conversionPattern: "%m%n"))
  def rollingPolicy = new org.apache.log4j.rolling.TimeBasedRollingPolicy
(fileNamePattern: '/tmp/events.%d{yyyy-MM-dd}', activeFileName:
'/tmp/events.log')
  rollingPolicy.activateOptions()
  eventAppender.setRollingPolicy(rollingPolicy)

  appenders {
    appender eventAppender

    rollingFile name: 'rootLog',
      file: '/tmp/mws.log',
      maxFileSize: '1GB'
  }

  root {
    warn 'rootLog'
  }

  trace additivity:false, events:'com.ace.mws.events.EventFlatFileWriter'
}
```

Note the **RollingFileAppender** and the **TimeBasedRollingPolicy** lines. These lines configure MWS to write the event log to the `events.log` file. Rolled log files will have a date appended to their name in this format: "yyyy-MM-dd" (for example, `events.log.2012-02-28`).

- If you want the event log file to roll at the beginning of each month, change the **fileNamePattern** TimeBasedRollingPolicy date format to `yyyy-MM`. For example:

```
Monthly event logs
-----------------------------------

def rollingPolicy = new org.apache.log4j.rolling.TimeBasedRollingPolicy
(fileNamePattern: '/tmp/events.%d{yyyy-MM}', activeFileName: '/tmp/events.log')
```

- If you want the event log file to roll at the beginning of each hour, change the date format to `yyyy-MM-dd_HH:00`. For example:

```
Hourly event logs
-----------------------------------

def rollingPolicy = new org.apache.log4j.rolling.TimeBasedRollingPolicy
(fileNamePattern: '/tmp/events.%d{yyyy-MM-dd_HH:00}', activeFileName:
'/tmp/events.log')
```

Configuring events.log to roll based on a file size threshold

You can also configure the events.log file to roll when the log size exceeds a specified threshold. The following example illustrates the configuration changes you will need to make to `mws-config.groovy` to cause the `events.log` file to roll on a size threshold. (In this example, `mws-`

`config.groovy` is configured so that `events.log rolls` when its size exceeds 50 MB.)

```
mws-config.groovy configuration that rolls events.log based on file size
-----------------------------------

log4j = {
  appenders {
    rollingFile name: 'events',
      file: '/tmp/events.log',
      maxFileSize: '50MB',
      maxBackupIndex:10

    rollingFile name: 'rootLog',
      file: '/tmp/mws.log',
      maxFileSize: '1GB'
  }

  root {
    warn 'rootLog'
  }

  trace additivity:false, events:'com.ace.mws.events.EventFlatFileWriter'
}
```

Note that **maxFileSize** is set to "50MB." This means that when the `events.log` file exceeds 50 MB, it will roll.

The name for the rolled log will be "events.log.1". When the *new* events.log file exceeds 50 MB, *it* will roll and be named "events.log.1", while the old "events.log.1" file will be renamed "events.log.2". This process will continue until the optional **maxBackupIndex** value is met. In the example above, **maxBackIndex** is set to 10. This means that MWS will delete all but the ten most recent `events.log` files. Using this feature helps prevent hard drives from filling up.

Additivity

The **additivity** attribute of the EventFlatFileWriter logger can be either `true` or `false`. If you specify `true`, events will be logged to the `events.log` file *and* the `mws.log` file. If you specify `false`, events will be logged to the `events.log` file only. (All other MWS logging information will be logged to the `mws.log` file, as configured by the `rootLog` appender.)

To log events to the `mws.log` file in addition to the `events.log` file, make the `additivity:true` configuration. For example:

```
Logging events to both events.log and mws.log
-----------------------------------

trace additivity:true, events:'com.ace.mws.events.EventFlatFileWriter'
```

For more configuration options, see Apache Extras Companion for log4j.

Deleting old events

If your MongoDB server is version 2.2 or later, MongoDB will automatically delete events older than 30 days (by default). For more information, including how to change this default, see `mws.events.expireAfterSeconds` in Configuration.

If your MongoDB server is older than version 2.2, MongoDB will store event data indefinitely. However, if disk space is limited, you may want to regularly delete old, unneeded events from MongoDB. This section contains some examples of how you can do this.

Let's say that you want to delete events that are older than 90 days. (There are 86,400,000 milliseconds in a day, so in this example, 90*86400000 corresponds to 90 days in milliseconds.):

- You could run this script:

```
Delete events older than 90 days
-----------------------------------

$ mongo
MongoDB shell version: 2.4.8
connecting to: test
> use mws
> db.event.remove({eventTime:{$lt:new Date(new Date().getTime()-90*86400000)}})
> exit
```

- To create a script to perform this task:

```
deleteOldEvents.sh
-----------------------------------

#!/bin/bash
printf 'use mws_dev\ndb.event.remove({eventTime:{$lt:new Date(new Date().getTime
()-90*86400000)}})\nexit' | mongo
```

- Now say that you want to set up a cron job so that old events are automatically deleted on a certain day of the week (for example, every Sunday at 2:00 a.m.). You would add an entry like this:

```
cron table entry to delete old events
-----------------------------------

00 02 * * 0 /root/deleteOldEvents.sh
```

Configuring an Audit Trail Log

Audit logging enables you to track changes to Permissions, Roles, and Principals.

```
mws-config.groovy configuration that enables audit logging
-----------------------------------

       def auditAppender = new org.apache.log4j.rolling.RollingFileAppender(
                       name: 'audit',
                       layout: new com.ace.mws.logging.ACPatternLayout("%j\t\t\t%c{1}\t\t\t%m%n"))
       def auditRollingPolicy = new org.apache.log4j.rolling.TimeBasedRollingPolicy(
                       fileNamePattern: '/opt/mws/log/audit.%d{yyyy-MM-dd}',
                       activeFileName: '/opt/mws/log/audit.log')
       auditRollingPolicy.activateOptions()
       auditAppender.setRollingPolicy(auditRollingPolicy)

       appenders {
               rollingFile name: 'stacktrace',
                               file: '/opt/mws/log/stacktrace.log',
                               maxFileSize: '100MB'
               rollingFile name: 'rootLog',
                               file: '/opt/mws/log/mws.log',
                               maxFileSize: '100MB', //The maximum file size for a single log file
                               maxBackupIndex: 10, //Retain only the 10 most recent log files, delet
to save space
                               layout: new com.ace.mws.logging.ACPatternLayout(), //Configures the c
of each log entry
                               threshold: org.apache.log4j.Level.ERROR //Ignore any logging entries
than this threshold

               appender eventAppender
               appender auditAppender
       }
```

You can customize audit logging in ways you can customize event logging. For example, you can specify how often the `audit.log` file rolls. You can also configure the `audit.log` file to roll when the log size exceeds a specified threshold.

> Follow the same steps indicated in the previous section on Configuring an Event Log for instruction on customizing audit logging; customization processes are the same for audit logging as for events logging.

audit.log file format

The default location to which the audit trail log is written is `/opt/mws/log/audit.log`. The log format is `yyyy-MM-dd HH:mm:ss resource username action data`. The following table offers a description for attributes included in the log format:

Parameter	Description
resource	The resource--permission, role, principal, or tenant--that changed.
username	The user's user name.

Parameter	Description
action	The type of change (create, update, or delete).
data	Dependent on what changed.

Sample `audit.log` format:

```
Audit trail log format
----------------------------------

2013-10-30 14:39:32,120 TENANT 'admin' updated resource named 'Engineering2' with
values:
     "name": "Engineering3",
     "attachedPrincipals": [{"name": "Engineering"}]
```

LDAP Configuration Using `mws-config.groovy`

The LDAP configuration provided below is for MWS to authenticate against a single LDAP server. If you wish to use LDAP to authenticate multiple servers, you must create and use a custom PAM module.

Using a Supported LDAP Directory Type

To configure an MWS connection to an LDAP server, add the following parameters to `mws-config.groovy`:

Throughout the following examples in this topic, you will see **dc=acme,dc=com**. "acme" is only used as an example to illustrate what you would use as your own domain controller if your domain name was "acme.com." You should replace any references to "acme" with your own organization's domain name.

Parameter	Description
ldap.server	The hostname or IP address of the LDAP server.
ldap.port	The port the LDAP server is listening on.
ldap.baseDNs	A list of distinguished names that are the root entries for LDAP searches.
ldap.bindUser	The distinguished name of the bind user.

Parameter	Description
ldap.password	The password of the ldap.bindUser.
ldap.directory.type	The type of LDAP directory (e.g. "Microsoft Active Directory"). This parameter can have the following values: • Microsoft Active Directory • OpenLDAP Using InetOrgPerson Schema • OpenLDAP Using NIS Schema • OpenLDAP Using Samba Schema

Here is a sample configuration for OpenLDAP.

If you followed the Adaptive Computing tutorial [link]"Setting up OpenLDAP on CentOS 6" your **ldap.directory.type** should be set to "OpenLDAP Using InetOrgPerson Schema".

```
Sample OpenLDAP configuration
-----------------------------

ldap.server  = "192.168.0.5"
ldap.port = 389
ldap.baseDNs = ["dc=acme,dc=com"]
ldap.bindUser = "cn=Manager,dc=acme,dc=com"
ldap.password = "*****"
ldap.directory.type = "OpenLDAP Using InetOrgPerson Schema"
```

Here is a sample configuration for Microsoft Active Directory.

```
Sample Active Directory configuration
-------------------------------------

ldap.server  = "192.168.0.5"
ldap.port = 389
ldap.baseDNs = ["CN=Users,DC=acme,DC=com","OU=Europe,DC=acme,DC=com"]
ldap.bindUser = "cn=Administrator,cn=Users,DC=acme,DC=com"
ldap.password = "*****"
ldap.directory.type = "Microsoft Active Directory"
```

To see how to configure a secure connection to the LDAP server, see Securing the LDAP Connection.

Using an Unsupported LDAP Directory Type

If you are not using one of the supported directory types, you can explicitly configure MWS to work with your LDAP schema by using the following parameters:

Parameter	Description
ldap.user.objectClass	The name of the class used for the LDAP user object. For example: • user • person • inetOrgPerson • posixAccount
ldap.group.objectClass	The name of the class used for the LDAP group object. For example: • group • groupOfNames • posixGroup
ldap.ou.objectClass	The name of the class used for the LDAP organizational unit object. for example: • organizationalUnit
ldap.user.membership.attribute	The attribute field in a user entry to use when loading the user's groups (optional if **ldap.group.membership.attribute** is defined). For example: • memberOf
ldap.group.membership.attribute	The attribute field in a group entry to use when loading the group's members (optional if **ldap.user.membership.attribute** is defined). For example: • member • memberUid
ldap.user.name.attribute	The attribute field to use when loading the username. This field must uniquely identify a user. For example: • sAMAccountName • uid

For example:

```
Advanced Active Directory configuration
---------------------------

ldap.server  = "myldaphostname"
ldap.port = 389
ldap.baseDNs = ["CN=Users,DC=acme,DC=com","OU=Europe,DC=acme,DC=com"]
ldap.bindUser = "cn=Administrator,cn=Users,DC=acme,DC=com"
ldap.password = "*****"
ldap.user.objectClass = "person"
ldap.group.objectClass = "group"
ldap.ou.objectClass = "organizationalUnit"
ldap.user.membership.attribute = "memberof"
ldap.group.membership.attribute = "member"
ldap.user.name.attribute = "sAMAccountName"
```

Here is a similar example for OpenLDAP. Note there is no user membership attribute in the OpenLDAP InetOrgPerson schema and thus **ldap.user.membership.attribute** is set to null. This is allowable because the **ldap.group.membership.attribute** is set.

```
Advanced OpenLDAP configuration
---------------------------

ldap.server  = "myldaphostname"
ldap.port = 389
ldap.baseDNs = ["dc=acme,dc=com"]
ldap.bindUser = "cn=Manager,dc=acme,dc=com"
ldap.password = "*****"
ldap.user.objectClass = "inetOrgPerson"
ldap.group.objectClass = "groupOfNames"
ldap.ou.objectClass = "organizationalUnit"
ldap.user.membership.attribute = null
ldap.group.membership.attribute = "memberUid"
ldap.user.name.attribute = "uid"
```

Overriding Attributes in a Supported LDAP Directory Type

You can also override attributes in supported directory types. For example, say you are using OpenLDAP with an NIS Schema. The group objectClass for NIS defaults to "groupOfNames," but you want to use "groupOfUniqueNames" instead while retaining all other defaults for NIS. You can do this by setting **ldap.directory.type** to "OpenLDAP Using NIS Schema" and overriding the **ldap.group.objectClass** attribute as follows:

```
Advanced OpenLDAP configuration
---------------------------

ldap.directory.type = "OpenLDAP Using NIS Schema"
ldap.group.objectClass = "groupOfUniqueNames"
```

LDAP is *not* currently used to authenticate users to MWS. LDAP is only used to map principals to roles, as explained in Principals.

The user class in your LDAP schema must have an attribute that uniquely identifies a user (for example: "uid" or "sAMAccountName").

Insight Database Configuration Using `mws-config.groovy`

You will need to create and configure a read-only user that MWS will use to connect to the Insight PostgreSQL database. It is recommended, but not required, that you configure an SSL connection to the Insight database.

To create a read-only PostgresSQL user

- On the machine that contains the PostgreSQL service, run the following commands:

```
[root]# su - postgres
[postgres]$ psql -c "CREATE USER mws WITH PASSWORD 'changeme!'"
[postgres]$ psql -d moab_insight -U moab_insight -h 127.0.0.1 -c "GRANT SELECT
ON ALL TABLES IN SCHEMA public TO mws;"
[postgres]$ psql -d moab_insight -U moab_insight -h 127.0.0.1 -c "ALTER DEFAULT
PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO mws;"
```

To allow MWS to query the PostgreSQL Moab Insight database

- Add the following parameters to `mws-config.groovy`:

Parameter	Description
dataSource_insight.url	The JDBC URL for the Insight database.
dataSource_insight.username	The username used to log in to the Insight database.
dataSource_insight.password	The password for the username.

The following is an example configuration:

```
dataSource_insight.url = "jdbc:postgresql://localhost/moab_insight"
dataSource_insight.username = "mws"
dataSource_insight.password = "changeme!"
```

To configure connections to the PostgreSQL Insight database over SSL

1. Ensure that your Java Runtime Environment (JRE) trusts the Insight database server's X.509 certificate. If the certificate was signed by a commercial certificate authority (CA), such as Verisign, then MWS should trust the certificate automatically. Otherwise, you must configure the JRE that MWS is using to trust this certificate explicitly. To do so, follow the steps described in the PostgreSQL documentation that describe how to use keytool to import the certificate.
2. Configure MWS to connect to the database over SSL by appending a URL

parameter of `ssl=true` to your **dataSource_insight.url**.

```
dataSource_insight.url = "jdbc:postgresql://localhost/moab_insight?ssl=true"
```

PAM (Pluggable Authentication Module) Configuration Using `mws-config.groovy`

PAM functions as bridge to the underlying Unix authentication system. PAM treats the user as if it is local to the Unix machine doing the authenticating and uses whatever the Unix user is authenticating with, whether it be LDAP or NIS. PAM uses configuration files that specify the how, when, or what for authentication, session management, and account management. Each configuration file can be different. For example, `sudo` configuration file for the "sudo" command will handle authentication differently than the `login` configuration file. These configuration files are dynamically read for `/etc/pam.d`.

Requirements for PAM

In order to use PAM with MWS, the following is required:

- The PAM application package must be installed. For example:

```
yum install pam
```

- You must have a PAM configuration file in the `/etc/pam.d` directory. The following is an example of what a PAM configuration file might look like:

```
#%PAM-1.0
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so


account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so


password    requisite     pam_pwquality.so try_first_pass retry=3 authtok_type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_
authtok
password    required      pam_deny.so


session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session     optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet
use_uid
session     required      pam_unix.so
```

- (Optional) You must have PAM modules installed for your specific needs.

The PAM application comes with default modules—for example, **pam_unix.xo**—that will check username and password credentials with Unix. You may have to install others for your distribution.

Configuring MWS to Use PAM

To configure an MWS connection to PAM, add the following parameter to `mws-config.groovy`:

Parameter	Description
pam.configuration.service	The name of the PAM configuration file located in `/etc/pam.d`. This parameter and specification tells MWS which PAM configuration file you want to use.

For example:

```
pam.configuration.service = "system-auth"
```

You can configure only one authentication method in `mws-config.groovy`—LDAP or PAM, but not both. If you have configured both LDAP and PAM, MWS defaults to using LDAP.

If you need multiple authentication methods, you must add them to your local PAM configuration. See your distribution documentation for details.

There is a security risk when authenticating local users through your PAM configuration. This behavior is highly discouraged and not supported by Adaptive Computing.

For more information about PAM, please see the following SLES and RedHat documentation.

OAuth Configuration Using `mws-config.groovy`

OAuth is a security framework designed to simplify authentication in web technologies. In the case of MWS, OAuth allows trusted client applications to securely delegate authentication to MWS. Once MWS has authenticated a user by verifying the username and password in LDAP, PAM, or NIS, MWS returns an access token to the client. The client then presents this access token to MWS to access resources. OAuth is very flexible and allows MWS to work in many different scenarios by use of grant types. For more information on OAuth and grant types, please see the following OAuth documentation.

Example Using 'password' Grant Type

Terminology

Resource Owner: The person accessing and manipulating data. For MWS, this would be the person who logs into the client (the user).

Service Provider: The site or service where protected resources live. This can be (but is not necessarily) also the identify provider, where usernames and passwords are stored. This is the MWS service itself.

Client: The application that wants to access a resource. For MWS this is the user interface, potentially including APIs and command-line tools.

Protected Resource: The data for which protection is desired. For MWS this would be Moab itself, and interaction with Moab.

Access Token: Instead of user credentials, OAuth uses tokens to issue requests, and the tokens get signed to indicate authorization.

Register a Client in MWS

Oauth requires client registration. Its client credentials are used to validate that the client is allowed to authenticate on behalf of a resource owner. It involves giving the client its own credentials (username and password). MWS will first authenticate the client using a client id (username) and client secret (password), then will authenticate the resource owner.

Add the following lines to /opt/mws/etc/mws-config.groovy:

```
grails.plugin.springsecurity.oauthProvider.clients = [
     [
     clientId:"THE_CLIENT_ID",
     clientSecret:"THE_CLIENT_SECRET",
     authorizedGrantTypes:["password"]
     ]
]
```

Replace `THE_CLIENT_ID` with client id (username). For example: `clientId:"iris"`. Also, replace `THE_CLIENT_SECRET` with client secret (password). For example: `clientSecret:"irisclientpassword"`, . Note that the values for `clientId` and `clientSecret` are case sensitive.

You can register more than one client. For example:

```
grails.plugin.springsecurity.oauthProvider.clients = [
     [
     clientId:"client_id_1",
     clientSecret:"client_secret_1",
     authorizedGrantTypes:["password"]
     ],
     [
     clientId:"client_id_2",
     clientSecret:"client_secret_1",
     authorizedGrantTypes:["password"]
     ]
]
```

Obtaining an Access Token from MWS for a Resource Owner (Logging In)

Before the client can access private data in MWS, the client must obtain an access token that grants access to the API. The token endpoint url is only used to gain an access token and log in a user.

Getting an access token:

```
POST http://localhost:8080/mws/rest/oauth/token?api-version=3
Adding header:
     "Content-Type: application/x-www-form-urlencoded"
Request body (String):
grant_type=password&client_id=THE_CLIENT_ID&client_secret=THE_CLIENT_
SECRET&username=RESOURCE_OWNER_USERNAME&password=RESOURCE_OWNER_PASSWORD
```

Example using curl:

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -v -d 'grant_
type=password&client_id=iris&client_secret=irisclientpassword&username=moab-
admin&password=secret' 'http://localhost:8080/mws/oauth/token'
```

Produces the following response:

```
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /mws/oauth/token HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:8080
> Accept: */*
> Content-Type: application/x-www-form-urlencoded
> Content-Length: 126
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Cache-Control: no-store
< Pragma: no-cache
< Set-Cookie: JSESSIONID=6CE8F9E7C454575FABCF3D156B153CFD; Path=/mws
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Fri, 18 May 2014 18:16:42 GMT
<
* Connection #0 to host localhost left intact
* Closing connection #0
{"access_token":"b693eec0-6c93-4540-8b2f-1e170be08046","token_type":"bearer","expires_
in":43096}
```

Sending the Access Token to MWS When Requesting Protected Resource

After the client obtains an access token, it will send the access token to MWS in an HTTP authorization header for each rest call.

> The client is responsible for handling user sessions with each access token, meaning the client has to request a new access token when a new user logs in.

Requesting an MWS resource (getting list of all nodes for example):

```
GET http://localhost:8080/mws/rest/nodes?api-version=3&fields=name
Adding authorization header:
     "Authorization: Bearer ACCESS_TOKEN"
```

Example using curl:

```
curl -X GET -H "Authorization: Bearer b693eec0-6c93-4540-8b2f-1e170be08046" -v
'http://localhost:8080/mws/rest/nodes?api-version=3&fields=name'
```

Produces the following response:

```
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /mws/rest/nodes?api-version=3&fields=name HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: localhost:8080
> Accept: */*
> Authorization: Bearer b693eec0-6c93-4540-8b2f-1e170be08046
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json;charset=UTF-8
< Pragma: no-cache
< Set-Cookie: JSESSIONID=6CE8F9E7C454575FABCF3D156B153CFD; Path=/mws
< Content-Type: application/json;charset=UTF-8
< Content-Language: en-US
< Transfer-Encoding: chunked
< Date: Fri, 18 May 2014 18:39:07 GMT
<
{"totalCount":3,"resultCount":3,"results":[{"name":"node1"},{"name":"node2"},
{"name":"node3"}]}
```

Related Topics

- Setting up MWS Security
- Version and Build Information

Setting up MWS Security

When running MWSin production environments, security is a major concern. This section focuses on securing the connections with MWS:

- The connection between MWS and Moab Workload Manager (see Securing the Connection with Moab).
- The connection between MWS and MongoDB (see Securing the Connection with MongoDB).
- The connections between clients and MWS (see Securing Client Connections to MWS).
- The connection between MWS and LDAP (see Securing the LDAP Connection).
- The connection with the message queue (see Securing the Connection with the Message Queue).

Related Topics

- Configuring Moab Web Services
- Version and Build Information

Securing the Connection with Moab

MWS communicates with Moab via the Moab Wire Protocol, which uses a direct connection between the two applications. The communication over this connection uses a shared secret key, which is discussed in the installation instructions (see Installing Moab Web Services in the installation documentation). However, the communication is not encrypted and is therefore susceptible to eavesdropping and replay attacks. For this reason, MWS is supported only when running on the same machine as Moab. This assures that any connections between the two applications occur internally on the server and are not exposed to external users.

Related Topics

- Setting up MWS Security

Securing the Connection with MongoDB

By default, the connection between MWS and MongoDB is not authenticated. To enable authentication, follow the instructions below. For further reading, see the MongoDB tutorial "Control Access to MongoDB Instances with Authentication."

To enable an authenticated connection between MWS and MongoDB

1. Add an administrative user to the `admin` database.
2. Add an MWS user to the `mws` database.
3. To support MWS API version 2, add an MWS user with "read-only" rights to the `moab` database.

   Here is an example of how to create all the required users. The users in the `moab` database are required only for MWS API version 2.

```
[root]# service mongod start
[root]# mongo
> use admin;
> db.addUser("admin_user", "secret1");
> use moab;
> db.addUser("moab_user", "secret2");
> db.addUser("mws_user", "secret3", true);
> use mws;
> db.addUser("mws_user", "secret3");
> exit;
```

The passwords used here ("secret1," "secret2," and "secret3") are examples. Choose your own passwords for these users.

4. Add the MWS user credentials (the ones you just created) to the `mws-config.groovy` file. For example:

```
grails.mongo.username = "mws_user"
grails.mongo.password = "secret3"
```

5. Enable authentication in the MongoDB configuration file (called `/etc/mongodb.conf` on many Linux distributions). In that file, look for **#auth = true** and uncomment it.
6. Restart MongoDB.
7. Restart Tomcat.

If authentication is enabled in MongoDB, but the MWS user was not properly created or configured, MWS will not start. In this case, see the log file(s) for additional information.

Related Topics

- Setting up MWS Security

Securing Client Connections to MWS

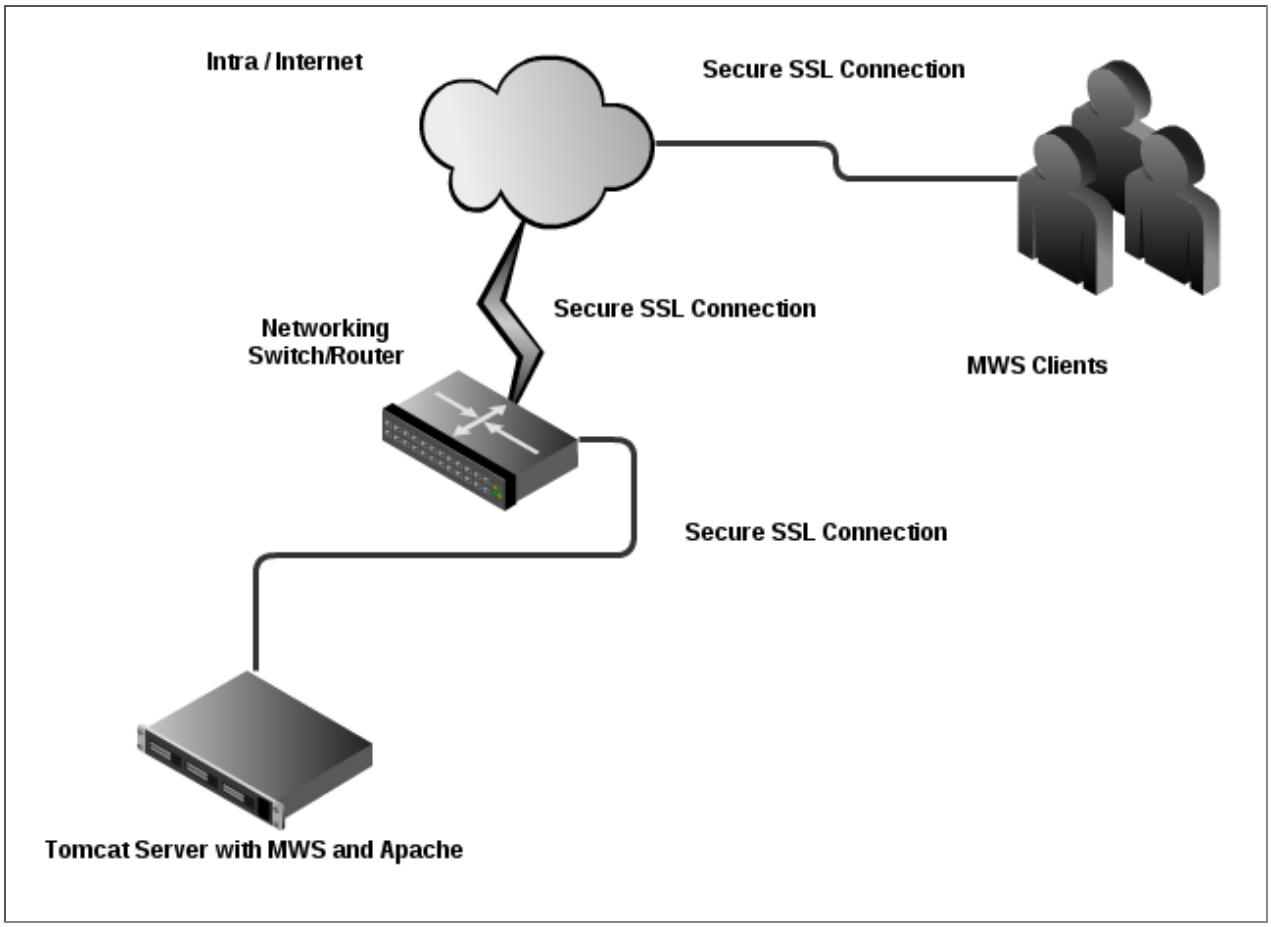
All connections to MWS, except those requesting the documentation or the main page, must be authenticated properly. MWS uses a single-trusted-user authentication model, meaning a single user exists that has access to all aspects of MWS. The username and password for this user are configured with the `auth.defaultUser` properties in the configuration file. For more information, see Configuration.

When using the MWS user interface in a browser, the user will be prompted for username and password. For information on how to authenticate requests when not using a browser, see Authentication.

The username and password in the Basic Authentication header are encoded but not encrypted. Therefore, it is *strongly* recommended that MWS be run behind a proxy (like Apache) with SSL enabled. The instructions below provide an example of how to do this.

Encrypting Client Connections Using Apache and SSL

This section shows how to encrypt client connections to MWS using Apache and SSL. These instructions have been tested on CentOS™ 6.2 with the "Web Server" software set installed. The same ideas are applicable to other operating systems, but the details might be different. As shown in the diagram below, these instructions assume that Tomcat and Apache are running on the same server.



To encrypt client connections using Apache and SSL

1. Create a self-signed certificate. (If desired, see http://www.openssl.org/docs/HOWTO/certificates.txt for more information.)

   > Instead of creating a self-signed certificate, you can buy a certificate from a certificate vendor. If you do, then the vendor will provide instructions on how to configure Apache with your certificate.

2. Do the following:

   a. Run these commands:

   ```
   cd /etc/pki/tls/certs
   cp -p make-dummy-cert make-dummy-cert.bak
   cp -p localhost.crt localhost.crt.bak
   ```

   b. Edit `make-dummy-cert` and replace the **answers()** function with code similar to this:

```
answers() {
   echo US
   echo Utah
   echo Provo
   echo Adaptive Computing Enterprises, Inc.
   echo Engineering
   echo test1.adaptivecomputing.com
   echo
}
```

c. Run this command:

```
./make-dummy-cert localhost.crt
```

3. Configure Apache to use the new certificate and to redirect MWS requests to Tomcat. To do so, edit `/etc/httpd/conf.d/ssl.conf`. Do the following"

   a. Comment out this line:

```
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

   b. Add these lines near the end, just above **</VirtualHost>**:

```
ProxyPass /mws http://127.0.0.1:8080/mws retry=5
ProxyPassReverse /mws http://127.0.0.1:8080/mws
```

4. Configure Apache to use SSL for all MWS requests. Add these lines to the end of `/etc/httpd/conf/httpd.conf`:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (/mws.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

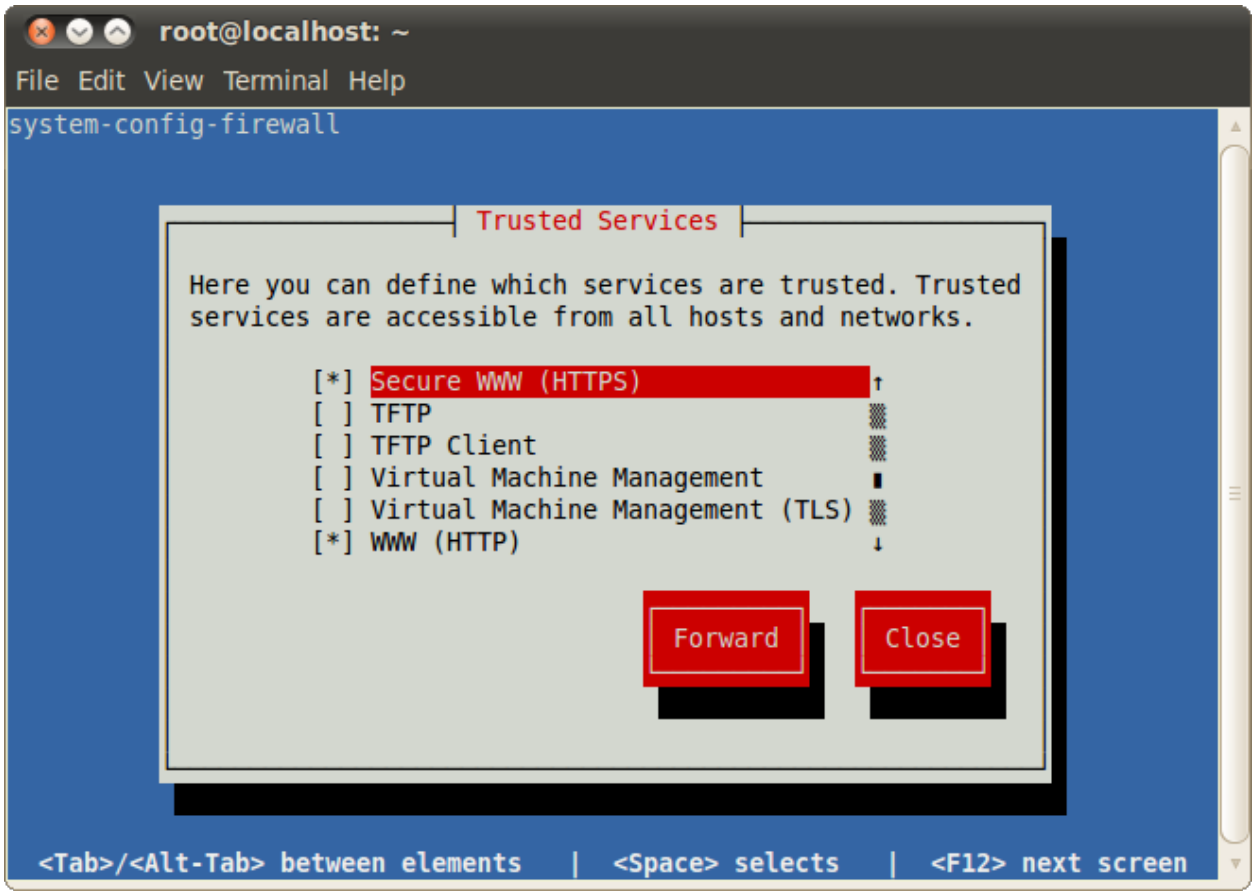
5. Give Apache permission to connect to Tomcat.

```
setsebool -P httpd_can_network_connect 1
```

6. Turn on Apache.

```
chkconfig httpd on
service httpd start
```

7. Using `system-config-firewall-tui`, enable "Secure WWW (HTTPS)" and "WWW (HTTP)" as trusted services.



Encrypting Client Connections Using Tomcat and SSL

This section shows how to encrypt client connections to MWS using Tomcat and SSL but without requiring the use of Apache. These instructions have been tested on CentOS™ 6.2 with Tomcat 6.0.

To encrypt client connections using Tomcat and SSL

1. First, you must generate a certificate. Do the following:
   a. Use the keytool utility that is shipped with the Oracle Java Runtime Environment. As the Tomcat user, run the following:

```
keytool -genkey -alias tomcat -keyalg RSA
```

   b. Specify a password value of "changeit". This will create a .keystore file that contains the new certificate in the user's home directory.
2. Enable the Tomcat SSL connector. Do the following:
   a. Open the `server.xml` file, usually located in `$CATALINA_HOME/conf/` (`$CATALINA_HOME` represents the directory where Tomcat is installed).

b. Verify the SSL HTTP/1.1 Connector entry is enabled. To do so locate the SSL HTTP/1.1 Connector entry and uncomment it.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true" maxThreads="150"
scheme="https" secure="true" clientAuth="false" sslProtocol="TLS" />
```

> The code above enables SSL access on port 8443. The default for HTTPS is 443, but just as Tomcat uses 8080 instead of 80 to avoid conflicts, 8443 is used instead of 443.

c. Save the `server.xml` file.

d. Verify that `server.xml` is owned by the Tomcat user.

```
chown -R tomcat:tomcat server.xml
```

e. Next modify the MWS`web.xml` file. Add a security-constraint section to the `$CATALINA_HOME/webapps/mws/WEB-INF/web.xml` file found in your Tomcat directory.

```
<web-app>
   …
      <security-constraint>
         <web-resource-collection>
            <web-resource-name>MWS Secure URLs</web-resource-name>
            <url-pattern>/*</url-pattern>
         </web-resource-collection>
         <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
         </user-data-constraint>
      </security-constraint>
</web-app>
```

f. Now restart tomcat.

Related Topics

- Setting up MWS Security

Securing the LDAP Connection

All connections from MWS to the LDAP server should be secured with SSL or StartTLS to ensure passwords and other sensitive information are encrypted as they pass to and from the LDAP server. If the LDAP server does not support SSL or StartTLS, the rest of this section is irrelevant.

Determine Whether the LDAP Server's Certificate is Trusted

If the LDAP server's X.509 certificate has been signed by a trusted certificate authority such as Verisign, Thawte, GeoTrust, and so on, Java will trust the certificate automatically and you won't need to add the certificate to Java's keystore. Consult your IT department to determine whether the LDAP server certificate has been signed by a trusted certificate authority. If the LDAP server certificate is signed by a trusted certificate authority, skip ahead to Configure MWS to Connect to LDAP Cerver Using SSL or StartTLS. Otherwise, follow the instructions in Trusting servers in Java to add the certificate to Java's keystore.

Configure MWS to Connect to LDAP Cerver Using SSL or StartTLS

To configure MWS to connect to LDAP using SSL/TLS

1. Update the `ldap.port` and `ldap.security.type` parameters in `/opt/mws/etc/mws-config.groovy`.

```
ldap.port = 636
ldap.security.type = "SSL"
```

To configure MWS to connect to LDAP using StartTLS

1. Update the `ldap.port` and `ldap.security.type` parameters in `/opt/mws/etc/mws-config.groovy`.

```
ldap.port = 389
ldap.security.type = "StartTLS"
```

The table below lists the possible values for `ldap.security.type`:

ldap.security.type	Default port	Notes
None	389	This is the default if no security type is configured. All data is sent in plain text.
SSL	636	Requires server certificate. All data is encrypted.
StartTLS	389	Starts as an insecure connection and is upgraded to an SSL/TLS connection. Requires server certificate. After upgrade all data is encrypted.

Securing the Connection with the Message Queue

MWS supports message queue security with AES. If the **moab.messageQueue.secretKey** property is set, then all messages MWS publishes on the message queue will be encrypted. Additionally, MWS can read

messages from Moab Workload Manager that are encrypted with the same key using the **MESSAGEQUEUESECRETKEY** parameter. For more information, see Configuration.

Encryption is done with AES in CBC mode where inputs are padded with PKCS5 padding. Only 128-bit (16-byte) keys are supported. Keys should be encoded in Base64.

For example:

```
moab.messageQueue.secretKey = "1r6RvfqJa6voezy5wAx0hw==" //must be a Base64-encoded
128-bit key
```

> **Important:** If MWS is configured to encrypt the message queue and Moab is not (or vice versa) then the messages from Moab will be ignored. Furthermore, all attempts to access the MWS service resource will fail.

Related Topics

- Resources Introduction
- Events
- Notifications
- Notification Conditions
- Creating Events and Notifications
- Plugin Developer's Guide
- Fields: Events
- Plugin Event Service
- Handling Events
- System Events
- Securing the Connection with the Message Queue

Version and Build Information

To get detailed version information about MWS, use one of the following three methods:

- Browser
- REST Request
- MANIFEST.MF File

Browser

Using a browser, visit the MWS home page (for example, `http://localhost:8080/mws/`). At the bottom of the page is the MWS version information. See the screenshot below:

REST Request

Using a REST client or other HTTP client software, send a GET request to the `rest/diag/about` resource. Here is an example:

```
curl -u username:password http://localhost:8080/mws/rest/diag/about?api-version=3
```

This resource is also described under Diagnostics.

MANIFEST.MF File

If MWS fails to start, version and build information can be found in the `META-INF/MANIFEST.MF` file inside the MWS WAR file. The version properties begin with **Implementation**. Below is an excerpt of a `MANIFEST.MF` file:

```
Implementation-Build: 26
Implementation-Build-Date: 2012-06-19_14-18-59
Implementation-Revision: 376079a5e5f552f2fe25e6070fd2e84c646a98fd

Name: Grails Application
Implementation-Title: mws
Implementation-Version: 7.1.0-rc2
Grails-Version: 2.0.3
```

Related Topics

- Setting up MWS Security

Chapter 2 Access Control

This section describes how to manage access control in MWS. Applications are the consumers of MWS. They include Moab Viewpoint and other applications that need the resources provided by MWS. An application account consists of four editable fields and resource-specific access control settings:

Table 2-1: Field information

Field	Required	Default value	Value type	Maximum length	Description
Application Name	Yes	--	String	32	The name of the application. Must start with a letter and may contain letters, digits, underscores, periods, hyphens, apostrophes, and spaces.
Username	Yes	--	String	32	Used for authentication. Must start with a letter and may contain letters, digits, underscores, periods, and hyphens.
Description	No	--	String	1000	The description of the application.
Enabled	--	true	Boolean	--	Controls whether the application is allowed to access MWS.
Access Control Settings	Yes	All Permissions	--	--	The permissions granted to the application. This is controlled by selecting specific check boxes in a grid.

An application account also contains an auto-generated password that is visible only when creating the account or when resetting its password. Whenever an application sends a REST request to MWS, it needs to pass its credentials (username and password) in a Basic Authentication header. For more information, see Authentication.

The **Application Name** is a human-friendly way to identify an application account, but MWS does not use it during authentication (or at any other time, for that matter).

The **Enabled** field is set to true automatically when an application account is created. To change the value of this field, see Modifying an Application Account.

Here is an example of how you might set the fields when creating an application account:

- **Application Name**: Moab Viewpoint
- **Username**: viewpoint
- **Description**: This application account grants access to Moab Viewpoint.

The permissions granted to an application account may be customized while creating or modifying the account. For more information, see Creating an Application Account and Modifying an Application Account.

Managing Application Accounts

Application accounts are used to grant access to MWS. Every application with an application account must be granted at least one access control permission to a resource in MWS. To manage application accounts, see Listing Application Accounts.

Listing Application Accounts

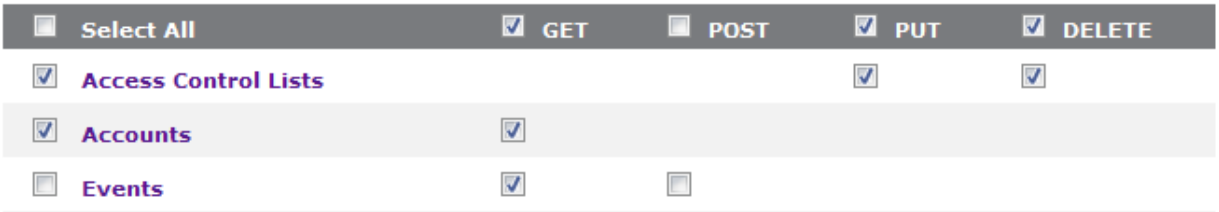
To list all applications accounts, browse to the MWS home page (for example, `https://servername/mws`). Log in as the admin user, click **Admin** and then **Application Accounts**.

Each column (except **Password**) can be sorted in ascending or descending order by clicking on the column heading.

Creating an Application Account

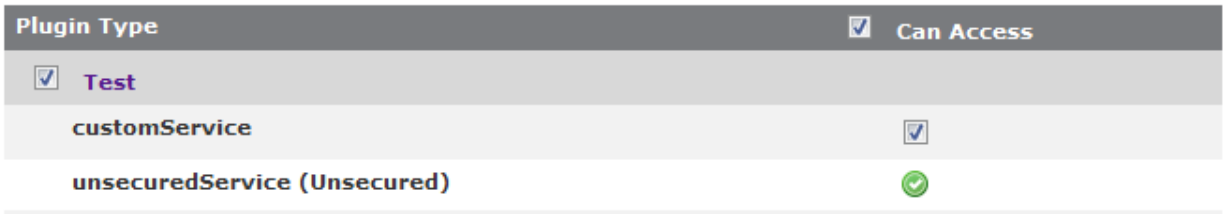
To create an application account, go to the **Application List** page and click **Add Application**. The "Application Name" and "Username" are required fields. For more details, see Field information.

Access to specific resources and plugin custom web services is granted or revoked by checking or unchecking the check boxes in the respective resources or plugin web services access control sections. For each resource, access may be granted to a resource for each method supported by MWS, including GET, POST, PUT, and DELETE. See the figure below for an example.



In this example, the application has access to all available methods for the **Access Control Lists** and **Accounts** resources as well as to retrieve the **Events** resource through the GET method, but is denied the permission to create new events through the POST method.

Access may also be granted to each plugin type's custom web service(s). When new plugin types or plugin web services are added to MWS, applications must be updated with the new access control settings. See below for an example.

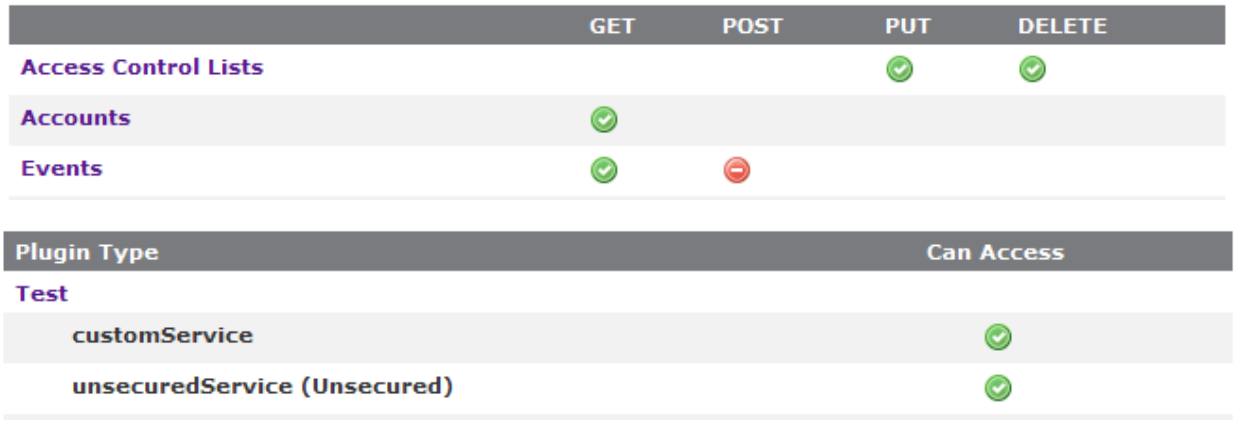


In this example, the application has access to all the custom web services defined for the **Test** plugin type. Note that though unsecured web services are listed, access to them cannot be denied (for more information, see Exposing Web Services).

Displaying an Application Account

To show information about an application account, go to the **Application List** page and click the desired application name.

In addition to displaying the values for fields, grids are also displayed which represent the application's access control permissions defined for resources and plugin custom web services. Examples of the resources and the plugin web services access control displays are shown below:



Modifying an Application Account

To modify an application account, go to the **Application List** page, click the desired application name, and then click **Edit**. See Creating an Application Account for more information on available fields and access control settings.

Resetting an Application Password

To reset an application password, go to the **Application List** page and click the **Reset** link for the desired application. Alternatively, go to the **Display**

Application page for the desired application and click the **Reset** link.

Deleting an Application Account

To delete an application account, go to the **Application List** page, click the desired application name, and then click **Delete**. A confirmation message is shown. If the **OK** button is clicked, the application account is deleted from the system and cannot be recovered.

Related Topics

- Moab Web Services Overview
- Setting up MWS Security

Chapter 3 About the API

Moab Web Services provide a set of RESTful resources that can be used to create, read, update, and delete various objects in the Moab Workload Manager. This section describes how to use RESTful web services, explains the JSON data format used for all communications with MWS, describes global URL parameters used in MWS calls, and contains other helpful information for using the Moab Web Services API.

This section contains these topics:

- RESTful Web Services
- Data Format
- Global URL Parameters
- Requesting Specific API Versions
- Responses and Return Codes
- Error Messages
- Pre- and Post-Processing Hooks
- Authentication

Related Topics

- Resources Introduction
- About Moab Web Services Plugins

RESTful Web Services

In order to understand how to use MWS, it is first necessary to give a brief introduction to REST. REST (Representational State Transfer) is a set of guidelines which utilizes the full HTTP (Hypertext Transfer Protocol) specification along with endpoint URLs that describe resources. The HTTP methods used in REST are comprised of the following:

Method	Description
GET	Query for a list or a single resource.
POST	Creating a resource.
PUT	Modifying a resource.

Method	Description
DELETE	Deleting a resource.

In comparison to other architectures of web services which use a single HTTP method and service endpoint to perform multiple types of operations (such as a POST operation to a URL), REST utilizes all of the available HTTP methods and URLs that directly correlate to resources. For example, RESTful web services for books in a library may expose many URL endpoints and the HTTP methods available for each such as GET, POST, PUT, and DELETE. The list below gives the methods, URLs, and descriptions for a sample set of services. The number 1 represents a unique identifier for books in each case.

Method	URL	Description
GET	`/books`	Retrieves a list of all books in the library.
POST	`/books`	Creates a new book.
GET	`/books/1`	Retrieves a single book.
PUT	`/books/1`	Modifies a single books.
DELETE	`/books/1`	Deletes a single book.

Note that in the cases of the POST and PUT operations, additional information may be needed to describe the resource to be created or the fields that should be modified.

Moab Web Services provides RESTful web services for many resources. The methods and URLs available are documented in Resources Introduction.

Related Topics

- About the API

Data Format

JSON (JavaScript Object Notation) is the data format used for all communication with MWS. This format makes use of two main structures: collections of key/value pairs called *objects* and ordered lists of values called *arrays*. Objects are defined by using curly braces ({}), and arrays are defined by using square brackets ([]). A JSON object or array may contain several

different types of values including numbers, booleans (true/false), strings, objects, arrays, or the keyword 'null' representing no value. For example, a simple JSON object might be defined as:

```
{
  "number": 1,
  "decimalNumber": 1.2,
  "boolean": true,
  "string": "Any string",
  "dateString": "2013-05-23 17:32:02 UTC",
  "object": {
    "key": "value"
  },
  "array": [
    "value1",
    "value2"
  ],
  "nullValue": null
}
```

Dates in MWS, for both input and output, use the pattern "`yyyy-MM-dd HH:mm:ss ZZZ`". For more details on that pattern, see Joda-Time DateTimeFormat. For a list of valid time zone IDs, see Joda-Time Available Time Zones.

For more information on JSON, see json.org.

The data format of MWS is defined as follows:

- Input for a POST or PUT must be in JSON format. Set the `Content-Type` header to `application/json`.
- Output is in JSON format and always consists of an object with zero or more key/value pairs.
- The output may also be "pretty-printed" or formatted for human viewing by sending a URL parameter. For more information, see Global URL Parameters.

Related Topics

- About the API

Global URL Parameters

All URL parameters are optional.

Parameter	Valid values	Description
api-version	Integer	Requests a specific API version

Parameter	Valid values	Description
pretty	*true*	Controls pretty printing of output
fields	Comma-separated string	Includes only specified fields in output
exclude-fields	Comma-separated string	Excludes specified fields from output
max	Integer	The maximum number of items to return
offset	Integer	The index of the first item to return

API Version (api-version)

See Requesting Specific API Versions for information on this parameter and how it should be used.

Pretty (pretty)

By default, the output is easy for a machine to read but difficult for humans to read. The **pretty** parameter formats the output so that it is easier to read.

Field Selection (fields)

The **fields** parameter will include *only* the specified fields in the output. For list queries, the field selection acts on the objects in **results** and not on the **totalCount** or **results** properties themselves.

The format of the **fields** parameter is a comma-separated list of properties that should be included, as in `id,state`. Using periods, sub-objects may also be specified, and fields of these objects may be included as well. This is done with the same syntax for both single sub-objects and lists of sub-objects, as in `id,requirements.requiredNodeCountMinimum,blockReason.message`.

Example 3-1: Example for a job query

```
Request
------------------------------

GET /rest/jobs?api-
version=3&fields=name,flags,requirements.taskCount,dates.createdDate
```

```
Response
-------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": [  {
    "dates": {"createdDate": "2012-10-17 01:11:54 UTC"},
    "flags": ["GLOBALQUEUE"],
    "name": "Moab.24",
    "requirements": [{"taskCount": 1}]
  }]
}
```

Field Exclusion (exclude-fields)

The **exclude-fields** parameter is the opposite of the **fields** parameter. All fields will be included in the output *except* those that are specified. For list queries, the field exclusion acts on the objects in **results** and not on the **totalCount** or **results** properties themselves.

The format of the **exclude-fields** parameter is a comma-separated list of properties that should be excluded from the output, as in `id,state`. Using periods, sub-objects may also be specified, and fields of these objects may be excluded as well. This is done with the same syntax for both single sub-objects and lists of sub-objects, as in `id,requirements.requiredNodeCountMinimum,blockReason.message`.

Example 3-2:

Suppose a query returns the following JSON:

```
Request with No Field Exclusion
-------------------------------

GET /objects
```

```
Response
-------------------------------

{
  "id": "1",
  "listOfStrings":   [
    "string1",
    "string2"
  ],
  "listOfObjects": [  {
    "item1": "value1",
    "item2": "value2"
  }],
  "singleObject":   {
    "id": "obj1",
    "field1": "value1"
  }
}
```

The same query with exclude-fields would return the following output:

```
Request with No Field Exclusion
-------------------------------

GET /objects?exclude-fields=id,listOfObjects.item2,singleObject.field1,listOfStrings
```

```
Response
-------------------------------

{
  "listOfObjects": [{"item1": "value1"}],
  "singleObject": {"id": "obj1"}
}
```

Sorting (sort)

Images and Events support sorting based on MongoDB syntax by using the sort parameter. To sort in ascending order, specify a `1` for the sorting field. To sort in descending order, specify a `-1`. Objects can also be sorted on nested fields by using dot notation to separate the sub-fields, such as `field.subfield1.subfield2`.

Related Topics

- About the API

Requesting Specific API Versions

Because of significant changes in the API introduced in release version 7.2.0, MWS possesses a versioned API. The api-version URL parameter may be used to change the requested API version for any call to MWS. The current valid API versions with their corresponding MWS versions are shown in the table below:

API version	MWS version	Documentation	Additional notes
2 (deprecated)	7.2.x	7.2.x documentation on http://-docs.adaptivecomputing.com/	As of the 8.0 release, API version 2 is **officially deprecated** and will be removed from Moab Web Services in a future release.
3	8.0	Contained within this document	--
latest	Latest	Contained within this document	When the `latest` API version is requested, it resolves to the latest API version of MWS, such as api-version=3 for MWS 8.0.

If no API version is specified, the request is rejected. An API version must be specified with every call in Moab Web Services 8.0 and later.

Resources Introduction and Resources reference contain information for the latest API version. For documentation of previous API versions, please see the table above.

Examples

```
GET http://localhost:8080/mws/rest/nodes?api-version=2
// Data returned uses API version 2

GET http://localhost:8080/mws/rest/nodes?api-version=latest
// Data returned uses API version 3
```

Related Topics

- About the API

Responses and Return Codes

Various HTTP responses and return codes are generated from MWS operations. These are documented below according to the operation that they are associated with.

- Listing and Showing Resources
- Creating Resources
- Modifying Resources
- Deleting Resources
- Moab Response Headers

Listing and Showing Resources

For any successful list or show operation `(GET)`, a `200 OK` response code is always returned. No additional headers beyond those typical of a HTTP response are given in the response.

The body of this response consists of the results of the list or show operation. For a list operation, the results are wrapped in metadata giving total and result counts. The result count represents the number of resource records returned in the current request, and the total count represents the number of all records available. These differ when querying or the `max` and `offset` parameters are used. The following is an example of a list operation response:

```
JSON List Response Body
-----------------------------------

{
  "resultCount":1,
  "totalCount":5,
  "results":[
    {
      "id":"Moab.1",
      …
    }
  ]
}
```

For a show operation, the result is given as a single object:

```
JSON Show Response Body
-----------------------------------

{
  "id":"Moab.1",
  …
}
```

Creating Resources

A successful creation (POST) of a resource has two potential response codes:

- If the resource was created immediately, a 201 Created response code is returned.
- If the resource is still being created, a 202 Accepted response code is returned.

In either case, a `Location` header is added to the response with the full URL which can be used to get more information about the newly created resource or the task associated with creating the resource (if a `202` is returned).

Additionally, the body of the response will contain the unique identifier of the newly created resource or the unique identifier for the task associated with creating the resource (if a `202` is returned).

For example, during creation or submission of a job, a `201` response code is returned with the following response headers and body:

```
Job Creation Response Headers
-----------------------------------

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Location: /mws/rest/jobs/Moab.21
X-Moab-Status: Success
X-Moab-Code: 000
Content-Type: application/json;charset=utf-8
Content-Length: 16
Date: Wed, 21 Dec 2011 23:04:47 GMT
```

```
Job Creation Response Body
-----------------------------------

{"id":"Moab.21"}
```

Modifying Resources

For any successful resource modification operation (`PUT`), a `200 OK` or `202 Accepted` response code is returned. A `200` response code signifies that the modification was immediately completed. No additional headers are returned in this case. A `202` response code is used again to signify that the modification is not yet complete and additional actions are taking place. In this case, a `Location` header is also returned with the full URL of the resource describing the additional actions.

In the case of a `200` response code, the body of this response typically consists of an object with a single `messages` property containing a list of statuses or results of the modification(s). However, a few exceptions to this rule exist as documented in Resources Introduction. In the case of a `202` response code, the format is the same as for a `202` during a creation operation, in that the body consists of an object with the unique identifier for the task associated with the additional action(s).

For example, when modifying a job, several messages may be returned as follows with the associated `200` response code.

```
Job Modification Response Headers
-----------------------------------

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Moab-Status: Success
X-Moab-Code: 000
X-Moab-Message:
Content-Type: application/json;charset=utf-8
Content-Length: …
Date: Thu, 22 Dec 2011 16:49:43 GMT
```

```
JSON Modify Response Body
-----------------------------------

{
  "messages":[
    "gevent processed",
    "variables successfully modified"
  ]
}
```

Deleting Resources

For any successful resource deletion operation (`DELETE`), a `200 OK` or `202 Accepted` response code is returned. A `200` response code signifies that the deletion was immediately completed. No additional headers are returned in this case. A `202` response code is used again to signify that the deletion is not

yet complete and additional actions are taking place. In this case, a `Location` header is also returned with the full URL of the resource describing the additional actions.

In the case of a `200` response code, the body of this response is empty. In the case of a `202` response code, the format is the same as for a `202` during a creation operation, in that the body consists of an object with the unique identifier for the task associated with the additional action(s).

For example, when deleting a job, a `200` response code is returned with an empty body as shown below.

```
Job Deletion Response
-------------------------------------

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Moab-Status: Success
X-Moab-Code: 000
X-Moab-Message:
Content-Type: application/json;charset=utf-8
Content-Length: 0
Date: Thu, 22 Dec 2011 16:49:43 GMT
```

Moab Response Headers

In addition to the typical HTTP headers and the Location header described above, several headers are returned if the operations directly interact with Moab. These headers are described in the following table:

Name	Description
X-Moab-Status	One of `Success`, `Warning`, or `Failure`. Describes the overall status of the Moab request.
X-Moab-Code	A three digit code specifying the exact error encountered, used only in debugging.
X-Moab-Message	An optional message returned by Moab during the request.

Related Topics

- About the API

Error Messages

Below is an explanation of what error message format to expect when an HTTP status code other than 20x is returned. All error codes have a response code of 400 or greater.

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error

400 Bad Request

This response code is returned when the request itself is at fault, such as when trying to modify a resource with an empty `PUT` request body or when trying to create a new resource with invalid parameters. The response body is as follows:

```
{
  "messages":[
    "Message describing error",
    "Possible prompt to take action"
  ]
}
```

401 Unauthorized

This response code is returned when authentication credentials are not supplied or are invalid. The response body is as follows:

```
{
  "messages":[
    "You must be authenticated to access this area"
  ]
}
```

403 Forbidden

This response code is returned when the credentials supplied are valid, but the permissions granted are insufficient for the operation. This occurs when using application accounts (see Access Control) with limited access.

```
{
  "messages":[
    "You are not authorized to access this area"
  ]
}
```

404 Not Found

This response code is returned when the request specifies a resource that does not exist. The response body is as follows:

```
{
  "messages":[
    "The resource with id 'uniqueId' was not found"
  ]
}
```

405 Method Not Allowed

This response code is returned when a resource does not support the specified HTTP method as an operation. The response body is as follows:

```
{
  "messages":[
    "The specified HTTP method is not allowed for the requested resource"
  ]
}
```

500 Internal Server Error

This indicates that there was an internal server error while performing the request, or that an operation failed in an unexpected manner. These are the most serious errors returned by MWS. If additional information is needed, the MWS log may contain further error data. The response body is as follows:

```
{
  "messages":[
    "A problem occurred while processing the request",
    "A message describing the error"
  ]
}
```

Related Topics

- About the API

Pre- and Post-Processing Hooks

MWS provides functionality to intercept and modify data sent to and returned from web services for all available resources. This is done by creating hooks in Groovy files located in a sub-directory of the `MWS_HOME` directory (by default, `/opt/mws/hooks`).

> Please see Reference in this topic for the full reference for available hooks and methods available to them.

- Configuring Hooks
- Defining Hooks for a Resource
- Before Hooks

- After Hooks
- Error Handling
- Defining Common Hooks
- Reference

Configuring Hooks

The directory of the `hooks` folder may be changed by providing a value for `mws.hooks.location` in the configuration file. If the directory starts with a path separator (ie `/path/to/hooks`), it will be treated as an absolute path. Otherwise, it will be used relative to the location of the MWS home directory (for more information, see Configuring Moab Web Services).

For example, if the MWS home directory is set to `/opt/mws`, the hooks directory by default would be in `/opt/mws/hooks`. Changing the `mws.hooks.location` property to `myhooks` would result in the hooks directory being located at `/opt/mws/myhooks`. Due to the default location of the MWS home directory, the default directory of the `hooks` directory is `/opt/mws/hooks`.

On startup, if the `hooks` directory does not exist, it will be created with a simple `README.txt` file with instructions on how to create hooks, the objects available, and the hooks available. If the folder or file is unable to be created, a message will be printed on the log with the full location of a README file, copied into a temporary directory.

Defining Hooks for a Resource

Hooks are defined for resources by creating groovy class files in the hooks directory (`MWS_HOME/hooks` by default). Each groovy file must be named by the resource URL it is associated with and end in ".groovy". The following table shows some possible hook files that may be created. Notice that the virtual machines hook file is abbreviated as `vms`, just as the URL for virtual machines is `/rest/vms`. In most cases, the hook file names will exactly match the URLs. However, in cases of nested URLs—such as with "accounting/users"—the hook file name must replace slashes with periods. For example:

Resource	Hook filename
Jobs	`jobs.groovy`
Nodes	`nodes.groovy`
Virtual Machines	`vms.groovy`

Resource	Hook filename
Accounting Users	`accounting.users.groovy`
Accounting Funds Reports Statement	`accounting.funds.reports.statement.groovy`
Accounting Charge Rates	`accounting.charge-rates.groovy`
url	*url.groovy*

`plugins.rm.groovy` is a valid hook filename. It works for the following URL: `/rest/plugins/<pluginID or all>/rm/<query or action>` (for example, `/rest/plugins/plugin1/rm/cluster-query`).

A complete example of a hook file is as follows:

```
Complete Hook File
-----------------------------------

// Example before hook
def beforeList = {
  // Perform actions here
  // Return true to allow the API call to execute normally
  return true
}

def beforeShow = {
  // Perform actions here
  // Render messages to the user with a 405 Method Not Allowed
  // HTTP response code
  renderMessages("Custom message here", 405)
  // Return false to stop normal execution of the API call
  return false
}

// Example after hook
def afterList = { o ->
  if (!isSuccess()) {
    // Handle error here
    return false
  }
  // Perform actions here
  return o
}
```

You must convert all actions or queries that are separated by dashes to a camel case. For example, the hooks called for "cluster-query" should be `beforeClusterQuery` and `afterClusterQuery`.

As the specific format for the hooks for `before` and `after` are different, each will be explained separately.

Before Hooks

As shown above, before hooks require no arguments. They can directly act on several properties, objects, and methods as described in Reference. The return value is one of the most important aspects of a `before` hook. If it is `false`, a `renderMessages`, `renderObject`, `renderList`, `render`, or `redirect` method *must* first be called. This signifies that the API call should be interrupted and the render or redirect action specified within the hook is to be completed immediately.

A return value of `true` signifies that the API call should continue normally. Parameters, session variables, request and response variables may all be modified within a `before` hook.

> If no return value is explicitly given, the result of the last statement in the `before` hook to be executed will be returned. This may cause unexpected behavior if the last statement resolves to `false`.

For all methods available to `before` hooks as well as specific examples, see beforeSave.

After Hooks

`After` hooks are always passed one argument: the object or list that is to be rendered as JSON. This may be modified as desired, but note that the object or list value is either a JSONArray or JSONObject. Therefore, it may not be accessed and modified as a typical groovy Map.

Unlike `before` hooks, `after` hooks should not call the `render*` methods directly. This method will automatically be called on the resulting object or list returned. The `redirect` and `render` methods should also not be called at this point. Instead, if a custom object or list is desired to be used, the `serializeObject` and `serializeList` methods are available to create suitable results to return.

The return value of an `after` hook may be one of two possibilities:

- The potentially modified object or list passed as the first argument to the hook. In this case, this value will override the output object or list unless it is null.
- Null or false. In this case, the original, unmodified object or list will be used in the output.

> The return value of the `after` hook, if not null or false, *must* be the modified object passed into the hook or an object or list created with the `serialize*` methods.

For all methods available to after hooks as well as specific examples, see afterSave.

Error Handling

`After` hooks, unlike the `before` hooks, have the possibility of handling errors encountered during the course of the request. Handling errors is as simple as adding a one-line check to the hook as shown above or in the following code:

```
if (!isSuccess()) {
  // Handle error
  return false
}
```

It is recommended that each `after` hook contain at least these lines of code to prevent confusion on what the input object or list represents or should look like.

The `isSuccess()` function is false if and only if the HTTP response code is 400 or higher, such as a 404 Not Found, 400 Bad Request, or 500 Internal Server Error and the cause of the error state was not in the associated `before` hook. In other words, objects and lists rendered in the `before` hook with any HTTP response code will never run the associated `after` hook.

When handling errors, the passed in object will always contain a `messages` property containing a list of strings describing the error(s) encountered.

Defining Common Hooks

Sometimes it is beneficial to create hooks which are executed for all calls of a certain type, such as a `beforeList` hook that is executed during the course of listing any resource. These are possible using an `all.groovy` file. The format of this file is exactly the same as other hook files. The order of execution is as follows:

1. `Before` common hook executed.
2. `Before` resource-specific hook executed.
3. Normal API call executed.
4. `After` resource-specific hook executed.
5. `After` common hook executed.

Reference

This page gives specific examples and reference for implementing hooks in MWS.

Available hooks

The following table lists the available hooks for each resource with their associated HTTP method and description.

Name	HTTP method	Description
beforeList	GET	Runs before an API call that lists resources (for example, `GET /rest/jobs`).
afterList	GET	Runs after an API call that lists resources.
beforeShow	GET	Runs before an API call that returns a single resource (for example, `GET /rest/jobs/job.1`).
afterShow	GET	Runs after an API call that returns a single resource.
beforeSave	POST	Runs before an API call that saves a new resource (for example, `POST /rest/-jobs`).
afterSave	POST	Runs after an API call that returns a single resource.
beforeUpdate	PUT	Runs before an API call that returns a single resource (for example, `PUT /rest/jobs/job.1`).
afterUpdate	PUT	Runs after an API call that returns a single resource.
beforeDelete	DELETE	Runs before an API call that returns a single resource (for example, `DELETE /rest/jobs/job.1`).
afterDelete	DELETE	Runs after an API call that returns a single resource.

If a resource does not support a certain operation, any hooks for that operation will simply be ignored—such as **beforeSave** and **afterSave** hooks for the Node resource, where saving is not supported.

Available properties

The following table lists the properties, objects, and methods available in all hooks. Note that although it is possible to directly call the `render*` methods in the `after` hooks, it is not recommended.

Name	Type	Description
params	Map	Contains all URL parameters as well as the body of the request as parsed JSON.

Name	Type	Description
request	HttpServletRequest	Contains properties of the HTTP request.
response	HttpServletResponse	Contains properties of the HTTP response which can be modified directly.
session	HttpSession	Contains the session parameters which can be modified directly.
flash	Map	Temporary storage that stores objects within the session for the next request only.
controllerName	String	The name of the controller responding to the request. Only available in `before` hooks.
actionName	String	The name of the action to be run on the controller. Only available in `before` hooks.
apiVersion	String	The API version for the current request (for example, 1 for 7.0 and 7.1, 2 for 7.2).

The parsed JSON may be accessed in `before` hooks as a simple groovy Map with `params[controllerName]`.

In addition, several methods are available to the hooks. These are described in the following sections.

Redirect

The `redirect` method may be used to redirect the request to another API call or an arbitrary URL.

```
redirect(uri:'/rest/jobs')      // uri is used for internal redirection within MWS
redirect(url:'http://adaptivecomputing.com')   // url is used for external redirection
redirect(uri:'http://adaptivecomputing.com', params:[lang:'en'])       // params may be used
for URL parameters
```

The `redirect` method will use the GET HTTP method for the resulting redirected request.

See the `redirect` method's documentation for more information.

Rendering objects, lists, or messages

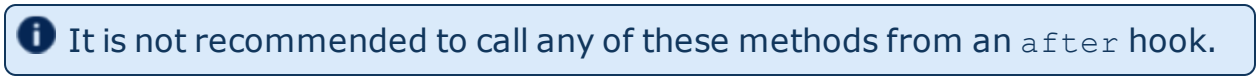
There are several `render*` methods available to handle any case where objects or lists are desired to be rendered directly from the hook without

continuing to the API call. Three different methods may be used depending on the desired output object type:

```
Render object
-----------------------------------

// Object that should be rendered as JSON
def objectToRender = …
// HTTP response code (bad request)
def responseCode = 400
// Render a simple object
renderObject(objectToRender)
// Render a simple object with a custom response code
renderObject(objectToRender, responseCode)
```

```
Render list
-----------------------------------

// List that should be rendered as JSON
def listToRender = …
// If the totalCount property differs from resultCount, use this value instead
def totalCount = …
// HTTP response code (bad request)
def responseCode = 400
// Render a simple list
//     Dynamically adds "resultCount" and "totalCount" properties based on the size of
the input list
renderList(listToRender)
// Render a simple list with a custom "totalCount"
renderList(listToRender, totalCount)
// Render a simple list without changing the "totalCount" but with a custom response
code
renderList(listToRender, null, responseCode)
// Render a simple list with a custom "totalCount" and response code
renderList(listToRender, totalCount, responseCode)
```

```
Render message(s)
-----------------------------------

// Messages
def messageToRender = "Single message"
def messagesListToRender = ["Message 1", "Message 2"]
// HTTP response code (bad request)
def responseCode = 400
// Render messages as an object with a property of "messages" containing a list of the
messages passed in
renderMessages(messageToRender)
renderMessages(messageToRender, responseCode)
// Supports either a single String or list of Strings
renderMessages(messagesListToRender)
renderMessages(messagesListToRender, responseCode)
```



Render

Less commonly used, the `render` method is also available directly. This may be used to render text directly, change the content-type of the output, and many

other functions. See the `render` method's documentation for more information.

> It is not recommended to call this method from an `after` hook.

Serialize objects

The `serializeObject` and `serializeList` methods may be used to convert a custom object or list respectively into a format usable for returning in the `after` hooks. Simply pass in the object or list and a serialized version will be returned from the method.

```
def afterShow = {
       def objectToRender = …
       def serializedObject = serializeObject(objectToRender)
       return serializedObject
}
```

```
def afterShow = {
       def listToRender = [...]
       def serializedList = serializeList(listToRender)
       return serializedList
}
```

Error handling

Error handling is only available in `after` hooks by using the following check:

```
if (!isSuccess()) {
       // Handle error
       return … // False or modified object/list to render
}
```

Usage examples

Override an API call

The following hook would serve to override an entire API call, the list call in this case, and return a `messages` list containing a single element of "Action is not supported" and a HTTP response code of 405 (Method Not Allowed):

```
def beforeList = {
       renderMessages("Action is not supported", 405)
       return false
}
```

To be even more specific and disallow the deletion of virtual machines, the following may be used as the `vms.groovy` file:

```
def beforeDelete = {
       renderMessages("Virtual Machine deletion is not allowed", 405)
       return false
}
```

Add an additional property during job creation

To add an additional property to a job definition during creation, create a `beforeSave` hook in the `jobs.groovy` file as follows:

```
def beforeSave = {
        // params[controllerName] is equivalent to params["job"] or params.job
        params[controllerName].user = "myuser"
}
```

This would cause the created job to have a user of `myuser`.

Redirect based on URL parameter

To redirect an API call if a certain URL parameter exists, create a `beforeSave` hook in the `jobs.groovy` file as follows:

```
def beforeSave = {
        if (params.external) {
                redirect(url:'http://example.com/create-job')
                return false;   // Stop API call
        }
}
```

This would cause an API call of `PUT /rest/jobs?external=1` to redirect to `GET http://example.com/create-job`.

Remove a property from getting a single job

To remove a property from the output of getting a single job, create an `afterShow` hook in the `jobs.groovy` file as follows:

```
def afterShow = { o ->
        o.discard("group")
        return o
}
```

This will cause the resulting JSON to be missing the `group` property of the job resource. Note again that these calls must use the JSONArray and JSONObject classes as mentioned in After Hooks.

Filter list items

To filter the items in a list nodes request based on user provided query parameter in the URL, use the following in the `nodes.groovy` file. A sample request that would activate the filter is `http://localhost:8080/mws/rest/nodes?api-version=3&filter-power=On`.

```
def afterList = { o ->
       // Do not filter if the user did not ask for it
       if (!params['filter-power'])
              return o
       // o = {resultCount: x, totalCount: x, results:[...]}

       // Using a built-in groovy method findAll to return all
       //      list items that return true from the block
       def results = o.results.findAll { node ->
              // Includes the node only if the power equals the user input
              return params['filter-power'].equalsIgnoreCase(node.power)
       }

       // Sets the results on the return object and updates the counts
       o.element("results", results)
       o.element("resultCount", results.size())
       return o
}
```

To filter the items in a list nodes request based on values within the list itself, such as variable values, use the following in the `nodes.groovy` file.

```
def afterList = { o ->
       // o = {resultCount: x, totalCount: x, results:[...]}
       // Using a built-in groovy method findAll to return all
       //      list items that return true from the block
       def results = o.results.findAll { node ->
              // Includes the node only if the variable "included" is set to "true"
              return node.variables?.included=="true"
       }

       // Sets the results on the return object and updates the counts
       o.element("results", results)
       o.element("resultCount", results.size())
       return o
}
```

Related Topics

- About the API

Authentication

MWS uses Basic Authentication for all REST API requests. This means that a username and password must be provided for each call to resources. There are two types of accounts that can be granted access: **Users** and **Applications**.

- For instructions on how to set the credentials for the default **User** account, see Setting up MWS Security.
- For instructions on how to manage **Application** accounts, see Access Control.

To use Basic Authentication, each client request must contain a header that looks like this:

```
Authorization: Basic YWRhcHRpdmU6YzNVU3R1bkU=
```

The string after the word `Basic` is the base64 encoding of *username : password*. In the example above, `YWRhcHRpdmU6YzNVU3R1bkU=` is the base64 encoding of `adaptive:c3UStunE`. For more details, see section 2 of RFC 2617.

> The username and password in the Basic Authentication header are encoded but not encrypted. Therefore, it is *strongly* recommended that MWS be run behind a proxy (like Apache) with SSL enabled. For more information, see Setting up MWS Security.

Related Topics

- About the API

System Events

The broad category of system events may be broken down into two subcategories: events and notification conditions.

- Events
- Notification Conditions

Events

Events are created by many components in the system, but most events originate from Moab Workload Manager and Moab Web Services. Events can be created via the MWS interface or by being placed on the message queue. The ZeroMQ™ message queue libraries were introduced in Moab and MWS 7.5.0. The message queue is critical to service lifecycle functionality (see "Service lifecycle" in the *Moab Cloud Suite Installation Guide*).

In a typical system, Moab will communicate events to MWS via a "private" message queue, and then MWS will replicate the events on the "public" message queue, or the message queue that is available to subscribers with the correct secret keys. In some cases, such as those related to the MWS service lifecycle, MWS uses events to determine activities or capabilities that are available.

A typical message on the message queue may look like the following (sent with a topic of `system.moab`):

```
Sample message on message queue
-----------------------------------

{
       "body" : {
              "associatedObjects" : [
                     {
                            "id" : "Moab",
                            "type" : "scheduler"
                     }
              ],
              "code" : 16777619,
              "eventDate" : "2014-02-28T10:57:21.000-0700",
              "message" : "A scheduler iteration is ending.",
              "origin" : "MSysMainLoop.c, MSysMainLoop, line 959"
       },
       "messageId" : "843269550",
       "messageType" : "event",
       "senderId" : "mwm@mwm-server",
       "sentDate" : "2014-02-28T10:57:21.000-0700",
       "ttl" : 3000
}
```

Notification Conditions

Notification Conditions are related to an event, but differ in three distinct areas:

1. Notification conditions are a persistent condition of the system or a component rather than a single occurrence.
   - They are ongoing rather than reoccurring, which is why they are generated from NotificationConditions.
   - They may be observed many times, but the condition is always the same.
   - A good test for this is if something "is" wrong rather than something "went" wrong.
2. Notification conditions can be acted on to result in a resolved state, mean the administrator or user can and must take actions to "fix" the condition or problem.
3. Notification conditions contain state information based on administrator or user input, meaning that they contain information about the condition (similar to events), but also contain the "status" of the administrator's view of the notification, whether it is currently open, dismissed, or ignored.

In general, questions may be asked to ascertain whether an event or a notification condition is the right fit for an occurrence. These questions, along with some sample situations, are provided below.

- Is the occurrence the root cause of a potentially ongoing condition?
  - A VM migration failed because the VM's state was unknown. The root cause was that the state was unknown, not that the VM migration failed. Therefore, VM migration failed would be an event, while the

unknown state would be a notification condition.

  - A VM service provision fails because there are no hypervisors that satisfy the requirements. This would be an event. Note that there may be a notification related to this failure, such as a service template requires a feature that does not exist on any hypervisors in the system, but this would be distinctly detected and managed from the provision failure event.
  - A request to MWS failed because the connection between Moab and MongoDB was misconfigured. The failed request may be represented as an event, but a notification condition should exists that the connection between Moab and MongoDB was down.

- Can an administrator or user affect the outcome of the occurrence?
  - The outcome of a VM migration failing is in the past and cannot be changed by the administrator. However, the outcome of a future VM migration may be changed when the administrator resolves the root problem (such as VM state is unknown).

Related Topics

- Events
- Notifications
- Notification Conditions
- Securing the Connection with the Message Queue
- Creating Events and Notifications (for plugin development only)
- Plugin Event Service

Chapter 4 Resources Introduction

The sections in this chapter show the MWS resources and the HTTP methods defined on them. The prefix for these resources depends on how the `mws.war` file is deployed. A typical prefix would be `http://localhost:8080/mws`. Using this example, one absolute resource URI would be `http://localhost:8080/mws/rest/jobs`.

This section only contains documentation for the latest API version. Please see the table in Requesting Specific API Versions for links to documentation for previous versions.

This chapter contains these sections:

- Access Control Lists (ACLs)
- Accounting Accounts
- Accounting Allocations
- Accounting Charge rates
- Accounting Funds
- Accounting Liens
- Accounting Organizations
- Accounting Quotes
- Accounting Transactions
- Accounting Usage Records
- Accounting Users
- Credentials
- Diagnostics
- Distinct
- Events
- Images
- Insight Database
- Job Arrays
- Jobs
- Job Templates
- Metric Types
- Nodes

- Notifications
- Notification Conditions
- Permissions
- Plugins
- Plugin Types
- Policies
- Principals
- Priority
- Reports
- Reservations
- Resource Types
- Roles
- Standing Reservations

Related Topics

- Resources reference

Access Control Lists (ACLs)

This topic describes behavior of the ACL Rules (Access Control List Rules) object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Access Control Lists (ACLs)** reference contains the type and description of all fields in the **ACL Rules** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

ACLs are not directly manipulated through a single URL, but with sub-URLs of the other objects such as Virtual Containers and Reservations.

Resource	GET	PUT	POST	DELETE
/rest/reservations/<rsvId>/acl-rules/<aclId>	--	Create or Update ACL	--	Delete ACL
/rest/vcs/<vcId>/acl-rules/<aclId>	--	Create or Update ACL	--	Delete ACL

This topic contains these sections:

- Getting ACLs
- Creating or Updating ACLs
  - Create or Update ACL
- Deleting ACLs
  - Delete ACL

Getting ACLs

Although **ACL Rules** cannot be retrieved directly using the GET method on any of the `acl-rules` resources, **ACL Rules** are attached to supported objects when querying for them. Each supported object contains a field named `aclRules`, which is a collection of the **ACL Rules** defined on that object.

Supported objects

The following is a list of objects that will return **ACL Rules** when queried:

- Reservations
- Standing Reservations

Creating or Updating ACLs

The HTTP PUT method is used to create or update **ACL Rules**. The request body can contain one or more **ACL Rules**. If an **ACL Rule** with the same `type` and `value` exists, then it will be overwritten.

Quick reference

```
PUT http://localhost:8080/mws/rest/reservations/<rsvId>/acl-rules?api-version=3
```

Create or Update ACL

URLs and parameters

```
PUT http://localhost:8080/mws/rest/reservations/<rsvId>/acl-rules?api-version=3
```

Parameter	Required	Type	Valid values	Description
objectId	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Request body

The request body below shows all the fields that are available for the PUT method, along with some sample values.

```
JSON Request Body
-----------------------------------

{"aclRules": [{
  "affinity": "POSITIVE",
  "comparator": "LEXIGRAPHIC_EQUAL",
  "type": "USER",
  "value": "ted"
}]}
```

Sample response

> This message may not match the message returned from Moab exactly, but is given as an example of the structure of the response.

```
JSON Request Body
-----------------------------------

{"messages":["Reservation 'rsv1' successfully modified"]}
```

Samples

Create or update multiple ACLs on a single object:

```
CPUT http://localhost:8080/mws/rest/reservations/system.21/acl-rules?api-version=3
-----------------------------------

{"aclRules": [
    {
    "affinity": "POSITIVE",
    "comparator": "LESS_THAN_OR_EQUAL",
    "type": "DURATION",
    "value": "3600"
  },
    {
    "affinity": "POSITIVE",
    "comparator": "LEXIGRAPHIC_EQUAL",
    "type": "USER",
    "value": "ted"
  }
]}
```

Restrictions

- **ACL Rules** cannot be added to or updated on **Standing Reservations**.

Deleting ACLs

The HTTP DELETE method is used to remove **ACL Rules**.

Quick reference

ACL Rules cannot be removed from **Standing Reservations**.

```
DELETE http://localhost:8080/mws/rest/reservations/<rsvId>/acl-rules?api-
version=3/<aclId>
```

Delete ACL

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/reservations/<objectId>/acl-rules?api-
version=3/<aclId>
```

Parameter	Required	Type	Valid values	Description
objectId	Yes	String	--	The unique identifier of the object from which to remove the **ACL Rule**.
aclId	Yes	String	--	A string representing the **ACL Rule**, with the format `type:value`.

See Global URL Parameters for available URL parameters.

Sample response

This message may not match the message returned from Moab exactly, but is given as an example of the structure of the response.

```
JSON Response
-------------------------------

{"messages":["Successfully modified reservation 'rsv1'"]}
```

Restrictions

- **ACL Rules** cannot be removed from **Standing Reservations**.

Related Topics

- Fields: Access Control Lists (ACLs)
- Resources Introduction

Accounting

Accounting Accounts

This section describes the services available through Moab Web Services for interacting with the **Account** object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Accounts** reference contains the type and description of the default fields for the **Accounts** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/accounts	Get All Accounts	--	--	--
/rest/accounting/accounts/<id>	Get Single Account	--	--	--

This topic contains these sections:

- Getting Accounts
  - Get All Accounts
  - Get Single Account

Getting Accounts

The HTTP GET method is used to retrieve **Accounts** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/accounts?api-version=3
GET http://localhost:8080/mws/rest/accounting/accounts/<id>?api-version=3
```

Get All Accounts

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/accounts?api-version=3&proxy-user=<user>
[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_sort>]|&show-
all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"organization":"sciences"}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,organization`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"organization":1}`

Parameter	Required	Type	Valid values	Description	Example
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/accounts?api-version=3&proxy-
user=amy&fields=id,organization&pretty=true
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
        {
      "organization": "sciences",
      "id": "biology"
    },
        {
      "organization": "sciences",
      "id": "chemistry"
    }
  ]
}
```

Get Single Account

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/accounts/<id>?api-version=3&proxy-
user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object.	--
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Valid values	Description	Example
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,organization`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/accounts/chemistry?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "id": "chemistry",
  "active": true,
  "organization": "",
  "description": "Chemistry Dept",
  "users":   [
        {
      "id": "amy",
      "active": true,
      "admin": false
    },
        {
      "id": "bob",
      "active": true,
      "admin": false
    },
        {
      "id": "dave",
      "active": true,
      "admin": false
    }
  ]
}
```

Related Topics

- Fields: Accounts
- Resources Introduction

Accounting Allocations

This section describes the services available through Moab Web Services for interacting with the **Allocation** object in Moab Accounting Manager. It contains

the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

> The **Fields: Allocations** reference contains the type and description of the default fields for the **Allocation** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/allocations	**Get All Allocations**	--	--	--
/rest/accounting/allocations/<id>	**Get Single Allocation**	--	--	--

This topic contains these sections:

- Getting Allocations
  - Get All Allocations
  - Get Single Allocation

Getting Allocations

The HTTP GET method is used to retrieve **Allocation** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/allocations?api-version=3
GET http://localhost:8080/mws/rest/accounting/allocations/<id>?api-version=3
```

Get All Allocations

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/allocations?api-version=3&proxy-
user=<user>[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_
sort>]|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"active":true}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,fund,amount`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"fund":1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/allocations?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "totalCount": 5,
  "resultCount": 5,
  "results":   [
        {
      "id": 1,
      "fund": 1,
      "startTime": "2013-07-12 22:16:33 UTC",
      "endTime": "infinity",
      "amount": 50000000,
      "creditLimit": 0,
      "initialDeposit": 50000000,
      "allocated": 50000000,
      "active": true,
      "description": ""
    },
        {
      "id": 3,
      "fund": 3,
      "startTime": "2013-07-12 22:16:33 UTC",
      "endTime": "infinity",
      "amount": 0,
      "creditLimit": 20000000,
      "initialDeposit": 0,
      "allocated": 0,
      "active": true,
      "description": ""
    },
        {
      "id": 2,
      "fund": 2,
      "startTime": "2013-07-12 22:16:33 UTC",
      "endTime": "infinity",
      "amount": 30000000,
      "creditLimit": 0,
      "initialDeposit": 30000000,
      "allocated": 30000000,
      "active": true,
      "description": ""
    }
  ]
}
```

Get Single Allocation

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/allocations/<id>?api-version=3&proxy-
user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object.	--
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,fund,amount`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/allocations/1?api-version=3&proxy-
user=amy&pretty=true
-------------------------------------

{
  "id": 1,
  "fund": 1,
  "startTime": "2013-07-12 22:16:33 UTC",
  "endTime": "infinity",
  "amount": 50000000,
  "creditLimit": 0,
  "initialDeposit": 50000000,
  "allocated": 50000000,
  "active": true,
}
```

Related Topics

- Fields: Allocations
- Resources Introduction

Accounting Charge rates

This section describes the services available through Moab Web Services for interacting with the **ChargeRate** object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Charge Rates** reference contains the type and description of the default fields for the **ChargeRates** object.

Resource	GET	PUT	POST	DELETE
/rest/accounting/charge-rates	Get All Charge Rates	--	--	--
/rest/accounting/charge-rates/<name>/<value>	Get Single Charge Rate	--	--	--
/rest/accounting/charge-rates/<name>	Get Single Charge Rate	--	--	--

This topic contains these sections:

- Getting Charge Rates
  - Get All Charge Rates
  - Get Single Charge Rate

Getting Charge Rates

The HTTP GET method is used to retrieve **ChargeRate** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/charge-rates?api-version=3
GET http://localhost:8080/mws/rest/accounting/charge-rates?api-version=3/<name>
[/<value>]
```

Get All Charge Rates

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/charge-rates?api-version=3&proxy-
user=<user>[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_
sort>]|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"name":"QualityOfService"}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,organization`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"organization":1}`

Parameter	Required	Type	Valid values	Description	Example
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/charge-rates?api-version=3&proxy-
user=moab&pretty=true
-----------------------------------

{
  "totalCount": 4,
  "resultCount": 4,
  "results":   [
        {
      "name": "Processors",
      "value": "",
      "amount": "1/s",
      "description": "1 credit per processor-second"
    },
        {
      "name": "QualityOfService",
      "value": "high",
      "amount": "*2",
      "description": "Charge double for high QOS"
    },
        {
      "name": "QualityOfService",
      "value": "low",
      "amount": "*.5",
      "description": "Charge half for low QOS"
    },
        {
      "name": "QualityOfService",
      "value": "",
      "amount": "*1",
      "description": "No extra charge for \"normal\" QOSes"
    }
  ]
}
```

Get Single Charge Rate

A regular charge rate is uniquely specified by both its name and its value. A default charge rate has a null value and is uniquely specified by only its name.

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/charge-rates?api-version=3/<name>
[/<value>]?proxy-user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
name	Yes	String	--	The name of the charge rate.	--
value	No	String	--	The value of the charge rate.	--
fields	No	String	--	Comma-separated list of field names to display.	`fields=name,value,amount`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/charge-rates/QualityOfService/high?api-
version=3&proxy-user=moab&pretty=true
-------------------------------------

{
  "name": "QualityOfService",
  "value": "high",
  "amount": "*2",
  "description": "Charge double for high QOS"
}
```

Related Topics

- Fields: Charge Rates
- Resources Introduction

Accounting Funds

This section describes the services available through Moab Web Services for interacting with the **Fund** object in Moab Accounting Manager. It contains the

URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

> The **Fields: Funds**, **Fields: Fund Balances**, **Fields: Fund Statements**, and **Fields: Fund Statement Summary** reference sections contain the type and description of the default fields in the **Fund** object as well as related objects and reports given in the URLs below.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/funds	Get All Funds	--	--	--
/rest/accounting/funds/<id>	Get Single Fund	--	--	--
/rest/accounting/funds/balances	Get All Fund Balances	--	--	--
/rest/accounting/funds/reports/statement	Get Fund Statement	--	--	--
/rest/accounting/funds/reports/statement/summary	Get Fund Statement Summary	--	--	--

This topic contains these sections:

- Getting Funds
  - Get All Funds
  - Get Single Fund
  - Get All Fund Balances
  - Get Fund Statement
  - Get Fund Statement Summary

Getting Funds

The HTTP GET method is used to retrieve **Fund** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/funds?api-version=3
GET http://localhost:8080/mws/rest/accounting/funds/<id>?api-version=3
GET http://localhost:8080/mws/rest/accounting/funds/balances?api-version=3
GET http://localhost:8080/mws/rest/accounting/funds/reports/statement?api-version=3
GET http://localhost:8080/mws/rest/accounting/funds/reports/statement/summary?api-
version=3
```

Get All Funds

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/funds?api-version=3&proxy-user=<user>
[&active=true][&filter=<filter_options>[&filter-type=<filter_type>]][&query=<query_
conditions>][&fields=<fields_to_display>[&sort=<fields_to_sort>]|&show-all=
(true|false)]
```

Parameter	Required	Type	Description	Example
proxy-user	Yes	String	Perform action as defined MAM user.	`proxy-user=amy`
active	No	Boolean	Lists only active or non-active allocations of the fund. The fund amount becomes the sum of the active/inactive allocations.	`active=true`
filter	No	JSON	Query funds based on defined MAM filter.	`filter={"account":"chemistry"}`
filter-type	No	String	Query funds based on defined MAM filter type.	`filter-type=NonExclusive`

Parameter	Required	Type	Description	Example
query	No	JSON	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"priority":"2","allocation.active":"false"}`
fields	No	String	Comma-separated list of field names to display.	`fields=id,name,amount`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"id":1}`

Parameter	Required	Type	Description	Example
show-all	No	Boolean (`true` or `false`)	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/funds?api-version=3&proxy-
user=amy&fields=id,name,amount&pretty=true
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
        {
      "id": 1,
      "name": "biology",
      "amount": 50000000
    },
        {
      "id": 2,
      "name": "chemistry",
      "amount": 99727
    }
  ]
}
```

Get Single Fund

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/funds/<id>?api-version=3&proxy-
user=<user>[&active=(true|false)][&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Description	Example
id	Yes	String	The unique identifier of the object	--
proxy-user	Yes	String	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Description	Example
active	No	Boolean	Lists only active or non-active allocations of the fund. The fund amount becomes the sum of the active/inactive allocations.	`active=true`
fields	No	String	Comma-separated list of field names to display.	`fields=id,name,amount`
show-all	No	Boolean (`true` or `false`)	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/funds/1?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "id": 1,
  "name": "biology",
  "priority": 0,
  "defaultDeposit": 50000000,
  "description": "",
  "amount": 50000000,
  "allocated": 50000000,
  "initialDeposit": 50000000,
  "creditLimit": 0,
  "allocations":   [
        {
      "id": 1,
      "startTime": "2013-08-21 16:57:53 UTC",
      "endTime": "infinity",
      "amount": 50000000,
      "creditLimit": 0,
      "initialDeposit": 50000000,
      "allocated": 50000000,
      "active": false,
      "description": ""
    }
  ],
  "fundConstraints": [  {
    "id": 1,
    "name": "Account",
    "value": "biology"
  }]
}
```

Get All Fund Balances

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/funds/balances?api-version=3&proxy-
user=<user>[&filter=<filter_options>][&filter-type=<filter_type>]
```

Parameter	Required	Type	Description	Example
proxy-user	Yes	String	Perform action as defined MAM user.	`proxy-user=amy`
filter	No	JSON	Query funds based on defined MAM filter.	`filter={"accoun-t":"chemistry"}`
filter-type	No	String	Query funds based on defined MAM filter type.	`filter-type-e=NonExclusive`

See Global URL Parameters for available URL parameters.

Sample response

The fund balances resource is an aggregation of fund data. For more information, see the Fields: Fund Balances reference section.

```
GET http://localhost:8080/mws/rest/accounting/funds/balances?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results": [
    {
      "id": 2,
      "name": 1204,
      "priority": 0,
      "description": "R&D for Manufacturing",
      "creationTime": "2012-02-02 09:34:42 UTC",
      "amount": 9060000,
      "deposited": 9060000,
      "creditLimit": 0,
      "reserved": 0,
      "allocations": [
        {
          "id": 2,
          "amount": 9060000,
          "creditLimit": 0,
          "deposited": 9060000
        }
      ],
      "fundConstraints": [
        {
          "id": 2,
          "name": "CostCenter",
          "value": 1204
        }
      ],
      "balance": 9060000,
      "available": 9060000,
      "allocated": 9060000,
      "used": 0,
      "percentRemaining": 100,
      "percentUsed": 0
    },
    {
      "id": 5,
      "name": "",
      "priority": 0,
      "description": "",
      "creationTime": "2012-04-03 09:25:47 UTC",
      "amount": 901290219001,
      "deposited": 901290219021,
      "creditLimit": 30,
      "reserved": 84018308897.68,
      "allocations": [
        {
          "id": 6,
          "amount": 901290219001,
          "creditLimit": 30,
          "deposited": 901290219021
        }
      ],
      "fundConstraints": [],
      "balance": 817271910103.32,
      "available": 817271910133.32,
      "allocated": 901290219051,
```

```
      "used": 20,
      "percentRemaining": 100,
      "percentUsed": 0
    }
  ]
}
```

Get Fund Statement

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/funds/reports/statement?api-
version=3&proxy-user=<user>[&filter=<filter_options>][&filter-type=<filter_type>]
[&start-time=<date_string>][&end-time=<date_string>][&context=<context>]
```

Parameter	Required	Type	Description	Example
proxy-user	Yes	String	Perform action as defined MAM user.	`proxy-user=amy`
filter	No	JSON	Query funds based on defined MAM filter.	`filter={"accoun-t":"chemistry"}`
filter-type	No	String	Query funds based on defined MAM filter type.	`filter-type-e=NonExclusive`
start-time	No	Date, -infinity, or now	Filter allocations and transaction after a start time.	`start-time=2012-04-03 15:24:39 UTC`
end-time	No	Date, -infinity, or now	Filter allocations and transactions before an end time.	`end-time=2012-04-03 15:24:39 UTC`
context	No	hpc or cloud	The context to use in Moab Accounting Manager. The **context** parameter overrides the default context set for MAM using the `mam.context` configuration parameter. For more information about this parameter, see Configuration.	`context=hpc`

See Global URL Parameters for available URL parameters.

Sample response

The fund statement report provides a snapshot of the current funds. For more information, see Fields: Fund Statements.

```
GET http://localhost:8080/mws/rest/accounting/funds/reports/statement?api-
version=3&proxy-user=amy&fields=startBalance,endBalance&pretty=true
-----------------------------------

{
  "startBalance":1234.01,
  "endBalance":1000
}
```

Get Fund Statement Summary

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/funds/reports/statement/summary?api-
version=3&proxy-user=<user>[&filter=<filter_options>][&filter-type=<filter_type>]
[&start-time=<date_string>][&end-time=<date_string>]
```

Parameter	Required	Type	Description	Example
proxy-user	Yes	String	Perform action as defined MAM user.	`proxy-user=amy`
filter	No	JSON	Query funds based on defined MAM filter.	`filter={"accoun-t":"chemistry"}`
filter-type	No	String	Query funds based on defined MAM filter type.	`filter-type-e=NonExclusive`
start-time	No	Date, -infin-ity, or now	Filter allocations and trans-action after a start time.	`start-time=2012-04-03 15:24:39 UTC`
end-time	No	Date, -infin-ity, or now	Filter allocations and trans-actions before an end time.	`end-time=2012-04-03 15:24:39 UTC`

See Global URL Parameters for available URL parameters.

Sample response

The fund statement summary is slightly different from the typical fund statement in that the transactions are provided as summaries grouped by object and action. For more information, see Fields: Fund Statement Summary.

```
GET http://localhost:8080/mws/rest/accounting/funds/reports/statement/summary?api-
version=3&proxy-
user=amy&fields=totalCredits,totalDebits,transactions.action,transactions.amount,trans
actions.count&pretty=true
-----------------------------------

{
  "totalCredits":200.02,
  "totalDebits":-100,
  "transactions":[ {
      "action":"Deposit",
      "amount":200.02,
      "count":2
    }, {
      "action":"Charge",
      "amount":-100,
      "count":1
    }
  ]
}
```

Related Topics

- Fields: Funds
- Resources Introduction

Accounting Liens

This section describes the services available through Moab Web Services for interacting with the Lien object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Liens** reference contains the type and description of the default fields for the **Liens** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/liens	**Get All Liens**	--	--	--
/rest/accounting/liens/<id>	**Get Single Lien**	--	--	--

This topic contains these sections:

- Getting Liens
  - Get Single Lien
  - Get All Liens

Getting Liens

The HTTP GET method is used to retrieve **Lien** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/liens?api-version=3
GET http://localhost:8080/mws/rest/accounting/liens/<id>?api-version=3
```

Get All Liens

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/liens?api-version=3&proxy-user=<user>
[&active=true][&filter=<filter_options>[&filter-type=<filter_type>]][&query=<query_
conditions>][&fields=<fields_to_display>[&sort=<fields_to_sort>]|&show-all=
(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
active	No	Boolean	--	Lists only active or non-active liens.	`active=true`
filter	No	JSON	--	Query funds based on defined MAM filter.	`filter={"accoun-t":"chemistry"}`
filter-type	No	String	--	Query funds based on defined MAM filter type.	`filter-type=NonExclusive`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"alloc-ations.fund":2}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,instance,amount`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"instance":1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/liens?api-version=3&proxy-
user=amy&filter={"account":"chemistry"}&fields=instance,amount&active=true&pretty=true
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
        {
      "instance": "job.1",
      "amount": 57600
    },
        {
      "instance": "job.2",
      "amount": 40762
    }
  ]
}
```

Get Single Lien

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/liens/<id>?api-version=3&proxy-
user=<user>[&active=(true|false)][&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object	--
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
active	No	Boolean	--	Lists only active or non-active liens.	`active=true`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,name,amount`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/liens/1?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "id": 1,
  "instance": "job.1",
  "usageRecord": 1,
  "startTime": "2013-08-21 16:45:57 UTC",
  "endTime": "2013-08-21 17:45:57 UTC",
  "duration": 3600,
  "description": "",
  "amount": 57600,
  "allocations": [  {
    "id": 2,
    "fund": 2,
    "amount": 57600
  }]
}
```

Related Topics

- Fields: Liens
- Resources Introduction

Accounting Organizations

This section describes the services available through Moab Web Services for interacting with the **Organization** object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Organizations** reference contains the type and description of the default fields for the **Organization** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/organizations	Get All Organizations	--	--	--
/rest/accounting/organizations/<id>	Get Single Organization	--	--	--

This topic contains these sections:

- Getting Organizations
  - Get All Organizations
  - Get Single Organization

Getting Organizations

The HTTP GET method is used to retrieve **Organizations** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/organizations?api-version=3
GET http://localhost:8080/mws/rest/accounting/organizations/<id>?api-version=3
```

Get All Organizations

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/organizations?api-version=3&proxy-
user=<user>[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_
sort>]|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
query	No	JSON	--	Results are restricted to those having the specified field values. The query parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"deleted":-false}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id`

Parameter	Required	Type	Valid values	Description	Example
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"requestedId":-1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/organizations?api-version=3&proxy-
user=moab&fields=id,description&sort={"id":1}&pretty=true
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
        {
      "description": "Arts College",
      "id": "arts"
    },
        {
      "description": "Sciences College",
      "id": "sciences"
    }
  ]
}
```

Get Single Organization

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/organizations/<id>?api-version=3&proxy-
user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object.	--
fields	No	String	--	Comma-separated list of field names to display.	`fields=id`

Parameter	Required	Type	Valid values	Description	Example
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all-l=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/organizations/sciences?api-
version=3&proxy-user=moab&pretty=true
-------------------------------------

{
  "description": "Sciences College",
  "id": "sciences"
}
```

Related Topics

- Fields: Organizations
- Resources Introduction

Accounting Quotes

This section describes the services available through Moab Web Services for interacting with the **Quote** object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Quotes** reference contains the type and description of the default fields for the **Quotes** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/quotes	Get All Quotes	--	--	--
/rest/accounting/quotes/<id>	Get Single Quote	--	--	--

This topic contains these sections:

- Getting Quotes
  - Get All Quotes
  - Get Single Quote

Getting Quotes

The HTTP GET method is used to retrieve **Quote** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/quotes?api-version=3
GET http://localhost:8080/mws/rest/accounting/quotes/<id>?api-version=3
```

Get All Quotes

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/quotes?api-version=3&proxy-user=<user>
[&active=true][&filter=<filter_options>[&filter-type=<filter_type>]][&query=<query_
conditions>][&fields=<fields_to_display>[&sort=<fields_to_sort>]|&show-all=
(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform actions as defined MAM user.	`proxy-user=amy`
active	No	Boolean	`true` or `false`	Lists only active or non-active quotes.	`active=true`
filter	No	JSON	--	Query funds based on defined MAM filter.	`filter={"accoun-t":"chemistry"}`
filter-type	No	String	--	Query funds based on defined MAM filter type.	`filter-type=NonExclusive`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"instance":"-job.1"}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,instance,amount`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"instance":1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/quotes?api-version=3&proxy-
user=amy&filter={"account":"chemistry"}
&fields=usageRecord,amount&active=true&pretty=true
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": [  {
    "usageRecord": 1,
    "amount": 57600
  }]
}
```

Get Single Quote

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/quotes/<id>?api-version=3&proxy-
user=<user>[&active=(true|false)][&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object.	--
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
active	No	Boolean	`true` or `false`	Lists only active or non-active quotes.	`active=true`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,name,amount`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/quotes/1?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "id": 1,
  "amount": 57600,
  "pinned": true,
  "instance": "",
  "usageRecord": 1,
  "startTime": "2013-08-21 16:45:57 UTC",
  "endTime": "2013-08-21 17:57:57 UTC",
  "duration": 3600,
  "description": "",
  "chargeRates": [  {
    "name": "Processors",
    "value": "",
    "amount": "1/s"
  }]
}
```

Related Topics

- Fields: Quotes
- Resources Introduction

Accounting Transactions

This section describes the services available through Moab Web Services for interacting with the **Transaction** object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Transactions** reference contains the type and description of the default fields for the **Transaction** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/transactions	**Get All Transactions**	--	--	--
/rest/accounting/transactions/<id>	**Get Single Transaction**	--	--	--

This topic contains these sections:

- Getting Transactions
  - Get All Transactions
  - Get Single Transaction

Getting Transactions

The HTTP GET method is used to retrieve **Transaction** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/transactions?api-version=3
GET http://localhost:8080/mws/rest/accounting/transactions/<id>?api-version=3
```

Get All Transactions

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/transactions?api-version=3&proxy-
user=<user>[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_
sort>]|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"action":"Charge","account":"chemistry"}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id`

Parameter	Required	Type	Valid values	Description	Example
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **Accounting Transactions** parameter.	`sort={"id":1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/transactions?api-version=3&proxy-
user=moab&query={"instance":"job.1"}&fields=object,action,instance,amount&pretty=true
-----------------------------------

{
  "totalCount": 310,
  "resultCount": 3,
  "results":   [
        {
      "object": "UsageRecord",
      "action": "Reserve",
      "instance": "job.1",
      "amount": 57600
    },
        {
      "object": "UsageRecord",
      "action": "Charge",
      "instance": "job.1",
      "amount": 11520
    },
        {
      "object": "UsageRecord",
      "action": "Refund",
      "instance": "job.1",
      "amount": 11520
    }
  ]
}
```

Get Single Transaction

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/transactions/<id>?api-version=3&proxy-
user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object.	--
fields	No	String	--	Comma-separated list of field names to display.	`fields=id`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all-l=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/transactions/1?api-version=3&proxy-
user=moab&pretty=true
-----------------------------------

{
  "id": 1,
  "object": "Organization",
  "action": "Create",
  "actor": "scottmo",
  "key": "sciences",
  "child": "",
  "count": 1,
  "instance": "",
  "amount": "",
  "delta": "",
  "user": "",
  "account": "",
  "machine": "",
  "fund": "",
  "allocation": "",
  "usageRecord": "",
  "duration": "",
  "description": ""
}
```

Related Topics

- Fields: Transactions
- Resources Introduction

Accounting Usage Records

This section describes the services available through Moab Web Services for interacting with the **Usage Record** object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Usage Records** reference section contains the type and description of all fields in the **Usage Record** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/accounting/usage-records	**Get All Usage Records**	--	--	--
/rest/accounting/usage-records/<id>	**Get Single Usage Record**	--	--	--

Resource	GET	PUT	POST	DELETE
/rest/accounting/usage-records/quote	--	--	Obtain a Quote For Resource Usage	--

This topic contains these sections:

- Getting Usage Records
  - Get All Usage Records
  - Get Single Usage Record
  - Obtain a Quote For Resource Usage

Getting Usage Records

The HTTP GET method is used to retrieve **Usage Record** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/usage-records?api-version=3
GET http://localhost:8080/mws/rest/accounting/usage-records/<id>?api-version=3
POST http://localhost:8080/mws/rest/accounting/usage-records/quote?api-version=3
```

Get All Usage Records

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/usage-records?api-version=3&proxy-
user=<user>[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_
sort>]|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"account":"query"}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,instance,charge,user,account`

Parameter	Required	Type	Valid values	Description	Example
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"user":1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/usage-records?api-version=3&proxy-
user=amy&fields=id,instance,charge,user,account&pretty=true
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
        {
      "id": 1,
      "instance": "job.1",
      "charge": 31,
      "user": "amy",
      "account": "chemistry"
    },
        {
      "id": 2,
      "instance": "job.2",
      "charge": 30,
      "user": "amy",
      "account": "biology"
    }
}
```

Get Single Usage Record

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/usage-records/<id>?api-version=3&proxy-user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object.	`code`
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
fields	No	String	--	Comma-separated list of field names to display.	`fields=id,instance,charge,user,account`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/usage-records/1?api-version=3&proxy-
user=amy&pretty=true
-----------------------------------

{
  "id": 1,
  "type": "Job",
  "instance": "job.1",
  "charge": 31,
  "stage": "Charge",
  "user": "amy",
  "group": "faculty",
  "account": "chemistry",
  "organization": "sciences",
  "qualityOfService": "",
  "machine": "colony",
  "nodes": "",
  "processors": 16,
  "memory": "",
  "disk": "",
  "network": "",
  "duration": 720,
  "startTime": "",
  "endTime": "",
  "description": ""
}
```

Obtain a Quote For Resource Usage

URLs and parameters

```
POST http://localhost:8080/mws/rest/accounting/usage-records/quote?api-
version=3&object-type=<object>&proxy-user=<user>&charge-duration=<seconds>
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`
charge-duration	Yes	Integer	--	The quote duration of the job in seconds.	`charge-duration=6400`

Parameter	Required	Type	Valid values	Description	Example
object-type	Yes	String	--	The object to quote. It can be job or service.	`object-type=job`
itemize	No	Boolean	`true` or `false`	Returns the composite charge information in the response data.	`itemize=true`
rate	No	JSONArray	--	Uses the specified charge rates in the quote. The specified rates override the standard and quote rates. If the guarantee field is set to true, these charge rates will be saved and used when this quote is referenced in a charge action.	`rate=[{"type":"VBR","name":"Memory","rate":1},{"type":"VBR","name":"Processors","rate":1}]`

Parameter	Required	Type	Valid values	Description	Example
guarantee	No	Boolean	`true` or `false`	Guarantees the quote and returns a quote id to secure the current charge rates. This results in the creation of a quote record and a permanent usage record. This parameter is mutually exclusive with the cost-only parameter.	`guarantee=true`

Parameter	Required	Type	Valid values	Description	Example
grace-duration	No	Integer	--	The guaranteed quote grace period in seconds. If the quote duration is specified but not the quote end time, the quote endtime will be calculated as the quote start time plus the quote duration plus the grace duration.	`grace-duration=6400`
cost-only	No	Integer	--	Returns the cost, ignoring all balance and validity checks. This parameter is mutually exclusive with the guarantee parameter.	`cost-only=true`

Parameter	Required	Type	Valid values	Description	Example
description	No	String	--	The guaranteed quote description.	`description="ABC Coupon Rate"`
start-time	No	Date	--	The guaranteed quote start time in the format `yyyy-MM-dd HH:mm:ss z`, `-Infinity`, `Infinity`, or `Now`.	`start-time="2012-04-09 13:49:40 UTC"`
end-time	No	Date	--	The guaranteed quote end time in the format `yyyy-MM-dd HH:mm:ss z`, `-Infinity`, `Infinity`, or `Now`.	`end-time="2012-04-09 14:49:40 UTC"`

See Global URL Parameters for available URL parameters.

Request body

The request body below shows all of the fields in a job that could affect the quote.

```
POST http://localhost:8080/mws/rest/accounting/usage-records/quote?api-
version=3&object-type=job&charge-duration=300
-------------------------------------

{
  "id": "Moab.1",
  "user": "amy",
  "group": "group",
  "rmName": "machine1",
  "templateList": [
  "genericVm"
  ],
  "account": "biology",
  "qosRequested": "QOS1",
  "variables": {
    "imageName": "centos6.6-stateless",
    "topLevelServiceId": "myService.1",
    "serviceId": "vmService.1",
    "vmid": "VmService.1",
    "pmid": "VmService.1"
  },
  "requirements": [
  {
      "requiredProcessorsPerTask": 2,
      "genericResources": {
         "gold": 100,
         "os": 500
      },
      "requiredNodeCountMinimum": 1,
      "requiredMemoryPerTask": 1024,
      "requiredClass": "batch"
  }
  ]
}
```

The request body below shows all of the fields in a service that affect the quote in a default MAM installation.

```
POST http://localhost:8080/mws/rest/accounting/usage-records/quote?api-
version=3&object-type=service&charge-duration=300
-----------------------------------

{
     "name":"service.1",
     "user": "amy",
     "account": "chemistry"
     "attributes":{
        "moab":{
           "job":{
              "resources":{
                 "procs":1,
                 "mem":2048,
                 "OS":500,
                 "gold":100
              },
              "variables":{
                 "Var1": 1524
              },
              "image":"centos6.6-stateless",
              "template":"genericVM",
           }
        }
     }
  }
```

Sample response

- If the quote is not guaranteed:

```
JSON response
-----------------------------------

{
    "instance": "Moab.1",
    "amount": 600
}
```

- If the quote is guaranteed:

```
JSON response
-----------------------------------

{
    "id": 1,
    "usageRecord": 2,
    "instance": "Moab.1",
    "amount": 600
}
```

- If the quote is guaranteed and itemized:

```
JSON response
-------------------------------------

{
    "details":   [
        {
      "name": "Processors",
      "value": "2",
      "duration": 300,
      "rate": 1,
      "scalingFactor": 1,
      "amount": 600,
      "details": "2 [Processors] * 1 [ChargeRate{VBR}{Processors}] * 300
[Duration]"
    },
        {
      "name": "Memory",
      "value": "1024",
      "duration": 300,
      "rate": 1,
      "scalingFactor": 1,
      "amount": 307200,
      "details": "1024 [Memory] * 1 [ChargeRate{VBR}{Memory}] * 300 [Duration]"
    }
  ],
  "id": 20,
  "instance": "Moab.1",
  "usageRecord": 20,
  "amount": 307800
}
```

- If the quote is on a service:

```
JSON response
-------------------------------------

{
  "services":   [
        {
      "details":       [
                {
          "name": "Processors",
          "value": "22",
          "duration": 30,
          "rate": 1,
          "scalingFactor": 1,
          "amount": 660,
          "details": "22 [Processors] * 1 [ChargeRate{VBR}{Processors}] * 30
[Duration]"
        },
                {
          "name": "Memory",
          "value": "32343242",
          "duration": 30,
          "rate": 1,
          "scalingFactor": 1,
          "amount": 970297260,
          "details": "32343242 [Memory] * 1 [ChargeRate{VBR}{Memory}] * 30
[Duration]"
        }
      ],
      "id": 120,
      "instance": "myVmWorkflow",
      "usageRecord": 157,
      "amount": 970297920
    },
        {
      "details": [      {
        "name": "Storage",
        "value": "2500",
        "duration": 30,
        "rate": 1.157E-7,
        "scalingFactor": 1,
        "amount": 0,
        "details": "2500 [Storage] * 1.157e-07 [ChargeRate{VBR}{Storage}] * 30
[Duration]"
      }],
      "id": 122,
      "instance": "myExtraStorageWorkflow",
      "usageRecord": 159,
      "amount": 0
    },
        {
      "details":       [
                {
          "name": "Processors",
          "value": "0",
          "duration": 30,
          "rate": 1,
          "scalingFactor": 1,
          "amount": 0,
          "details": "0 [Processors] * 1 [ChargeRate{VBR}{Processors}] * 30
[Duration]"
        },
                {
```

```
            "name": "Memory",
            "value": "0",
            "duration": 30,
            "rate": 1,
            "scalingFactor": 1,
            "amount": 0,
            "details": "0 [Memory] * 1 [ChargeRate{VBR}{Memory}] * 30 [Duration]"
          }
        ],
        "id": 123,
        "instance": "myPmWorkflow",
        "usageRecord": 160,
        "amount": 0
      }
    ],
    "amount": 970297920
}
```


Restrictions

The `details` field is only available with MAM version 7.1.0 or later.

Related Topics

- Fields: Usage Records
- Resources Introduction

Accounting Users

This section describes the services available through Moab Web Services for interacting with the User object in Moab Accounting Manager. It contains the URLs, request bodies, and responses delivered to and from MWS as an intermediary for MAM.

The **Fields: Users** reference contains the type and description of all fields in the **User** object.

Supported methods

Resource	GET	PUT	POST	DELETE
rest/accounting/users	Get All Users	--	--	--
rest/accounting/users/<id>	Get Single User	--	--	--

This topic contains these sections:

- Getting Users
  - Get All Users
  - Get Single User

Getting Users

The HTTP GET method is used to retrieve **User** information.

Quick reference

```
GET http://localhost:8080/mws/rest/accounting/users?api-version=3
GET http://localhost:8080/mws/rest/accounting/users/<id>?api-version=3
```

Get All Users

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/users?api-version=3&proxy-user=<user>
[&query=<query_conditions>][&fields=<fields_to_display>[&sort=<fields_to_sort>]|&show-
all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
proxy-user	Yes	String	--	Perform action as defined MAM user.	`proxy-user=amy`

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Results are restricted to those having the specified field values. The **query** parameter does not support the full Mongo query syntax. Only querying for a simple, non-nested JSON object is allowed.	`query={"active":true}`
fields	No	String	--	Comma-separated list of field names to display.	`fields=name,defaultAccount`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending. Should be used in conjunction with the **fields** parameter.	`sort={"defaultAccount":1}`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/users?api-version=3&proxy-
user=moab&query={"active":true}&pretty=true
-----------------------------------

{
  "totalCount": 6,
  "resultCount": 4,
  "results":   [
        {
      "active": true,
      "commonName": "",
      "phoneNumber": "",
      "emailAddress": "",
      "defaultAccount": "",
      "description": "Accounting Admin",
      "id": "scottmo"
    },
        {
      "active": true,
      "commonName": "Amy Miller",
      "phoneNumber": "(801) 555-1437",
      "emailAddress": "amy@hpc.com",
      "defaultAccount": "chemistry",
      "description": "",
      "id": "amy"
    },
        {
      "active": true,
      "commonName": "Robert Taylor",
      "phoneNumber": "(801) 555-1474",
      "emailAddress": "bob@hpc.com",
      "defaultAccount": "biology",
      "description": "",
      "id": "bob"
    },
        {
      "active": true,
      "commonName": "David Jones",
      "phoneNumber": "(801) 555-1436",
      "emailAddress": "dave@hpc.com",
      "defaultAccount": "film",
      "description": "",
      "id": "dave"
    }
  ]
}
```

Get Single User

URLs and parameters

```
GET http://localhost:8080/mws/rest/accounting/users/<id>?api-version=3&proxy-
user=<user>[&fields=<fields_to_display>|&show-all=(true|false)]
```

Parameter	Required	Type	Valid values	Description	Example
id	Yes	String	--	The unique identifier of the object	--
fields	No	String	--	Comma-separated list of field names to display.	`fields=name,defaultAccount`
show-all	No	Boolean	`true` or `false`	`true` shows all fields including metadata and hidden fields. Default is `false`.	`show-all=true`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/accounting/users/amy?api-version=3&proxy-
user=moab&pretty=true
-----------------------------------

{
  "active": true,
  "commonName": "Amy Miller",
  "phoneNumber": "(801) 555-1437",
  "emailAddress": "amy@hpc.com",
  "defaultAccount": "chemistry",
  "description": "",
  "id": "amy"
}
```

Related Topics

- Fields: Users
- Resources Introduction

Credentials

This section describes behavior of the **Credential** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The Credential API is new with *API version 2*. The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3`.

For more information, see Requesting Specific API Versions on page 42.

The **Fields: Credentials** reference contains the type and description of all fields in the **Credential** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/credentials/accounts	Get All Account Credentials Get Single Account Credential	Modify Account Credentials	--	--
/rest/credentials/classes	Get All Class Credentials Get Single Class Credential	Modify Class Credentials	--	--
/rest/credentials/groups	Get All Group Credentials Get Single Group Credential	Modify Group Credentials	--	--
/rest/credentials/qoses	Get All QoS Credentials Get Single QoS Credential	Modify QoS Credentials	--	--
/rest/credentials/users	Get All User Credentials Get Single User Credential	Modify User Credentials	--	--
/rest/credentials/belongs-to	Get Credentials to which the User Belongs	--	--	--

In this topic:

- Getting Credentials on page 125
  - Get All Account Credentials on page 125
  - Get Single Account Credential on page 126
  - Get All Class Credentials on page 127
  - Get Single Class Credential on page 128
  - Get All Group Credentials on page 129
  - Get Single Group Credential on page 130
  - Get All QoS Credentials on page 131
  - Get Single QoS Credential on page 132
  - Get All User Credentials on page 133
  - Get Single User Credential on page 134
  - Get Credentials to which the User Belongs on page 135
- Modifying Credentials on page 136
  - Modify Account Credentials on page 137
  - Modify Class Credentials on page 137
  - Modify Group Credentials on page 137
  - Modify QoS Credentials on page 138
  - Modify User Credentials on page 138

Getting Credentials

The HTTP GET method is used to retrieve **Resource Type** information.

Quick reference

```
GET http://localhost:8080/mws/rest/credentials/accounts[/<name>]?api-version=3
GET http://localhost:8080/mws/rest/credentials/classes[/<name>]?api-version=3
GET http://localhost:8080/mws/rest/credentials/groups[/<name>]?api-version=3
GET http://localhost:8080/mws/rest/credentials/qoses[/<name>]?api-version=3
GET http://localhost:8080/mws/rest/credentials/users[/<name>]?api-version=3
```

Get All Account Credentials

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/accounts?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/accounts?api-version=3
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results":   [
    {
      "name": "Administration",
      "account_access_list": ["Administration"],
      "default_account": "Administration",
      "qos_access_list":   [
        "qos1",
        "qos2"
      ],
      "default_qos": "qos1",
      "partition_access_list":   [
        "partition1",
        "SHARED"
      ],
      "default_partition": "partition1",
      "target_type": "CEILING",
      "target": 18.43,
      "priority": 53,
      "max_job_duration_in_seconds": 234,
      "max_idle_jobs": "42",
      "max_jobs": "523",
      "max_processors": "4",
      "max_processor_seconds": "525",
      "max_nodes": "75",
      "reservation": "system.1"
    }
   ]
}
```

Get Single Account Credential

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/accounts/<name>?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/accounts/Administration?api-version=3
-----------------------------------

{
  "name": "Administration",
  "account_access_list": ["Administration"],
  "default_account": "Administration",
  "qos_access_list":   [
    "qos1",
    "qos2"
  ],
  "default_qos": "qos1",
  "partition_access_list":   [
    "partition1",
    "SHARED"
  ],
  "default_partition": "partition1",
  "target_type": "CEILING",
  "target": 18.43,
  "priority": 53,
  "max_job_duration_in_seconds": 234,
  "max_idle_jobs": "42",
  "max_jobs": "523",
  "max_processors": "4",
  "max_processor_seconds": "525",
  "max_nodes": "75",
  "reservation": "system.1",
  "user_access_list": ["adaptive"]
}
```

Get All Class Credentials

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/classes?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/classes?api-version=3
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results":   [
    {
      "name": "highprio",
      "account_access_list": ["Administration"],
      "default_account": "Administration",
      "qos_access_list":   [
        "qos1",
        "qos2"
      ],
      "default_qos": "qos1",
      "partition_access_list":   [
        "partition1",
        "SHARED"
      ],
      "default_partition": "partition1",
      "target_type": "CEILING",
      "target": 18.43,
      "priority": 53,
      "max_job_duration_in_seconds": 234,
      "max_idle_jobs": "42",
      "max_jobs": "523",
      "max_processors": "4",
      "max_processor_seconds": "525",
      "max_nodes": "75",
      "reservation": "system.1"
    }
  ]
}
```

Get Single Class Credential

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/classes/<name>?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/classes/highprio?api-version=3
-------------------------------------

{
  "name": "highprio",
  "account_access_list": ["Administration"],
  "default_account": "Administration",
  "qos_access_list":   [
    "qos1",
    "qos2"
  ],
  "default_qos": "qos1",
  "partition_access_list":   [
    "partition1",
    "SHARED"
  ],
  "default_partition": "partition1",
  "target_type": "CEILING",
  "target": 18.43,
  "priority": 53,
  "max_job_duration_in_seconds": 234,
  "max_idle_jobs": "42",
  "max_jobs": "523",
  "max_processors": "4",
  "max_processor_seconds": "525",
  "max_nodes": "75",
  "reservation": "system.1",
  "user_access_list": ["adaptive"]
}
```

Get All Group Credentials

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/groups/<name>?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/groups?api-version=3
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results":   [
    {
  "name": "students",
  "account_access_list": ["Administration"],
  "default_account": "Administration",
  "qos_access_list":   [
    "qos1",
    "qos2"
  ],
  "default_qos": "qos1",
  "partition_access_list":   [
    "partition1",
    "SHARED"
  ],
  "default_partition": "partition1",
  "target_type": "CEILING",
  "target": 18.43,
  "priority": 53,
  "max_job_duration_in_seconds": 234,
  "max_idle_jobs": "42",
  "max_jobs": "523",
  "max_processors": "4",
  "max_processor_seconds": "525",
  "max_nodes": "75",
  "reservation": "system.1"
}
```

Get Single Group Credential

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/groups/<name>?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/groups/students?api-version=3
-------------------------------------

{
  "name": "students",
  "account_access_list": ["Administration"],
  "default_account": "Administration",
  "qos_access_list":   [
    "qos1",
    "qos2"
  ],
  "default_qos": "qos1",
  "partition_access_list":   [
    "partition1",
    "SHARED"
  ],
  "default_partition": "partition1",
  "target_type": "CEILING",
  "target": 18.43,
  "priority": 53,
  "max_job_duration_in_seconds": 234,
  "max_idle_jobs": "42",
  "max_jobs": "523",
  "max_processors": "4",
  "max_processor_seconds": "525",
  "max_nodes": "75",
  "reservation": "system.1",
  "user_access_list": ["adaptive"]
}
```

Get All QoS Credentials

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/qoses?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/qoses?api-version=3
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results":   [
    {
      "name": "special",
      "account_access_list": ["Administration"],
      "default_account": "Administration",
      "qos_access_list":   [
        "qos1",
        "qos2"
      ],
      "default_qos": "qos1",
      "partition_access_list":   [
        "partition1",
        "SHARED"
      ],
      "default_partition": "partition1",
      "target_type": "CEILING",
      "target": 18.43,
      "priority": 53,
      "max_job_duration_in_seconds": 234,
      "max_idle_jobs": "42",
      "max_jobs": "523",
      "max_processors": "4",
      "max_processor_seconds": "525",
      "max_nodes": "75",
      "reservation": "system.1",
      "flags": [
         "DEADLINE",
         "RESERVEALWAYS",
         "DEDICATED"
      ]
      "queue_time_weight": 30,
      "expansion_factor_weight": 40,
      "quality_of_service_priority": 20
    }
   ]
}
```

Get Single QoS Credential

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/qoses/<name>?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/qoses/special?api-version=3
-------------------------------------

{
  "name": "special",
  "account_access_list": ["Administration"],
  "default_account": "Administration",
  "qos_access_list":   [
    "qos1",
    "qos2"
  ],
  "default_qos": "qos1",
  "partition_access_list":   [
    "partition1",
    "SHARED"
  ],
  "default_partition": "partition1",
  "target_type": "CEILING",
  "target": 18.43,
  "priority": 53,
  "max_job_duration_in_seconds": 234,
  "max_idle_jobs": "42",
  "max_jobs": "523",
  "max_processors": "4",
  "max_processor_seconds": "525",
  "max_nodes": "75",
  "reservation": "system.1",
  "user_access_list": ["adaptive"]
  "flags": [
    "DEADLINE",
    "RESERVEALWAYS",
    "DEDICATED"
   ]
   "queue_time_weight": 30,
   "expansion_factor_weight": 40,
   "quality_of_service_priority": 20
}
```

Get All User Credentials

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/users?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/users?api-version=3
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results":   [
    {
      "name": "root",
      "account_access_list": ["Administration"],
      "default_account": "Administration",
      "qos_access_list":   [
        "qos1",
        "qos2"
      ],
      "default_qos": "qos1",
      "partition_access_list":   [
        "partition1",
        "SHARED"
      ],
      "default_partition": "partition1",
      "target_type": "CEILING",
      "target": 18.43,
      "priority": 53,
      "max_job_duration_in_seconds": 234,
      "max_idle_jobs": "42",
      "max_jobs": "523",
      "max_processors": "4",
      "max_processor_seconds": "525",
      "max_nodes": "75",
      "email": "root@root.com"
    }
   ]
}
```

Get Single User Credential

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/users/<name>?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/credentials/users/root?api-version=3
-------------------------------------

{
  "name": "root",
  "account_access_list": ["Administration"],
  "default_account": "Administration",
  "qos_access_list":   [
    "qos1",
    "qos2"
  ],
  "default_qos": "qos1",
  "partition_access_list":   [
    "partition1",
    "SHARED"
  ],
  "default_partition": "partition1",
  "target_type": "CEILING",
  "target": 18.43,
  "priority": 53,
  "max_job_duration_in_seconds": 234,
  "max_idle_jobs": "42",
  "max_jobs": "523",
  "max_processors": "4",
  "max_processor_seconds": "525",
  "max_nodes": "75",
  "email": "root@root.com"
}
```

Get Credentials to which the User Belongs

Returns the groups, accounts, classes, and qualities of service to which the current user has access.

URLs and parameters

```
GET http://localhost:8080/mws/rest/credentials/belongs-to?api-version=3
```

See Global URL Parameters on page 39 for available URL parameters.

Sample response

```
{
  "account_access_list":   [
    "Test",
    "Research",
    "Engineering"
  ],
  "class_access_list":   [
    "batch3",
    "batch2",
    "batch"
  ],
  "group_access_list":   [
    "hgranger"
  ],
  "qos_access_list":   [
    "high",
    "medium",
    "low"
  ]
}
```

Modifying Credentials

The HTTP PUT method is used to modify **credentials**.

Quick reference

```
PUT http://localhost:8080/mws/rest/credentials/accounts/<name>?api-version=3[&change-
mode=<add|remove|set>]
PUT http://localhost:8080/mws/rest/credentials/classes/<name>?api-version=3[&change-
mode=<add|remove|set>]
PUT http://localhost:8080/mws/rest/credentials/groups/<name>?api-version=3[&change-
mode=<add|remove|set>]
PUT http://localhost:8080/mws/rest/credentials/qoses/<name>?api-version=3[&change-
mode=<add|remove|set>]
PUT http://localhost:8080/mws/rest/credentials/users/<name>?api-version=3[&change-
mode=<add|remove|set>]
```

URL parameters

URL parameters for modifying a credential.

Credentials parameter	Required	Type	Valid values	Description
change-mode	No	String	set (default) add remove	If **set**, replace existing list with the given one. If **add**, add the given field(s) to the existing list. If **remove**, remove the given field(s) from the existing list.

Moab Workload Manager will automatically add SHARED and the value of default_partition to the partition_access_list.

Modify Account Credentials

URLs and parameters

```
PUT http://localhost:8080/mws/rest/credentials/accounts/<name>?api-version=3[&change-
mode=<add|remove|set>]
```

See Global URL Parameters on page 39 for available URL parameters.

Sample body

```
PUT http://localhost:8080/mws/rest/credentials/accounts/biology?api-version=3&change-
mode=add
-----------------------------------

{
  "qos_access_list":   [
    "qos3",
    "qos4"
  ],
  "max_job_duration_in_seconds": 234
}
```

Modify Class Credentials

URLs and parameters

```
PUT http://localhost:8080/mws/rest/credentials/classes/<name>?api-version=3[&change-
mode=<add|remove|set>]
```

See Global URL Parameters on page 39 for available URL parameters.

Sample body

```
PUT http://localhost:8080/mws/rest/credentials/classes/highprio?api-version=3
-----------------------------------

{
  "max_idle_jobs": "50",
  "max_jobs": "300"
}
```

Modify Group Credentials

URLs and parameters

```
PUT http://localhost:8080/mws/rest/credentials/groups/<name>?api-version=3[&change-
mode=<add|remove|set>]
```

See Global URL Parameters on page 39 for available URL parameters.

Sample body

```
PUT http://localhost:8080/mws/rest/credentials/groups/students?api-version=3&change-
mode=set
-----------------------------------

{
  "reservation": "system.2",
  "user_access_list": ["tom"]
}
```

Modify QoS Credentials

URLs and parameters

```
PUT http://localhost:8080/mws/rest/credentials/qoses/<name>?api-version=3[&change-
mode=<add|remove|set>]
```

See Global URL Parameters on page 39 for available URL parameters.

Sample body

```
PUT http://localhost:8080/mws/rest/credentials/qoses/special?api-version=3
-----------------------------------

{
  "max_processors": "5",
  "max_processor_seconds": "500"
}
```

Modify User Credentials

URLs and parameters

```
PUT http://localhost:8080/mws/rest/credentials/users/<name>?api-version=3[&change-
mode=<add|remove|set>]
```

See Global URL Parameters on page 39 for available URL parameters.

Sample body

```
PUT http://localhost:8080/mws/rest/credentials/users/tom?api-version=3
-----------------------------------

{
  "email": "tom@root.com"
}
```

Related Topics

- Fields: Credentials
- Resources Introduction

Diagnostics

This section describes additional REST calls that are available for performing diagnostics on Moab Web Services.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/diag/about	Get Version Information	--	--	--
/rest/diag/auth	Diagnose Authentication	--	--	--
/rest/diag/health/summary	Get Health Summary	--	--	--
/rest/diag/health/detail	Get Health Detail	--	--	--

This topic contains these sections:

- Get Version Information
- Diagnose Authentication
- Connection Health Information
  - Get Health Summary
  - Get Health Detail
- Diagnostics

Get Version Information

The HTTP GET method is used to retrieve version and build information.

Quick reference

```
GET http://localhost:8080/mws/rest/diag/about?api-version=3
```

URLs and parameters

```
GET http://localhost:8080/mws/rest/diag/about?api-version=3
```

Sample response

The response contains the application suite, version, build date, and revision.

```
{
  "suite": "CLOUD",
  "version": "7.2.2",
  "buildDate": "2013.03.15_13.12.45",
  "revision": "302238e24e327f4aa45ab4c91834216a7fc19d63"
}
```

Diagnose Authentication

The HTTP GET method is used to test for proper authentication. This resource is designed to be used as a simple validation of credentials and gives no output besides the response code.

Quick reference

```
GET http://localhost:8080/mws/rest/diag/auth?api-version=3
```

URLs and parameters

```
GET http://localhost:8080/mws/rest/diag/auth?api-version=3
```

Sample response

A successful result is indicated by the `200` response code while a failure is indicated by a `401` response code.

```
{}
```

Connection Health Information

The HTTP GET method is used to retrieve health or status information for connections to external systems or software. There are two available resources for health, one that returns simple summary information and another that returns detailed information.

Quick reference

```
GET http://localhost:8080/mws/rest/diag/health/summary?api-version=3
GET http://localhost:8080/mws/rest/diag/health/detail?api-version=3
```

Get Health Summary

URLs and parameters

```
GET http://localhost:8080/mws/rest/diag/health/summary?api-version=3
```

If the MongoDB connection is down, authenticated resources are not available. While this resource does not possess much detail beyond that of simple connection information, it is still useful as it does not require authentication and therefore can be used to determine connection problems with MongoDB.

Sample response

The response contains the connection health for Moab Workload Manager (MWM), Moab Accounting Manager (MAM), MongoDB, LDAP, ZeroMQ, PAM, and the Insight database. A `true` response value indicates that the connection is healthy and available, and a `false` response indicates that the connection is currently down. Likewise, the `mongoConnected` property for Moab signifies the state of the Moab to MongoDB connection. The possible values of this state are `UP`, `DOWN`, `NOT_CONFIGURED` (when the MongoDB server is not configured in Moab), `NOT_SUPPORTED` (when Moab is not compiled with MongoDB support), and `UNKNOWN` (when MWS cannot communicate with Moab).

```
{
  "mam": {"connected": true},
  "mongo": {"connected": true},
  "mwm":   {
    "connected": true,
    "mongoConnected": "UP",
    "zmqConnected": true
  },
  "ldap": {"connected": true},
  "pam": {"connected": true},
  "zmq": {"connected": true},
  "insight": {"connected": true}
}
```

Get Health Detail

URLs and parameters

```
GET http://localhost:8080/mws/rest/diag/health/detail?api-version=3
```

If the MongoDB connection is down, authenticated resources such as this are not available. In this case, using the **Get Health Summary** instead may be required.

Sample response

The response contains the connection health and information for Moab Workload Manager (MWM), Moab Accounting Manager (MAM), MongoDB, LDAP, ZeroMQ, PAM, and the Insight database. A `"connected": true` response value indicates that the connection is healthy and available, and a `false` response indicates that the connection is currently down. Likewise, the `mongoConnected` property for Moab signifies the state of the Moab to MongoDB connection. The possible values of this state are `UP`, `DOWN`, `NOT_CONFIGURED`

(when the MongoDB server is not configured in Moab), `NOT_SUPPORTED` (when Moab is not compiled with MongoDB support), and `UNKNOWN` (when MWS cannot communicate with Moab). A message is also present for all down connections except Moab to MongoDB giving a reason for the error state.

```
{
  "mam":   {
    "connected": true,
    "adminUser": "root",
    "host": "mamhost",
    "port": 7741,
    "version": "7.5",
    "message": null
  },
  "mongo":   {
    "connected": true,
    "host": "127.0.0.1",
    "port": 27017,
    "replicaSet": null,
    "databaseName": "mws",
    "username": {},
    "version": "2.4.8",
    "message": null
  },
  "mwm":   {
    "connected": true,
    "adminUser": "root",
    "host": "localhost",
    "port": 42559,
    "version": "7.5",
    "licensedFeatures":     [
      "green",
      "provision",
      "vm"
    ],
    "state": "RUNNING",
    "mongo":     {
      "connected": "UP",
      "credentialsSet": false,
      "host": "myhost",
      "port": 27017
    },
    "zmq":     {
      "connected": true,
      "encryptionStatus": "OFF",
      "port": 5563
    },
    "message": null
  },
  "ldap":   {
    "connected": true,
    "message": null,
    "server": "openldapnis.ac",
    "port": 389,
    "baseDNs": ["dc=testldap,dc=ac"],
    "bindUser": "cn=admin,dc=testldap,dc=ac",
    "directoryType": "OpenLDAP Using InetOrgPerson Schema",
    "securityType": "NONE",
    "userObjectClass": "inetOrgPerson",
    "groupObjectClass": "groupOfNames",
    "ouObjectClass": "organizationalUnit",
    "userMembershipAttribute": null,
    "groupMembershipAttribute": "member",
    "userNameAttribute": "uid"
  },
  "pam":   {
    "connected": true,
```

```
    "authenticationModule": "system-auth",
    "message": "PAM is configured in MWS."
  },
  "zmq":   {
    "connected": true,
    "version": "3.2.3",
    "message": null,
    "mwmSubscriber":     {
      "connected": true,
      "address": "localhost",
      "port": 5563,
      "message": null
    },
    "mwsSubscriber":     {
      "connected": true,
      "address": "localhost",
      "port": 5564,
      "message": null
    },
    "publisher":     {
      "connected": true,
      "address": "*",
      "port": 5564,
      "message": null
    }
  },
  "insight":     {
    "connected": true,
    "jdbcUrl": "jdbc:postgresql://localhost/moab",
    "username": "moab-admin",
    "databaseType": "PostgreSQL",
    "version": "9.3.3",
    "message": null
  }
}
```


Related Topics

- Resources Introduction

Distinct

The **Distinct** resource enables clients to retrieve distinct (unique) values from another MWS resource. For example, a client can request the list of all `featuresReported` across all nodes like this:

```
GET http://localhost:8080/mws/rest/distinct/nodes/featuresReported/?api-version=3
```

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/distinct/<resource>/<field>	Get Distinct Values	--	--	--

This topic contains these sections:

- Get Distinct Values

Get Distinct Values

The HTTP GET method is used to retrieve **distinct** values from another MWS resource.

URLs and parameters

```
GET http://localhost:8080/mws/rest/distinct/<resource>/<field>?api-version=3
```

Parameter	Required	Type	Valid values	Example
resource	Yes	String	The MWS resource to query.	`nodes`
field	Yes	String	The field for which to return the distinct values.	`featuresReported`
query	No	JSON	Determines the subset of objects from which to retrieve the distinct values.	`query={"states.-powerState": "On"}`

The **Distinct** resource has no access control of its own. Rather, it depends on the access control of the MWS resource being queried.

For example, for a client to run a query like `/rest/distinct/nodes/featuresReported`, it must have `GET` rights on the `Nodes` resource. For more information, see Access Control.

Examples

Example 4-1: Get all `featuresReported` across all nodes

```
http://localhost:8080/mws/rest/distinct/nodes/featuresReported?api-version=3
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": ["vlan1"]
}
```

Example 4-2: Get all available operating system images across all nodes that are powered on

```
http://localhost:8080/mws/rest/distinct/nodes/operatingSystem.imagesAvailable?api-
version=3&query={"states.powerState": "On"}
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
    "linux",
    "windows"
  ]
}
```

Related Topics

- Resources Introduction

Events

This section describes the URLs, request bodies, and responses delivered to and from Moab Web Services for handling events.

The Event API is new with API version 3. The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3` in order to behave as documented.

For more information, see Requesting Specific API Versions.

The **Fields: Events** reference contains the type and description of all fields in the **Event** object. It also contains details regarding which fields are valid during POST actions.

Important changes

- The following fields have been renamed in API version 3:

Name in version 1 & 2	Name in version 3
eventTime	eventDate
sourceComponent	origin
errorMessage.message	message
relatedObjects	associatedObjects

- The following fields have been removed in API version 3.

  Warning: MWS will no longer report these fields, even if there are existing events in the database with these fields.

  - eventCategory
  - status
  - facility
  - initiatedBy
  - primaryObject (Primary objects are now reported in associatedObjects.)
  - errorMessage.originator
  - errorMessage.errorCode
  - details

- The following fields are new in API version 3 (see Fields: Events):
  - arguments
  - code

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/events	Get All Events	--	Create Event	--
/rest/events/<id>	Get Single Event	--	--	--

This topic contains these sections:

- Getting Events
  - Get All Events
  - Get Single Event
- Creating Events
  - Create Event

Getting Events

The HTTP GET method is used to retrieve **Event** information. Queries for all objects and a single object are available.

Quick reference

```
GET http://localhost:8080/mws/rest/events?api-version=3[&query={"field":"value"}&sort=
{"field":<1|-1>}]
GET http://localhost:8080/mws/rest/events/<id>?api-version=3
```

Get All Events

URLs and parameters

```
GET http://localhost:8080/mws/rest/events?api-version=3[&query={"field":"value"}&sort=
{"field":<1|-1>}]
```

Parameter	Required	Type	Valid values	Example
query	No	JSON	Query for specific results. It is possible to query events by one or more fields based on MongoDB query syntax.	`query={"sever-ity":"ERROR"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"id":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/events?api-version=3
-------------------------------------

{
   "totalCount":2,
   "resultCount":2,
   "results":[
      {
         "arguments":[

         ],
         "associatedObjects":[
            {
               "type":"VM",
               "id":"vm1"
            }
         ],
         "tenant":
            {
               "id":"1234567890abcdef12345678",
               "name":"Research"
            },
         "code":234881023,
         "eventDate":"2013-06-10 17:13:31 UTC",
         "eventType":"VM Provision",
         "message":null,
         "origin":"CSA Plugin",
         "severity":"INFO",
         "id":"51b6093bc4aa708a5bebb6ae"
      },
      {
         "arguments":[
            "51b608ddc4aa708a5bebb684"
         ],
         "associatedObjects":[
            {
               "type":"Service",
               "id":"51b608ddc4aa708a5bebb684"
            }
         ],
         "tenant":
            {
               "id":"1234567890abcdef12345678",
               "name":"Research"
            },
         "code":33554944,
         "eventDate":"2013-06-10 17:11:59 UTC",
         "eventType":"Service Create",
         "message":"The service '51b608ddc4aa708a5bebb684' was created",
         "origin":"MWS/ServiceEvents/CREATE_1ID",
         "severity":"INFO",
         "id":"51b608dfc4aa708a5bebb686"
      }
   ]
}
```

Querying events

It is possible to query events by one or more fields based on MongoDB query syntax. The following contains examples of simple and complex event queries

and event queries by date.

Simple queries:

- To see only events that are of type "Service Create":

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventType":"Service Create"}
```

- To see only events of type "Service Create" with the severity of "INFO":

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventType":"Service Create","severity":"INFO"}
```

- To see only events with a code of 33554946

```
http://localhost:8080/mws/rest/events?api-version=3&query={code:33554946}
```

More complex queries:

- You can query on embedded JSON objects within the event JSON. For example, to see events associated with service 51b608ddc4aa708a5bebb684:

```
http://localhost:8080/mws/rest/events?api-version=3&query=
{"associatedObjects.id":"51b608ddc4aa708a5bebb684"}
```

- To see only events that are NOT associated with service 51b608ddc4aa708a5bebb684:

```
http://localhost:8080/mws/rest/events?api-version=3&query=
{"associatedObjects.id":{"$ne":"51b608ddc4aa708a5bebb684"}}
```

- When the field values of the desired events are a finite set, you can use the $in operator. For example, to see events that have a severity of either WARN or ERROR:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"severity":{"$in":
["ERROR","WARN"]}}
```

Querying events by date

- To see events created before January 27, 2012 at 12:08 a.m. UTC:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventDate":
{"$lt":"2012-01-27 12:08:00 UTC"}}
```

- To see events created before or on January 27, 2012 at 12:08 a.m. UTC:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventDate":
{"$lte":"2012-01-27 12:08:00 UTC"}}
```

- To see all events created after January 27, 2012 at 12:04 a.m. UTC:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventDate":
{"$gt":"2012-01-27 12:04:00 UTC"}}
```

- To see all events created after or on January 27, 2012 at 12:04 a.m. UTC:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventDate":
{"$gte":"2012-01-27 12:04:00 UTC"}}
```

- To see events created between 12:04 a.m. and 12:08 a.m. UTC inclusive:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventDate":
{"$gte":"2012-01-27 12:04:00 UTC","$lte":"2012-01-27 12:08:00 UTC"}}
```

- To see events created between 12:04 a.m. and 12:08 a.m. UTC inclusive that have a severity of ERROR:

```
http://localhost:8080/mws/rest/events?api-version=3&query=
{"severity":"ERROR","eventDate":{"$gte":"2012-01-27 12:04:00 UTC","$lte":"2012-
01-27 12:08:00 UTC"}}
```

Sorting

See the sorting section of Global URL Parameters.

Limiting the number of results

- If you want to limit the number of results of events, you can use the `max` parameter. For example, to see only 10 "VM Provision" events:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventType":"VM
Provision"}&sort={"eventDate":1}&max=10
```

- To see "VM Provision" events 51-60 when sorted by eventDate in descending order, you can combine `max` with `offset`, as follows:

```
http://localhost:8080/mws/rest/events?api-version=3&query={"eventType":"VM
Provision"}&sort={"eventDate":-1}&max=10&offset=50
```

Get Single Event

URLs and parameters

```
GET http://localhost:8080/mws/rest/events/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/events/51b608dfc4aa708a5bebb686?api-version=3
-----------------------------------

{
  "arguments": ["51b608ddc4aa708a5bebb684"],
  "associatedObjects": [  {
    "type": "Service",
    "id": "51b608ddc4aa708a5bebb684"
  }],
  "tenant":
    {
      "id":"1234567890abcdef12345678",
      "name":"Research"
    },
  "code": 33554944,
  "eventDate": "2013-06-10 17:11:59 UTC",
  "eventType": "Service Create",
  "message": "The service '51b608ddc4aa708a5bebb684' was created",
  "origin": "MWS/ServiceEvents/CREATE_1ID",
  "severity": "INFO",
  "id": "51b608dfc4aa708a5bebb686"
}
```

Creating Events

The HTTP POST method is used to create an **Event**.

Quick reference

```
POST http://localhost:8080/mws/rest/events?api-version=3
```

Create Event

URLs and parameters

```
POST http://localhost:8080/mws/rest/events?api-version=3
```

Request body

```
POST http://localhost:8080/mws/rest/events?api-version=3 Content-Type:application/json
-----------------------------------

{
      "arguments": ["vm1"],
      "associatedObjects": [      {
        "type": "VM",
        "id": "vm1"
      }],
      "code": 234881023,
      "eventDate": "2013-06-10 17:13:31 UTC",
      "eventType": "VM Provision",
      "message": "The virtual machine \"vm1\" was provisioned",
      "origin": "CSA Plugin",
      "severity": "INFO"
    }
```

> An event's `tenant` is automatically inherited from the `associatedObjects`.

Sample response

If the request was successful, the response will be an object with an **id** property containing the ID of the newly created events. On failure, the response is an error message.

```
JSON response
-------------------------------------

{"arguments":["vm1"],"associatedObjects":[{"_
id":"vm1","id":"vm1","type":"VM","version":0}],"code":234881023,"eventDate":"2013-06-
10 17:13:31 UTC","eventType":"VM
Provision","id":"51b62046c4aa708a5bebc018","message":"The virtual machine vm1 was
provisioned","origin":"CSA Plugin","severity":"INFO","version":0}
```

Below is an example of `events.log` output for a successful event request:

```
2013-06-10T11:13:31.000-06:00 severity="INFO" code="0x0dffffff" type="VM Provision"
origin="CSA Plugin" associatedObject.0.type="VM" associatedObject.0.id="vm1"
arguments=["vm1"] message="The virtual machine \"vm1\" was provisioned"
```

> Note that " (double quote) characters in the input have been replaced by **\"** characters in the output. (For other character restrictions, see Restrictions.)

Restrictions

Special characters—such as newline, carriage return, and " (double quote) characters—are encoded in the output of `events.log` to make `events.log` easy to parse with scripts and third party tools. For example, if the input XML contains:

```
<ErrorMessage>RM says, "Cannot provision vm21"</ErrorMessage>
```

Then the following will be output to `events.log`:

```
error.message="RM says, \"Cannot provision vm21\""
```

(Notice that **"** has been replaced with **\"**.)

This table contains the most common encodings. (For more information, see escape sequences for Java Strings.)

Character	Escape sequence
" (double quote)	\"

Character	Escape sequence
\ (backslash)	\\
newline	\n
carriage return	\r
tab	\t

Other restrictions include:

- **origin**, **eventType**, **associatedObject.id**, and **associatedObject.type** cannot contain single quotes (') or double quotes (").

Related Topics

- Resources Introduction
- Notifications
- Fields: Notifications
- Notification Conditions
- Fields: Notification Conditions
- Fields: Events
- System Events
- Creating Events and Notifications
- Plugin Event Service
- Handling Events
- Securing the Connection with the Message Queue

Images

This section describes behavior of the **Image** object in Moab Web Services. An image resource is used to track the different types of operating systems and hypervisors available in the data center. It also tracks which virtual machines are available on the hypervisors. This section describes the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Images** reference contains the type and description of all fields in the **Image** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/images	Get All Images	--	Create Single Image	--
/rest/images/<id>	Get Single Image	Modify Single Image	--	Delete Single Image
/rest/images/<name>	Get Single Image	Modify Single Image	--	Delete Single Image

This topic contains these sections:

- Getting Images
  - Get All Images
  - Get Single Image
- Creating Images
  - Create Single Image
- Modifying Images
  - Modify Single Image
- Deleting Images
  - Delete Single Image

Getting Images

The HTTP GET method is used to retrieve **Image** information. You can query all objects or a single object.

Quick reference

```
GET http://localhost:8080/mws/rest/images?api-version=3[&query={"field":"value"}&sort=
{"field":<1|-1>}]
GET http://localhost:8080/mws/rest/images/<id>?api-version=3
GET http://localhost:8080/mws/rest/images/<name>?api-version=3
```

Get All Images

URLs and parameters

```
GET http://localhost:8080/mws/rest/images?api-version=3[&query={"field":"value"}&sort=
{"field":<1|-1>}]
```

Parameter	Required	Type	Description	Example
query	No	JSON	Queries for specific results. It is possible to query images by one or more fields based on MongoDB query syntax.	`query={"type":"state-ful","osType":"linux"}`
sort	No	JSON	Sort the results. Use 1 for ascending and -1 for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/images?api-version=3&fields=id,name
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": [  {
    "id": "4fa197e68ca30fc605dd1cf0",
    "name": "centos6-stateful"
  }]
}
```

Sorting and querying

See the sorting and querying sections of Global URL Parameters.

Get Single Image

URLs and parameters

```
GET http://localhost:8080/mws/rest/images/<id>?api-version=3
GET http://localhost:8080/mws/rest/images/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Image.
name	Yes	String	--	The name of the Image.

You must specify either **id** or **name**, but you do not have to specify both.

See Global URL Parameters for available URL parameters.

Sample response

Virtual machine image example:

```
GET http://localhost:8080/mws/rest/images/centos6-compute-stateful?api-version=3
-----------------------------------

{
  "active":true,
  "extensions":{
    "xcat":{
      "os":"centos",
      "architecture":"x86_64",
      "profile":"compute"
    }
  },
  "features":[],
  "hypervisor":false,
  "hypervisorType": null,
  "id":"4fa197e68ca30fc605dd1cf0",
  "name":"centos6-compute-stateful",
  "osType":"linux",
  "supportsPhysicalMachine":false,
  "supportsVirtualMachine":true,
  "templateName":null,
  "type":"stateful",
  "version":0,
  "virtualizedImages":[]
}
```

Hypervisor image example:

```
GET http://localhost:8080/mws/rest/images/esxi-4.1-stateful?api-version=3
-----------------------------------

{
  "active":true,
  "extensions":{
    "xcat":{
      "hvGroupName":"hvGroup",
      "vmGroupName":"vmGroup",
      "os":"esxi-4.1",
      "architecture":"x86_64",
      "profile":"hv"
    }
  },
  "features":[],
  "hypervisor":true,
  "hypervisorType":"ESX",
  "id":"4fa197e68ca30fc605dd1cf0",
  "name":"centos6-compute-stateful",
  "osType":"linux",
  "supportsPhysicalMachine":true,
  "supportsVirtualMachine":false,
  "templateName":null,
  "type":"stateful",
  "version":0,
  "virtualizedImages":[]
}
```

The `version` field contains the current version of the database entry and does not reflect the version of the operating system. For more information, see Modify Single Image.

Creating Images

The HTTP POST method is used to submit **Images**.

Quick reference

```
POST http://localhost:8080/mws/rest/images?api-version=3
```

Create Single Image

URLs and parameters

```
POST http://localhost:8080/mws/rest/images?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

Three fields are required to submit an image: **name**, **hypervisor**, and **osType**. Each image must also support provisioning to either a physical machine or a virtual machine by using the **supportsPhysicalMachine** or **supportsVirtualMachine** fields.

> The **name** field must contain only letters, digits, periods, dashes, and underscores.

The array of virtualized images are themselves objects that contain image IDs or names. For more information on available fields and types, see Fields: Images.

The following is an example of the most basic image that can be created:

```
POST http://localhost:8080/mws/rest/images?api-version=3
------------------------------------

{
  "name": "centos6-stateful",
  "osType": "linux",
  "hypervisor": false,
  "supportsVirtualMachine":true
}
```

Note that this example does not provide any information for a provisioning manager (such as xCAT) to actually provision the machine. In order to provide this, you must add an entry to the **extensions** field that contains provisioning manager-specific information. Each key in the extensions field corresponds to the provisioning manager, and certain properties are required based on this key. For example, the xCAT extension key must be named `xcat` and must contain certain fields. These extension keys are documented in Fields: Images. See the following examples of creating images with xCAT-specific provisioning information below.

Sample response

If the request was successful, the response body is the new image that was created exactly as shown in **Get Single Image**. On failure, the response is an error message.

Samples

The **virtualizedImages** field only accepts input when the image is a hypervisor and expects an array of image IDs *or* names, as shown in the following example:

```
Example payload of hypervisor with 2 vms
-----------------------------------

{
  "hypervisor":true,
  "name":"esx5-stateful",
  "osType":"linux",
  "supportsPhysicalMachine":true,
  "type":"stateful",
  "hypervisorType":"ESX",
  "virtualizedImages":   [
    {"id": "4fa197e68ca30fc605dd1cf0"},
    {"name": "centos6-stateful"}
  ]
}
```

The following example shows how to create an image that utilizes a cloned template for a virtual machine. (Note that the **type** must be set to `linkedclone` in order to set the **templateName** field.)

```
VM Utilizing a Cloned Template
-----------------------------------

{
  "active": true,
  "hypervisor": false,
  "name": "centos6-compute-stateful",
  "osType": "linux",
  "type": "linkedclone",
  "supportsVirtualMachine":true,
  "templateName":"centos6-compute"
}
```

The following are samples of a virtual machine and a hypervisor image that can be provisioned with xCAT:

```
xCAT Virtual Machine Image
-----------------------------------

{
  "active": true,
  "features": [],
  "hypervisor": false,
  "name": "centos6-compute-stateful",
  "osType": "linux",
  "type": "stateful",
  "supportsVirtualMachine":true,
  "extensions": {
    "xcat": {
      "os": "centos",
      "architecture": "x86_64",
      "profile": "compute"
    }
  }
}
```

```
xCAT Hypervisor Image
-------------------------------------

{
  "active": true,
  "features": [],
  "hypervisor": true,
  "name": "esxi5-base-stateless",
  "osType": "linux",
  "virtualizedImages":   [
    {"name": "centos6-compute-stateless"}
  ],
  "type": "stateless",
  "hypervisorType": "ESX",
  "supportsPhysicalMachine":true,
  "extensions": {
    "xcat": {
      "os": "esxi5",
      "architecture": "x86_64",
      "profile": "base",
      "hvType": "esx",
      "hvGroupName": "esx5hv",
      "vmGroupName": "esx5vm"
    }
  }
}
```

Modifying Images

The HTTP PUT method is used to modify **Images**.

Quick reference

```
PUT http://localhost:8080/mws/rest/images/<id>?api-version=3
PUT http://localhost:8080/mws/rest/images/<name>?api-version=3
```

Modify Single Image

URLs and parameters

```
PUT http://localhost:8080/mws/rest/images/<id>?api-version=3
PUT http://localhost:8080/mws/rest/images/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Image.
name	Yes	String	--	The name of the Image.

You must specify either **id** or **name**, but you do not have to specify both. The **name** field must contain only letters, digits, periods, dashes, and underscores.

See Global URL Parameters for available URL parameters.

Example request

```
PUT http://locahost/mws/rest/images/centos6-stateful?api-version=3
-----------------------------------

{
  "name": "centos6-stateful",
  "type": "stateful",
  "hypervisor": false,
  "osType": "linux",
  "virtualizedImages": []
}
```

The **version** field contains the current version of the database entry and does not reflect the version of the operating system. This field cannot be updated directly. However, if **version** is included in the modify request, it will be used to verify that another client did not update the object in between the time the data was retrieved and the modify request was delivered.

Sample response

If the request was successful, the response body is the modified image as shown in **Get Single Image**. On failure, the response is an error message.

Deleting Images

The HTTP DELETE method is used to delete **Images**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/images/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/images/<name>?api-version=3
```

Delete Single Image

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/images/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/images/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Image.
name	Yes	String	--	The name of the Image.

Only one of **id** or **name** are required.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON Response
-------------------------------------

{}
```

Related Topics

- Fields: Images
- Resources Introduction

Insight Database

This section describes the behavior of the Insight database query API in Moab Web Services. It contains the URLs and responses delivered to and from MWS.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/insight/priv/<view>	**Get All Rows in a View**	--	--	--
/rest/insight/user/<view>	Get All Rows the Current User Can See	--	--	--

This topic contains these sections:

- Getting Rows in a Relational Database View
  - Get All Rows in a View
  - Get All Rows the Current User Can See

Getting Rows in a Relational Database View

Queries with JSON operators

The HTTP GET method is used to retrieve rows in a view or table of the Insight database in JSON format. An array of JSON objects is returned. Each JSON object corresponds to a row in the view/table. The field name and values in each JSON object correspond the column names and values in the view/table. For example if view **myview1** in the Insight database contained the following data:

Table 4-1: "myview1"

column1	column2	column3	user_name
alpha	delta	1	wsmith
bravo	echo	2	tjones
charlie	echo	3	wsmith

Then you would query this data as follows:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3
```

The resulting JSON would look like this:

```
[
  {
    "column1":"alpha",
    "column2":"delta",
    "column3":1,
    "user_name":"wsmith"
  },
  {
    "column1":"bravo",
    "column2":"echo",
    "column3":2,
    "user_name":"tjones"
  },
  {
    "column1":"charlie",
    "column2":"echo",
    "column3":3,
    "user_name":"wsmith"
  }
]
```

You can query using MongoDB comparision and logical operators. For example:

- To query for all rows where column2 = echo:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"column2":"echo"}
```

- To query for all rows where column2 = echo or column1 = bravo:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"$or":[{"column2":"echo"},{"column1":"bravo"}]}
```

- To query for all rows where column3 >= 2:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"column3":{"$gte":2}}
```

Supported Mongo comparison operators include `$gt`, `$gte`, `$in`, `$lt`, `$lte`, `$ne`, and `$nin`. Supported logical operators include `$or`, `$and`, `$not`, and `$nor`.

Queries with dates and times

You can also query using date/time. For example:

- If create_datetime is a column in the "myview1" view and its data type is TIMESTAMP, then the following query is possible:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"create_datetime":{"$gte":"2014-03-08 4:00:00 EST"}}
```

This queries "myview1" for all rows whose value for create_datetime is greater than or equal to March 8, 2014 at 4:00:00 EST.

- To query for all rows where create_date is found in a date range, use `$and`, as follows:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"$and":[{"some_datetime":{"$gte":"2014-03-08 4:00:00 EST"}},{"some_datetime":
{"$lte":"2014-03-08 7:00:00 EST"}}]}
```

Queries with wildcards

You can query for rows where a column value matches a specified pattern, instead of matching an exact string, by using the `$like` operator, which is similar to the SQL LIKE operator. `$like` allows the user of "%" as a wildcard that matches any substring. For example:

- To select all rows where column3 starts with the letter "e" and ends with any substring:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"column3":{"$like":"e%"}}
```

This matches rows where column3 is "echo," "excellent," "endeavor," etc.

- To query for all rows where column3 ends with the letter, "o":

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"column3":{"$like":"%o"}}
```

- To query for all rows where column3 contains the substring "ch":

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"column3":{"$like":"%ch%"}}
```

When performing `$like` queries, you may need to replace `%` with the URL-encoded value of **%25**; otherwise, the meta-character may be interpreted as `%`.

You can also use "_" to substitute for any single character. For example, to query for all rows that start with any character and end in "cho":

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query=
{"column3":{"$like":"_cho"}}
```

This matches rows where colum3 has the value "acho," "bcho," "ccho," "dcho," "echo," etc.

Privileged queries vs. user queries

The examples above show queries performed using the privileged query URL (`/rest/insight/priv`). The privileged query does not impose any additional conditions to what you specify in your query and max and offset parameters. By default only administrators have permission to use the privileged query URL. If you grant an unprivileged user access to the privileged query URL, the user could see data he should not be permitted to see.

If an unprivileged user needs to query the database view, he should only have permission to do so with the user query URL. This URL (`/rest/insight/user`) adds the implicit condition that the value for the user_name column must match that of the user that is currently logged in.

For example, the following two queries, the first a user query performed by user wsmith and the second a privileged query, are essentially the same:

```
GET http://localhost:8080/mws/rest/insight/user/myview1?api-version=3&query=
{"column2":"echo"}
```

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&query={$and":
[{"column2":"echo"},{"user_name":"wsmith"}]
```

Note that the user query URL can only access views that have a user_name column.

Field selection

To select which columns you want to see in the results, use the **fields** parameter. For example, to see only column1 and column2:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-
version=3&fields=column1,column2
```

Sorting

You can also sort the results by a one or more columns in ascending or descending order. Specifying additional sort columns provides an alternate value by which MWS will sort the rows if the values in the first column(s) match in multiple rows. There is no limit to the number of columns you can specify in your sort URL.

> All sort columns must be specified in the fields selection parameter.

To sort the results by a single column in ascending order:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&sort=
{"column1":1}
```

To sort the results by a single column in descending order:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&sort=
{"column1":-1}
```

To add a second sort column by which MWS will sort any rows with the same value in the first column:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&sort=
[{"column1":1},{"column2":1}]
```

To add a third sort column by which MWS will sort any rows with the same values in the first column and the second column:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&sort=
[{"column1":1},{"column2":1},{"column3":1}]
```

Using aggregate functions, group by, and having clauses

You can use SQL aggregate functions like count(), sum(), and avg(). Aggregate functions perform an operation on a group of rows that have the same value with respect to an SQL 'GROUP BY' clauses. Aggregate functions are specified in the fields URL parameter.

Usage Scenario:

1. Assume you have the following data:

user_name	column1	column2
alice	3.0	alpha
alice	11.0	bravo
alice	4.0	charlie
bob	5.0	delta
bob	1.0	echo
dave	10.0	foxtrot

2. You want to get the sum of the column1 value for all rows corresponding to a user. For example you would like to know that the sum of column1 values for alice is 3 + 11 + 4 = 18.

3. You would then use the sum() function and **group-by** clause as follows:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&fields=user_
name,sum(column1)&group-by=user_name
```

Via SQL

```
SELECT user_name,SUM(column1) FROM myview1 GROUP BY user_name
```

It will return the following data:

```
[
  {
    "user_name":"alice",
    "sum":"18",
  },
  {
    "user_name":"bob",
    "sum":"6",
  },
  {
    "user_name":"dave",
    "sum":"10",
  },
]
```

4. Now try using the **having** clause to query only for rows where the results of an aggregate function meet some constraint. For example, to query only for users where the sum of column1 is greater than or equal to 10. Run the following:

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-version=3&fields=user_
name,sum(column1)&group-by=user_name&having={"sum(column1)":{"$gte":10}}
```

Via SQL

```
SELECT user_name,SUM(column1) FROM myview1 GROUP BY user_name HAVING SUM(column1)
>= 10
```

Having clauses have an identical syntax to query clauses except that query clauses constrain individual columns and having clauses constrain the results of aggregate functions over a set of rows that have been grouped together.

Other commonly used SQL aggregate functions include average and count. For example, to find out the

- average column1 value for each user.

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-
version=3&fields=user_name,avg(column1)&group-by=user_name
```

- count of all rows associated with each user (use the count function with a parameter of 1).

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-
version=3&fields=user_name,count(1)&group-by=user_name
```

- count of all rows associated with each user that has less than 3 rows.

```
GET http://localhost:8080/mws/rest/insight/priv/myview1?api-
version=3&fields=user_name,count(1)&group-by=user_name&having={"count(1)":
{"$lt":3}}
```

Quick reference

```
GET http://localhost:8080/mws/rest/insight/priv/<view>?api-version=3
GET http://localhost:8080/mws/rest/insight/user/<view>?api-version=3
```

Get All Rows in a View

URLs and parameters

```
GET http://localhost:8080/mws/rest/insight/priv/<view>?api-version=3
GET http://localhost:8080/mws/rest/insight/user/<view>?api-version=3
```

Parameter	Required	Type	Description	Example
fields	No	Comma-separated string	Includes only specified fields in output.	`fields=name,type`
group-by	No	Comma-separated-string	Used in conjunction with aggregate functions.	group-by=name
having	No	JSON	Queries for specific rows using results from aggregate functions.	having= {"sum(col1)":100
max	No	Integer	The maximum number of items to return.	`max=100`
offset	No	Integer	The index of the first item to return.	`offset=200`
pretty	No	Boolean	`true` shows pretty printing of output. Default is `false`.	`pretty=true`
query	No	JSON	Queries for specific results. It is possible to query by one or more fields based on a subset of MongoDB query syntax. (See above.)	`query={"name":"node003"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

Get all rows

Queries performed with the privileged query URL (`/rest/insight/priv/<view>`) return all rows in a view that match that query.

Sample response

```
GET http://localhost:8080/mws/rest/insight/priv/node_management_view?api-
version=3&pretty=true&fields=name,state,user_name
-----------------------------------

{
  "totalCount": 5,
  "resultCount": 5,
  "results":   [
    {
    "job_name": "Moab.613",
    "job_state": "Running",
    "user_name": "wsmith"
    },
    {
    "job_name": "Moab.614",
    "job_state": "Running",
    "user_name": "tjones"
    },
    {
    "job_name": "Moab.615",
    "job_state": "Idle",
    "user_name": "wsmith"
    },
    {
    "job_name": "Moab.616",
    "job_state": "Running",
    "user_name": "tjones"
    },
    {
    "job_name": "Moab.617",
    "job_state": "Idle",
    "user_name": "tjones"
    },
   ]
}
```

Get All Rows the Current User Can See

Queries performed with the user query URL (`/rest/insight/user/<view>`) only return rows that the current user is authorized to see. The rows returned contain values in the user_name column that match that of the user who is currently logged in. For example, if tjones is logged in, then the user query URL will only return rows where user_name is equal to "tjones" and that match any other query submitted.

Note that you can only use the user query URL to query views with a user_name column.

Sample response

```
GET http://localhost:8080/mws/rest/insight/user/node_management_view?api-
version=3&pretty=true&fields=name,state,user_name&query={"job_state":"Running"}
{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
    {
      "job_name": "Moab.614",
      "job_state": "Running",
      "user_name": "tjones"
    },
    {
      "job_name": "Moab.616",
      "job_state": "Running",
      "user_name": "tjones"
    },
  ]
}
```

Because tjones is currently logged in, MWS only returns information about running jobs submitted by tjones even though the query does not explicitly filter by user_name.

Related Topics

- Reports
- Configuring Moab Web Services

Job Arrays

This section describes behavior of the **Job Array** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Job Arrays** reference section contains the type and description of all fields in the **Job Array** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/job-arrays	--	--	Submit Job Array	--

This topic contains these sections:

- Submitting Job Arrays
  - Submit Job Array

Submitting Job Arrays

The HTTP POST method is used to submit **Job Arrays**.

Quick reference

```
POST http://localhost:8080/mws/rest/job-arrays?api-version=3[&proxy-user=<username>]
```

While the `Job Array` resource only gives access to create job arrays, job arrays are retrieved using the operations in Getting Job Information.

Restrictions

All restrictions present for **Submitting Jobs** are present for job arrays. In addition, job arrays are *only* supported if the `ENABLEJOBARRAYS` parameter is set to `TRUE` in the `moab.cfg` file. For example:

```
ENABLEJOBARRAYS         TRUE
```

Submit Job Array

URLs and parameters

```
POST http://localhost:8080/mws/rest/job-arrays?api-version=3[&proxy-user=<username>]
```

Parameter	Required	Type	Valid values	Description
proxy-user	No	String	--	Perform this action as this user.

See Global URL Parameters for available URL parameters.

Request body

To submit a job array, only two fields are required: `jobPrototype` and one of `indexValues` or `indexRanges`. Both index ranges and values may be specified if desired.

The request body below shows all the fields that are available during job array submission, although the `jobPrototype` shown is a simple example and does not utilize all fields of a job submission.

The `jobPrototype` field has the same properties as a typical job submission. Consequently the api-version of the job array will apply to the `jobPrototype` like it does when you submit jobs, so the api-version in the call must match the api-version of the job. Examples of this can be seen in Submitting Jobs.

```
JSON request body
-------------------------------------

{
  "name": "myarray",
  "indexRanges": [  {
    "startIndex": 11,
    "endIndex": 25,
    "increment": 2
  }],
  "indexValues":   [ 2, 4, 6, 8, 10 ],
  "slotLimit": 2,
  "cancellationPolicy":   {
    "firstJob": "FAILURE",
    "anyJob": "SUCCESS"
  },
  "jobPrototype":   {
    "commandFile": "/tmp/test.sh",
    "initialWorkingDirectory": "/tmp",
    "requirements": [{"taskCount": 4}]
  }
}
```

Sample response

The response of this task is the same as submitting a job (see Submit Job).

Related Topics

- Fields: Job Arrays
- Resources Introduction
- Jobs
- Job Templates

Jobs

This section describes behavior of the **Job** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The Job API is new with *API version 2*. The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3` in order to behave as documented.

For more information, see Requesting Specific API Versions.

The **Fields: Jobs** reference contains the type and description of all fields in the **Job** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported Methods

Resource	GET	PUT	POST	DELETE
/rest/jobs	Get All Jobs	--	Submit Job	--
/rest/jobs/<name>	Get Single Job Get Job Priority Information Get Job Analysis Information	Generic Resources Modify Job Attributes	--	Cancel Job
/rest/jobs/<name>/<modifyAction>	--	Perform Actions on Job	--	--

Detailed information on each of the supported methods are provided later in this topic, see:

- Getting Job Information
  - Get All Jobs
  - Get Single Job
  - Get Job Priority Information
  - Get Job Analysis Information
- Submitting Jobs
  - Submit Job
- Modifying Jobs
  - Modify Job Attributes
  - Generic Resources
  - Perform Actions on Job
- Deleting (Canceling) Jobs
  - Cancel Job

Getting Job Information

The HTTP GET method is used to retrieve **Job** information. You can also use append the command with priority-analysis=true or node-analysis=true to get priority or eligibility information about the job.

Quick reference

```
GET http://localhost:8080/mws/rest/jobs/<name>?api-version=3
```

Get All Jobs

URLs and parameters

```
GET http://localhost:8080/mws/rest/jobs?api-version=3
```

Parameter	Required	Type	Description	Example
query	No	JSON	Queries for specific results. It is possible to query by one or more fields based on MongoDB query syntax.	`query={"isAct-ive":true}`
sort	No	JSON	Sort the results. Use 1 for ascending and -1 for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

How to get all jobs

```
GET http://localhost:8080/mws/rest/jobs?api-version=3&fields=name,flags&max=3
-------------------------------------

{
  "totalCount": 8,
  "resultCount": 3,
  "results":   [
        {
      "flags": ["GLOBALQUEUE"],
      "name": "Moab.1"
    },
        {
      "flags": ["GLOBALQUEUE"],
      "name": "Moab.2"
    },
        {
      "flags": ["GLOBALQUEUE"],
      "name": "Moab.4"
    }
  ]
}
```

How to get a subset of jobs

```
Get active jobs
-------------------------------------

http://localhost:8080/mws/rest/jobs?api-version=3&query={"isActive":true}
```

```
Get completed jobs
----------------------------------

http://localhost:8080/mws/rest/jobs?api-version=3&query={"isActive":false}
```

```
Get jobs owned by a particular user
----------------------------------

http://localhost:8080/mws/rest/jobs?api-version=3&query={"credentials.user":"fred"}
```

Known issues

Some jobs are not returned if `DisplayFlags UseBlocking` is set in the `moab.cfg` file.

Get Single Job

URLs and Parameters

```
GET http://localhost:8080/mws/rest/jobs/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the object.

See Global URL Parameters for available URL parameters.

The `attributes` field is only applicable in API version 2 and later, and the `MOAB_TENANT` field only applies if the job is attached to a tenant.

Sample response

JSON response


```
{
  "arrayIndex": null,
  "arrayMasterName": null,
  "attributes": [],
  "blocks": [  {
    "category": "jobBlock",
    "message": null,
    "type": null
  }],
  "bypassCount": 0,
  "cancelCount": 0,
  "commandFile": "/tmp/test.sh",
  "commandLineArgs": null,
  "completionCode": null,
  "cpuTime": 0,
  "credentials":   {
    "account": null,
    "group": "adaptive",
    "jobClass": null,
    "qos": "NONE",
    "qosRequested": null,
    "user": "adaptive"
  },
  "customName": null,
  "dates":   {
    "completedDate": null,
    "createdDate": "2012-10-11 17:58:16 UTC",
    "deadlineDate": "2037-10-24 12:26:40 UTC",
    "dispatchedDate": null,
    "earliestRequestedStartDate": null,
    "earliestStartDate": "2012-10-11 17:58:18 UTC",
    "eligibleDate": "2012-10-11 17:59:19 UTC",
    "lastCanceledDate": null,
    "lastChargedDate": null,
    "lastPreemptedDate": null,
    "lastUpdatedDate": "2012-10-11 17:59:19 UTC",
    "startDate": null,
    "submitDate": "2012-10-11 17:58:16 UTC",
    "terminationDate": "2037-10-24 12:26:40 UTC"
  },
  "deferCount": 0,
  "dependencies": [],
  "description": null,
  "duration": 8639999,
  "durationActive": 0,
  "durationQueued": 31,
  "durationRemaining": 0,
  "durationSuspended": 0,
  "emailNotifyAddresses": [],
  "emailNotifyTypes": [],
  "environmentRequested": false,
  "environmentVariables": {},
  "epilogScript": null,
  "flags": ["GLOBALQUEUE"],
  "holdDate": null,
  "holdReason": null,
  "holds": [],
  "initialWorkingDirectory": "/tmp",
  "isActive": true,
```

```
  "jobGroup": null,
  "masterNode": null,
  "memorySecondsDedicated": 0,
  "memorySecondsUtilized": 0,
  "messages": [],
  "migrateCount": 0,
  "minimumPreemptTime": 0,
  "mwmName": "Moab",
  "name": "Moab.15",
  "nodesExcluded": [],
  "nodesRequested": [],
  "nodesRequestedPolicy": null,
  "partitionAccessList":   [
    "msm",
    "SHARED"
  ],
  "partitionAccessListRequested":   [
    "msm",
    "SHARED"
  ],
  "preemptCount": 0,
  "priorities":   {
    "run": 0,
    "start": 1,
    "system": 0,
    "user": 0
  },
  "processorSecondsDedicated": 0,
  "processorSecondsLimit": 0,
  "processorSecondsUtilized": 0,
  "prologScript": null,
  "queueStatus": "blocked",
  "rejectPolicies": [],
  "requirements": [  {
    "architecture": null,
    "attributes":{
      "matlab":   [
             {
           "restriction":"must",
           "comparator": "<=",
           "value": "7.1",
           "displayValue": null
        }
      ],
      "MOAB_TENANT":   [   {
           "value": "1234567890aabbccddeeff00",
           "displayValue": "ResearchGroup"
      }],
      "soffice":   [
             {
           "restriction":"must",
           "comparator": "%=",
           "value": "3.1",
           "displayValue": null
        }
      ]
    },
    "features": [],
    "index": 0,
    "featuresRequested": [],
    "featuresRequestedMode": "AND",
    "featuresExcluded": [],
```

```
    "featuresExcludedMode": "AND",
    "metrics": {},
    "nodeAccessPolicy": null,
    "nodeAllocationPolicy": null,
    "nodeCount": 0,
    "nodes": [],
    "nodeSet": null,
    "image": null,
    "reservation": null,
    "resourcesPerTask":     {
      "processors":       {
        "dedicated": 1,
        "utilized": 0
      },
      "memory":       {
        "dedicated": 0,
        "utilized": 0
      },
      "disk":       {
        "dedicated": 0,
        "utilized": null
      },
      "swap":       {
        "dedicated": 0,
        "utilized": null
      }
    },
    "taskCount": 4,
    "tasksPerNode": 0
  }],
  "reservationRequested": null,
  "resourceFailPolicy": null,
  "resourceManagerExtension": null,
  "resourceManagers": [  {
    "isDestination": false,
    "isSource": true,
    "jobName": "Moab.15",
    "name": "internal"
  }],
  "rmStandardErrorFilePath": null,
  "rmStandardOutputFilePath": null,
  "shellName": "/bin/bash",
  "standardErrorFilePath": null,
  "standardOutputFilePath": null,
  "startCount": 0,
  "states":   {
    "state": "Idle",
    "stateExpected": "Idle",
    "stateLastUpdatedDate": null,
    "subState": null
  },
  "submitHost": "0:0:0:0:0:0:0:1",
  "systemJobAction": null,
  "systemJobType": null,
  "targetedJobAction": null,
  "targetedJobName": null,
  "templates": [{"name": "DEFAULT"}],
  "triggers": [],
  "variables": {},
  "virtualContainers": [],
  "virtualMachines": [],
  "vmUsagePolicy": null
```

```
}
```

Job arrays

- If a job is the master of a job array, the response will have some additional fields set as shown in the following example. The **name** field is chosen by the Moab, and the **customName** field comes from the **Fields: Job Arrays** `name` field.

```
Job array master
-------------------------------------

{
  "name": "Moab.5",
  "customName": "myarray",
  "flags":   [
    "ARRAYMASTER",
    "GLOBALQUEUE",
    "CANCELONFIRSTFAILURE",
    "CANCELONANYSUCCESS"
  ]
}
```

- If a job is a sub-job of an array, the response will have other fields set as shown in the following example.

```
Array sub-job
-------------------------------------

{
  "name": "Moab.5[21]",
  "customName": "myarray",
  "arrayIndex": 21,
  "arrayMasterName": "Moab.5",
  "flags":   [
    "ARRAYJOB",
    "GLOBALQUEUE",
    "CANCELONFIRSTFAILURE",
    "CANCELONANYSUCCESS"
  ]
}
```

Get Job Priority Information

The priority-analysis parameter is used to get job priority information.

URLs and Parameters

```
GET http://localhost:8080/mws/rest/jobs/<name>?api-version=3&priority-analysis=true
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the job.

See Global URL Parameters for available URL parameters.

Sample Response

```
JSON response
------------------------------------
{
    priorities: {
        start: 36,
        system: 0,
        components: {
            service: {
                weight: 2,
                queuetime: {
                    weight: 1,
                    value: 33
                },
                xfactor: {
                    weight: 0,
                    value: 1.559722
                },
                deadline: {
                    weight: 0,
                    value: 0
                },
                policyviolation: {
                    weight: 0,
                    value: 0
                },
                userprior: {
                    weight: 3,
                    value: -5
                },
                startcount: {
                    weight: 0,
                    value: 0
                },
                bypass: {
                    weight: 0,
                    value: 0
                }
            },
            target: {
                weight: 1,
                queuetime: {
                    weight: 0,
                    value: 0
                },
                xfactor: {
                    weight: 0,
                    value: 0
                }
            },
            credential: {
                weight: 1,
                user: {
                    weight: 0,
                    value: 0
                },
                group: {
                    weight: 0,
                    value: 0
                },
                account: {
                    weight: 0,
                    value: 0
                },
                qos: {
                    weight: 0,
                    value: 0
                }
```

```
            },
            attribute: {
                weight: 1,
                attribute: {
                    weight: 0,
                    value: 0
                },
                gres: {
                    weight: 0,
                    value: 0
                },
                jobid: {
                    weight: 0,
                    value: 0
                },
                jobname: {
                    weight: 0,
                    value: 0
                },
                state: {
                    weight: 0,
                    value: 0
                }
            },
            fairshare: {
                weight: 1,
                user: {
                    weight: 0,
                    value: 0
                },
                group: {
                    weight: 0,
                    value: 0
                },
                account: {
                    weight: 0,
                    value: 0
                },
                qos: {
                    weight: 0,
                    value: 0
                },
                guser: {
                    weight: 0,
                    value: 0
                },
                ggroup: {
                    weight: 0,
                    value: 0
                },
                gaccount: {
                    weight: 0,
                    value: 0
                },
                userwcacc: {
                    weight: 0,
                    value: 0
                },
                jobsperuser: {
                    weight: 0,
                    value: 0
                },
                jobsrunningperuser: {
                    weight: 0,
                    value: 0
                },
                procsperuser: {
```

```
                    weight: 0,
                    value: 0
                },
                psperuser: {
                    weight: 0,
                    value: 0
                }
            },
            resource: {
                weight: 1,
                node: {
                    weight: 0,
                    value: 0
                },
                proc: {
                    weight: 0,
                    value: 1
                },
                memory: {
                    weight: 0,
                    value: 0
                },
                swap: {
                    weight: 0,
                    value: 0
                },
                disk: {
                    weight: 0,
                    value: 0
                },
                procsecond: {
                    weight: 0,
                    value: 3600
                },
                procequivalent: {
                    weight: 0,
                    value: 1
                },
                walltime: {
                    weight: 0,
                    value: 3600
                }
            },
            usage: {
                weight: 1,
                consumed: {
                    weight: 0,
                    value: 0
                },
                remaining: {
                    weight: 0,
                    value: 0
                },
                percentconsumed: {
                    weight: 0,
                    value: 0
                },
                executiontime: {
                    weight: 0,
                    value: 0
                }
            }
        }
    }
}
```

Get Job Analysis Information

The job-analysis parameter is used to get an analysis of the job's eligibility to run on the nodes managed by Moab.

URLs and parameters

```
GET http://localhost:8080/mws/rest/jobs/<name>?api-version=3&job-analysis=true
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	Name of the job.

See Global URL Parameters for available URL parameters.

Sample Response

```
JSON response
-----------------------------------
{
  "name": "37",
  "warnings":   [
    "job cannot run  (job has hold in place)",
    "job cannot run  (insufficient available procs:  0 available)"
  ],
  "nodes": [  {
    "name": "node01",
    "message": "node01 rejected: State (Busy)"
  }]
}
```

Submitting Jobs

The HTTP POST method is used to submit **Jobs**.

Quick reference

```
POST http://localhost:8080/mws/rest/jobs?api-version=3[&proxy-user=<username>]
```

Restrictions

- No more than one virtual container can be specified in the request. The virtual container must already exist.
- The `credentials.user` and `credentials.group` properties are used to submit a job as the specified user belonging to the specified group.
- Job `variables` have the following restrictions:
  - Variable names cannot contain equals (=), semicolon (;), colon (:), plus (+), question mark (?), caret (^), backslash (\), or white space.
  - Variable values cannot contain semicolon (;), colon (:), plus (+), or caret (^).

- When submitting jobs, the only supported `hold` type is `User`.
- The proxy-user parameter is ignored unless you set `ENABLEPROXY=TRUE` in the `moab.cfg` file. For example:

```
ADMINCFG[1]           USERS=root,ted ENABLEPROXY=TRUE
```

Submit Job

URLs and parameters

```
POST http://localhost:8080/mws/rest/jobs?api-version=3[&proxy-user=<username>]
```

Parameter	Required	Type	Valid values	Description
proxy-user	No	String	--	Perform the action as this user.

See Global URL Parameters for available URL parameters.

Request body

```
JSON request body (specified host list)
---------------------------------------

{
  "attributes":   [
    "attr1",
    "attr2"
  ],
  "commandFile": "/tmp/test.sh",
  "commandScript": "c2xlZXAgNjAK",
  "commandLineArguments": "-x -v",
  "credentials":   {
    "account": "account",
    "group": "group",
    "jobClass": "BATCH",
    "qosRequested": "QOS1",
    "user": "saadmin"
  },
  "customName": "custom name for job",
  "dates":   {
    "earliestRequestedStartDate": "2012-11-08 13:18:47 UTC",
    "deadlineDate": "2014-02-17 14:00:00 UTC"
  },
  "dependencies":   [
        {
      "type": "set",
      "name": "vc1.varA"
    },
        {
      "type": "set",
      "name": "vc2.varB"
    },
        {
      "type": "set",
      "name": "vc3.varC"
    }
  ],
  "duration": 600,
  "emailNotifyAddresses":   [
    "user3@ac.com",
    "user4@ac.com"
  ],
  "emailNotifyTypes":   [
    "JobStart",
    "JobEnd"
  ],
  "environmentRequested": true,
  "environmentVariables":   {
    "var1": "val1",
    "var2": "val2"
  },
  "epilogScript": "/tmp/epilog.sh",
  "flags":   [
    "RESTARTABLE",
    "SUSPENDABLE"
  ],
  "holds": ["User"],
  "initialWorkingDirectory": "/tmp",
  "jobGroup": "job_group",
  "nodesExcluded":   [
    {"name": "node07"},
    {"name": "node08"}
```

```
  ],
  "nodesRequested":   [
    {"name": "node01"},
    {"name": "node02"}
  ],
  "nodesRequestedPolicy": "SUBSET",
  "partitionAccessListRequested":   [
    "p1",
    "p2"
  ],
  "priorities": {"user": 5},
  "prologScript": "/tmp/prolog.sh",
  "requirements": [  {
    "architecture": "x86_64",
    "attributes":{
      "matlab":   [
             {
           "restriction":"must",
           "comparator": "<=",
           "value": "7.1"
        }
      ],
      "soffice":   [
             {
           "restriction":"must",
           "comparator": "%=",
           "value": "3.1"
        }
      ]
    },
    "featuresRequested":     [
      "a",
      "b",
      "c"
    ],
    "featuresRequestedMode": "OR",
    "featuresExcluded":     [
      "d",
      "e",
      "f"
    ],
    "featuresExcludedMode": "AND",
    "nodeAccessPolicy": "SINGLEJOB",
    "nodeAllocationPolicy": "PRIORITY",
    "nodeCount": 6,
    "nodeSet":"FIRSTOF:FEATURE:vlan2",
    "image": "linux",
    "resourcesPerTask":     {
      "disk": {"dedicated": 1024},
      "memory": {"dedicated": 512},
      "processors": {"dedicated": 2},
      "swap": {"dedicated": 4096},
      "matlab": {"dedicated": 6},
      "intellij": {"dedicated": 2}
    },
    "taskCount": 4,
    "tasksPerNode": 14
  }],
  "reservationRequested": {"name": "rsv.1"},
  "resourceFailPolicy": "RETRY",
  "resourceManagerExtension": "x=PROC=4",
  "shellName": "/bin/bash",
```

```
   "standardErrorFilePath": "/tmp/error",
   "standardOutputFilePath": "/tmp/out",
   "submitHost": "admin-node",
   "templates":   [
     {"name": "template1"},
     {"name": "template2"}
   ],
   "variables":   {
     "var1": "val1",
     "var2": "val2"
   },
   "virtualContainers": [{"name": "vc1"}],
   "vmUsagePolicy": "CREATEVM"
 }
```

Sample response

The response of this task is one of three possibilities:

- An object with a single **messages** property containing a list of error messages on failure.

  ```
  {"messages":["Could not create job - invalid requirements"]}
  ```

- An object with a **name** property containing the name of the newly created job.

  ```
  {"name":"Moab.1"}
  ```

- An object with a **name** property and a **virtualContainers** list containing the name of the newly created virtual container.

  ```
  { "name": "Moab.1", "virtualContainers": [{"name": "vc1"}] }
  ```

  > The virtual container will only be reported when a *new* virtual container has been created by Moab for the job.

Examples of job submission

This section includes some sample job submission requests.

Example 4-3: Submit job to run on node2 and node3

```
POST http://localhost:8080/mws/rest/jobs?api-version=3
-----------------------------------

{
  "commandFile": "/tmp/test.sh",
  "credentials":   {
    "group": "adaptive",
    "user": "adaptive"
  },
  "initialWorkingDirectory": "/tmp",
  "nodesRequested":   [
    {"name": "node2"},
    {"name": "node3"}
  ]
}
```

Example 4-4: Submit job that requires 20 processors

```
POST http://localhost:8080/mws/rest/jobs?api-version=3
-----------------------------------

{
  "commandFile": "/tmp/test.sh",
  "credentials":   {
    "group": "adaptive",
    "user": "adaptive"
  },
  "initialWorkingDirectory": "/tmp",
  "requirements": [{"taskCount": 20}]
}
```

Example 4-5: Submit job to run after a certain time

```
POST http://localhost:8080/mws/rest/jobs?api-version=3
-----------------------------------

{
  "commandFile": "/tmp/test.sh",
  "credentials":   {
    "group": "adaptive",
    "user": "adaptive"
  },
  "dates": {"earliestRequestedStartDate": "2012-10-11 18:36:35 UTC"},
  "initialWorkingDirectory": "/tmp",
  "requirements": [{"taskCount": 20}]
}
```

Example 4-6: Submit job based on `msub` example

Given this msub command:

```
msub -l nodes=3:ppn=2,walltime=1:00:00,pmem=100 script2.pbs.cmd
```

Here is an equivalent MWS request:

```
POST http://localhost:8080/mws/rest/jobs?api-version=3
-------------------------------------

{
  "duration": 3600,
  "commandFile": "/home/adaptive/script2.pbs.cmd",
  "credentials":   {
    "group": "adaptive",
    "user": "adaptive"
  },
  "initialWorkingDirectory": "/home/adaptive",
  "requirements": [  {
    "resourcesPerTask": {"memory": {"dedicated": 100}},
    "taskCount": 6,
    "tasksPerNode": 2
  }]
}
```

To emulate what msub does, make `commandFile` an absolute path, and add `credentials.user`, `credentials.group`, and `initialWorkingDirectory`.

As shown above, `nodes=3:ppn=2` is equivalent to setting `taskCount` to 6 and `tasksPerNode` to 2.

Example 4-7: Submit a job array

For information on how to submit a job array, see Submitting Job Arrays.

Modifying Jobs

The HTTP PUT method is used to modify **Jobs**.

Quick reference

```
PUT http://localhost:8080/mws/rest/jobs/<name>[/<modifyAction>]?api-version=3[&proxy-
user=<username>]
```

Restrictions

The **proxy-user** parameter is ignored unless you set `ENABLEPROXY=TRUE` in the `moab.cfg` file. For example:

```
ADMINCFG[1]            USERS=root,ted ENABLEPROXY=TRUE
```

Modify Job Attributes

URLs and parameters

```
PUT http://localhost:8080/mws/rest/jobs/<name>?api-version=3[&proxy-user=<username>]
[&change-mode=set]
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the object.
proxy-user	No	String	--	Perform the action as this user.

See Global URL Parameters for available URL parameters.

Additional URL parameters

Parameter	Required	Valid values	Description
change-mode	No	set (default) add remove	If **set**, replace all fields with the fields specified. If **add**, add the specified fields to existing fields. If **remove**, remove the specified fields from existing fields.

Request body

The request body below shows all the fields that are available when modifying a job, along with some sample values.

```
JSON request body
-------------------------------------

{
  "credentials":   {
    "account": "account",
    "jobClass": "BATCH",
    "qosRequested": "QOS1"
  },
  "customName": "custom name for job",
  "dates": {"earliestRequestedStartDate": "2012-11-08 13:18:47 UTC"},
  "duration": 600,
  "flags":   [
    "RESTARTABLE",
    "SUSPENDABLE"
  ],
  "holds": ["User"],
  "messages":   [
    {"message": "Message one"},
    {"message": "Message two"}
  ],
  "nodesRequested":   [
    {"name": "n015"},
    {"name": "n016"},
    {"name": "n017"},
    {"name": "n018"}
  ],
  "partitionAccessListRequested":   [
    "p1",
    "p2"
  ],
  "priorities":   {
    "system": 3,
    "user": 5
  },
  "requirements": [  {
    "features":     [
      "vlan1",
      "vlan2"
    ],
    "resourcesPerTask":     {
      "matlab": {"dedicated": 1},
      "tape": {"dedicated": 2}
    }
  }],
  "reservationRequested": {"name": "rsv.1"},
  "variables":   {
    "var1": "val1",
    "var2": "val2"
  }
}
```

Sample response

These messages may not match the messages returned from Moab exactly, but are given as an example of the structure of the response.

Not all messages are shown for the above request body.

```
JSON response
-------------------------------------

{"messages": [
  "Account modified successfully",
  "Messages modified successfully",
  "Variables modified successfully"
]}
```

Restrictions

- Old messages are not removed from jobs; only new messages are added.
- Job `variables` have the restrictions documented in Submitting Jobs.
- Although the client can modify `features` and `resourcesPerTask`, Moab only considers these elements when they appear in the first element of the `requirements` array. If the requirements array contains two or more elements, all elements but the first are silently ignored.

Generic Resources

Jobs can require configurable, site-specific consumable resources called generic resources. For example, some jobs may require a matlab license. Only one job at a time may legally consume this license. Matlab is not a standard resource and may only be available on some sites. Nevertheless Moab allows this to be configured and tracked as is explained in Managing Consumable Generic Resources.

You must specify generic resources in the `requirements.resourcesPerTask` portion of the JSON document. Any resource in `requirement.resourcesPerTask` that is not a standard resource is considered a generic resource. Standard resources include disk, memory, processors, and swap. Assume a job has the following in `requirement.resourcesPerTask`:

```
{
  "resourcesPerTask":{
    "processors":{
      "dedicated":4,
      "utilized":0
    },
    "memory":{
      "dedicated":2048,
      "utilized":0
    },
    "disk":{
      "dedicated":4096,
      "utilized":0
    },
    "swap":{
      "dedicated":1024,
      "utilized":0
    },
    "tape":{
      "dedicated":1,
      "utilized":0
    },
    "matlab":{
      "dedicated":2,
      "utilized":0
    }
  }
}
```

The standard resources the job requires are:

- 4 processors
- 2048 MB of memory
- 4096 MB of disk
- 1024 MB of swap

The generic resources the job requires are

- 1 tape
- 2 matlab

To modify a job so that it requires 1 matlab license, run the following:

```
PUT http://localhost:8080/mws/rest/jobs/Moab.2?api-version=3
{
  "requirements":[
    {
      "resourcesPerTask":{
        "matlab":{
          "dedicated":1
        }
      }
    }
  ]
}
```

Perform Actions on Job

URLs and parameters

```
PUT http://localhost:8080/mws/rest/jobs/<name>/<modifyAction>?api-version=3[&proxy-user=<username>]
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the object.
modifyAction	Yes	String	cancel checkpoint execute hold requeue rerun resume suspend unhold	If **cancel**, attempts to cancel the job (equivalent to deleting a job). If **checkpoint**, attempts to checkpoint the job. Note that the OS must support checkpointing for this to work. If **execute**, executes the job (if possible). If **hold**, attempts to hold the job using the holds set in the request body. If **requeue**, attempts to requeue the job. If **rerun**, attempts to rerun the job. If **resume**, attempts to resume the job. If **suspend**, attempts to suspend the job. If **unhold**, attempts to release the holds set in the request body.
proxy-user	No	String	--	Perform the action as this user.

See Global URL Parameters for available URL parameters.

Request body

Request bodies are only required for holding or unholding jobs. All other actions do not require request bodies of any kind.

```
JSON request body to add holds to a job
-----------------------------------

{"holds": ["User"]}
```

```
JSON request body to remove holds from a job
-----------------------------------

{"holds": ["User"]}
```

If no holds are specified when unholding a job, all holds will be removed. This is equivalent to specifying `holds` as a list with a single element of `All`.

Sample response

This message may not match the message returned from Moab exactly, but is given as an example of the structure of the response.

```
JSON response
-------------------------------------

{"messages": ["Job modified successfully"]}
```

Deleting (Canceling) Jobs

The HTTP DELETE method is used to cancel **Jobs**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/jobs/<name>?api-version=3[&proxy-
user=<username>]
```

Restrictions

The **proxy-user** parameter is ignored unless you set `ENABLEPROXY=TRUE` in the `moab.cfg` file. For example:

```
ADMINCFG[1]           USERS=root,ted ENABLEPROXY=TRUE
```

Cancel Job

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/jobs/<name>?api-version=3[&proxy-
user=<username>]
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the object.
proxy-user	No	String	--	Perform the action as this user.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response for successful DELETE
-------------------------------------

{}
```

Additional information about the DELETE can be found in the HTTP response header `X-MWS-Message`.

Related Topics

- Fields: Jobs
- Resources Introduction
- Job Arrays
- Job Templates

Job Templates

This section describes behavior of the **Job Template** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Job Templates** reference section contains the type and description of all fields in the **Job Template** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/job-templates	Get All Job Templates	--	--	--
/rest/job-templates/<id>	Get Single Job Template	--	--	--

This topic contains these sections:

- Getting Job Templates
  - Get All Job Templates
  - Get Single Job Template

Getting Job Templates

The HTTP GET method is used to retrieve **Job Template** information. Queries for all objects and a single object are available.

Quick reference

```
GET http://localhost:8080/mws/rest/job-templates/<id>?api-version=3
```

Get All Job Templates

URLs and parameters

```
GET http://localhost:8080/mws/rest/job-templates?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/job-templates?api-version=3&fields=id
-------------------------------------

{
  "totalCount": 14,
  "resultCount": 14,
  "results":   [
    {"id": "DEFAULT"},
    {"id": "genericVM"},
    {"id": "genericVM-setup"},
    {"id": "genericVM-destroy"},
    {"id": "genericVM-migrate"},
    {"id": "genericPM"},
    {"id": "genericPM-setup"},
    {"id": "genericPM-destroy"},
    {"id": "OSStorage"},
    {"id": "OSStorage-setup"},
    {"id": "OSStorage-destroy"},
    {"id": "extraStorage"},
    {"id": "extraStorage-setup"},
    {"id": "extraStorage-destroy"}
  ]
}
```

Get Single Job Template

URLs and parameters

```
GET http://localhost:8080/mws/rest/job-templates/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{
  "account": "account",
  "args": "arg1 arg2",
  "commandFile": "/tmp/script",
  "description": "description",
  "genericSystemJob": true,
  "id": "genericVM",
  "inheritResources": false,
  "jobDependencies": [  {
    "name": "genericVM-setup",
    "type": "JOBSUCCESSFULCOMPLETE"
  }],
  "jobFlags": ["VMTRACKING"],
  "jobTemplateFlags": ["SELECT"],
  "jobTemplateRequirements": [  {
    "architecture": "x86_64",
    "diskRequirement": 500,
    "genericResources": {"tape": 3},
    "nodeAccessPolicy": "SINGLEJOB",
    "operatingSystem": "Ubuntu 10.04.3",
    "requiredDiskPerTask": 200,
    "requiredFeatures": ["dvd"],
    "requiredMemoryPerTask": 1024,
    "requiredProcessorsPerTask": 2,
    "requiredSwapPerTask": 512,
    "taskCount": 4
  }],
  "priority": 20,
  "qos": "qos",
  "queue": "queue",
  "durationRequested": 600,
  "select": true,
  "trigger": null,
  "version": 0,
  "vmUsagePolicy": "REQUIREPM"
}
```

Related Topics

- Fields: Job Templates
- Resources Introduction
- Jobs
- Job Arrays

Metric Types

This section describes behavior of the **Metric Type** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Metric Types** reference section contains the type and description of all fields in the **Metric Type** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/metric-types	Get All Metric Types	--	--	--

This topic contains these sections:

- Getting Metric Types
  - Get All Metric Types

Getting Metric Types

The HTTP GET method is used to retrieve **Metric Type** information.

Quick reference

```
GET http://localhost:8080/mws/rest/metric-types?api-version=3
```

Get All Metric Types

URLs and parameters

```
GET http://localhost:8080/mws/rest/metric-types?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/metric-types?api-version=3&fields=id
-----------------------------------

{
  "totalCount": 9,
  "resultCount": 9,
  "results":   [
    {"id": "vmcount"},
    {"id": "watts"},
    {"id": "pwatts"},
    {"id": "temp"},
    {"id": "cpu"},
    {"id": "mem"},
    {"id": "io"},
    {"id": "ccores"},
    {"id": "threads"}
  ]
}
```

Related Topics

- Fields: Metric Types
- Resources Introduction

Nodes

This section describes behavior of the **Node** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The Node API is new with *API version 2*. The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3` in order to behave as documented.

For more information, see Requesting Specific API Versions.

The **Fields: Nodes** reference contains the type and description of all fields in the **Node** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/nodes	Get All Nodes	--	--	--
/rest/nodes/<name>	Get Single Node	Modify Node	--	--

This topic contains these sections:

- Getting Nodes
  - Get All Nodes
  - Get Single Node
- Modifying Nodes
  - Modify Node

Getting Nodes

The HTTP GET method is used to retrieve **Node** information.

Quick reference

```
GET http://localhost:8080/mws/rest/nodes/<name>?api-version=3
```

Get All Nodes

URLs and parameters

```
GET http://localhost:8080/mws/rest/nodes?api-version=3
```

Parameter	Required	Type	Description	Example
query	No	JSON	Queries for specific results. It is possible to query by one or more fields based on MongoDB query syntax.	`query={"type":"-compute"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

> This query will not return the `DEFAULT` or `GLOBAL` nodes from Moab. However, the **Get Single Node** task may be used to retrieve them individually if desired.

Sample response

```
GET http://localhost:8080/mws/rest/nodes?api-version=3&fields=name
-----------------------------------

{
  "totalCount": 3,
  "resultCount": 3,
  "results":   [
    {"name": "node1"},
    {"name": "node2"},
    {"name": "node3"}
  ]
}
```

Get Single Node

URLs and parameters

```
GET http://localhost:8080/mws/rest/nodes/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the object.

See Global URL Parameters for available URL parameters.

The `attributes` field is only applicable in API version 2 and later, and the `MOAB_TENANT` field only applies if the node is attached to a tenant.

Sample response

```
JSON response
-------------------------------------

{
  "name": "l26.csa",
  "architecture": null,
  "classes": ["class1"],
  "attributes":   {
    "MOAB_TENANT":   {
      "value": "1234567890abcdef12345678",
      "displayValue": "ResearchGroup"
    },
    "MOAB_DATACENTER":   {
      "value": "vcenter-datacenter-401",
      "displayValue": "vcenter-vcenter - adaptive data center"
    },
    "vcenter-vcenter-adaptive data center-compute nodes":     {
      "value": null,
      "displayValue": null
    }
  },
  "featuresCustom": ["feature1", "feature2"],
```

```
  "featuresReported": ["vcenter-vcenter-adaptive data center-compute nodes"],
  "index": 26,
  "ipAddress": "10.0.8.76",
  "isHypervisor": true,
  "lastUpdatedDate": "2013-05-24 20:18:11 UTC",
  "migrationDisabled": false,
  "partition": "mws",
  "processorSpeed": null,
  "profilingEnabled": false,
  "rack": null,
  "resourceManagerMessages":   {
    "torque": null,
    "mws": null
  },
  "slot": null,
  "type": "compute",
  "messages": [  {
    "count": 11,
    "createdDate": "2012-10-24 04:06:04 UTC",
    "expireDate": "2037-10-24 12:26:40 UTC",
    "message": "This is a message"
  }],
  "metrics":   {
    "vmcount": 0,
    "cpuUtilization": 0.275,
    "cpuLoad": 0.01115
  },
  "variables":   {
    "VCENTER_DATASTORE_LOCAL1": "datastore-415",
    "VCENTER_DATASTORE_REMOTE1": "datastore-448"
  },
  "states":   {
    "powerState": "On",
    "powerStateExpected": null,
    "state": "Idle",
    "stateExpected": "Idle",
    "stateLastUpdatedDate": "2013-05-24 09:33:45 UTC",
    "subState": null,
    "subStateLast": null,
    "subStateLastUpdatedDate": null
  },
  "operatingSystem":   {
    "hypervisorType": "esx",
    "image": "vcenter-vcenter-esx-5.0",
    "imageExpected": null,
    "imageLastUpdatedDate": null,
    "imagesAvailable": [],
    "virtualMachineImages":     [
      "win2008",
      "centos6"
    ]
  },
  "resources":   {
    "processors":     {
      "configured": 4,
      "real": 4,
      "dedicated": 0,
      "available": 4,
      "utilized": -1
    },
    "memory":     {
      "configured": 10239,
```

```
      "real": 10239,
      "dedicated": 0,
      "available": 9227,
      "utilized": 0
    },
    "disk":     {
      "configured": 0,
      "real": 0,
      "dedicated": 0,
      "available": 0,
      "utilized": 0
    },
    "swap":     {
      "configured": 0,
      "real": 0,
      "dedicated": 0,
      "available": 0,
      "utilized": 0
    }
  },
  "resourceManagers": [  {
    "name": "mws",
    "isMaster": true,
    "stateReported": "Active"
  }],
  "jobs": [],
  "reservations":   [
        {
      "name": "system.5",
      "type": "user"
    },
        {
      "name": "system.17",
      "type": "user"
    }
  ],
  "virtualContainers": [],
  "virtualMachines": [],
  "triggers": []
}
```


Modifying Nodes

The HTTP PUT method is used to modify **Nodes**.

Quick reference

```
PUT http://localhost:8080/mws/rest/nodes/<name>?api-version=3[&proxy-user=<username>]
```

Restrictions

The **proxy-user** parameter is ignored unless you set `ENABLEPROXY=TRUE` in the `moab.cfg` file. For example:

```
ADMINCFG[1]           USERS=root,ted ENABLEPROXY=TRUE
```

Modify Node

URLs and parameters

```
PUT http://localhost:8080/mws/rest/nodes/<name>?api-version=3[&proxy-user=<username>]
[&change-mode=set]
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the object.
proxy-user	No	String	--	Perform the action as this user.

See Global URL Parameters for available URL parameters.

Additional URL parameters

Parameter	Required	Valid values	Description
change-mode	No	set (default) add remove	If **set**, replace all features with the features specified. If **add**, add the specified features to existing features. If **remove**, remove the specified features from existing features.

Request body

The request body below shows all the fields that are available when modifying a node, along with some sample values.

```
Sample JSON request body to modify a node
-----------------------------------

{
  "featuresCustom": ["feature1", "feature2"],
  "messages":   [
    {"message": "Message one"},
    {"message": "Message two"}
  ],
  "metrics": {"pwatts": 211},
  "operatingSystem": {"image": "esx4.1"},
  "partition": "part1",
  "states":   {
    "powerState": "On",
    "state": "Running"
  },
  "variables":   {
    "key": "value",
    "arbitrary text key": "more value"
  }
}
```

Sample response

This message may not match the message returned from Moab exactly, but is given as an example of the structure of the response.

```
JSON response
-------------------------------------

{"messages":[
  "Successfully modified os to 'linux'",
  "Successfully powered node off"
]}
```

Related Topics

- Fields: Nodes
- Resources Introduction

Notification Conditions

This section describes behavior of the **Notification Conditions** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The Notification Conditions API is new with *API version 3*, and is not available with older API versions. The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3`.

For more information, see Requesting Specific API Versions.

The **Fields: Notification Conditions** reference contains the type and description of all fields in the **Notification Conditions** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/notification-conditions	Get All Notification Conditions	Update Notification Condition	--	--
/rest/notification-conditions/<id>	Get Single Notification Condition	--	--	--

This topic contains these sections:

- Getting Notification Conditions
  - Get All Notification Conditions
  - Get Single Notification Condition
- Updating Notification Conditions
  - Update Notification Condition

Getting Notification Conditions

The HTTP GET method is used to retrieve **Notification Condition** information.

Quick reference

```
GET http://localhost:8080/mws/rest/notification-conditions?api-version=3
GET http://localhost:8080/mws/rest/notification-conditions/<id>?api-version=3
```

Get All Notification Conditions

URLs and parameters

```
GET http://localhost:8080/mws/rest/notification-conditions?api-version=3[&query=
{"escalationLevel":"ADMIN"}][&sort={"observedDate":-1}]
```

Parameter	Required	Type	Description	Example
query	No	JSON	Query for specific results. It is possible to query notifications by one or more fields based on MongoDB query syntax.	`query={"escal-ationLevel":"ADMIN"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"observedDate":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/notification-conditions?api-version=3&query=
{"escalationLevel":"ADMIN"}&sort={"observedDate":-1}
-----------------------------------

 {
    "totalCount": 2,
    "resultCount": 2,
    "results": [
        {
            "createdDate": "2013-09-10 23:13:33 UTC",
            "details": {
                "pluginType": "NodeUtilizationReport",
                "pluginId": "node-report"
            },
            "escalationLevel": "ADMIN",
            "expirationDate": null,
            "expirationDuration": null,
            "message": "The node 'testnode' has not been updated since the last poll,
which is likely due to a misconfiguration.",
            "objectId": "testnode",
            "objectType": "Node",
            "observedDate": "2013-09-10 23:13:33 UTC",
            "origin": "MWS/plugins/NodeUtilizationReport/node-report",
            "tenant": {
                 "id":"1234567890abcdef12345678",
                 "name":"Research"
            },
            "id": "522fa79de4b0cafeaec6f83e"
        },
        {
            "createdDate": "2013-09-11 17:19:35 UTC",
            "details": {
                "pluginType": "VCenter",
                "pluginId": "vcenter42"
            },
            "escalationLevel": "ADMIN",
            "expirationDate": null,
            "expirationDuration": null,
            "message": "The node 'node1' does not have vcenter tools installed,
therefore the state is unknown and migrations may not work correctly",
            "objectId": null,
            "objectType": "System",
            "observedDate": "2013-09-11 17:19:35 UTC",
            "origin": "MWS/plugins/VCenter/vcenter42",
            "tenant": {
                 "id":"1234567890abcdef12345678",
                 "name":"Research"
            },
            "id": "5230a627e4b0d51bef490e86"
        }
    ]
}
```

A notification's `tenant` is automatically inherited from the `objectId` and `objectType` fields. If no object is associated with the notification condition, the notification is visible to all tenants.

Get Single Notification Condition

URLs and parameters

```
GET http://localhost:8080/mws/rest/notification-conditions/<id>?api-version=3
```

Parameter	Required	Type	Description
id	Yes	String	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/notification-
conditions/521a1f18e4b0e3f9031f47f5?api-version=3
-----------------------------------

{
    "createdDate": "2013-09-10 23:13:33 UTC",
    "details": {
        "pluginType": "NodeUtilizationReport",
        "pluginId": "node-report"
    },
    "escalationLevel": "ADMIN",
    "expirationDate": null,
    "expirationDuration": null,
    "message": "The node 'testnode' has not been updated since the last poll, which is
likely due to a misconfiguration.",
    "objectId": "testnode",
    "objectType": "Node",
    "observedDate": "2013-09-10 23:13:33 UTC",
    "origin": "MWS/plugins/NodeUtilizationReport/node-report",
    "tenant": {
        "id":"1234567890abcdef12345678",
        "name":"Research"
    },
    "id": "522fa79de4b0cafeaec6f83e"
}
```

A notification's `tenant` is automatically inherited from the `objectId` and `objectType` fields. If no object is associated with the notification condition, the notification is visible to all tenants.

Updating Notification Conditions

The HTTP PUT method is used to update **Notification Condition** information. The PUT operation is idempotent, meaning that is used for both creating new notification conditions and updating existing ones. If the `escalationLevel`, `origin`, `message`, `objectType`, and `objectId` fields match an existing notification condition, it will be updated. Otherwise, a new condition will be created.

Quick reference

```
PUT http://localhost:8080/mws/rest/notification-conditions?api-version=3
```

Update Notification Condition

URLs and parameters

```
PUT http://localhost:8080/mws/rest/notification-conditions?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

The request body below shows some fields that are available when updating a notification condition, along with some sample values.

```
Sample JSON request body to update a notification condition
----------------------------------

{
    "details": {
        "pluginType": "NodeTester",
        "pluginId": "my-tester1"
    },
    "escalationLevel": "ADMIN",
    "expirationDuration": 30,
    "message": "Node 'node2' is powered off, please check your hardware.",
    "objectId": "node2",
    "objectType": "Node",
    "origin": "NodeTester/my-tester1/Test.groovy:141"
}
```

Sample response

```
JSON response
----------------------------------

{
    "createdDate": "2013-09-10 23:13:33 UTC",
    "details": {
        "pluginType": "NodeTester",
        "pluginId": "my-tester1"
    },
    "escalationLevel": "ADMIN",
    "expirationDate": "2013-09-10 23:14:03 UTC",
    "expirationDuration": 30,
    "observedDate": "2013-09-10 23:13:33 UTC",
    "message": "Node 'node2' is powered off, please check your hardware.",
    "objectId": "node2",
    "objectType": "Node",
    "origin": "NodeTester/my-tester1/Test.groovy:141",
    "tenant": {
        "id":"1234567890abcdef12345678",
        "name":"Research"
    },
    "id": "5230a627e4b0d51bef490e86"
}
```

Related Topics

- Resources Introduction
- Events
- Fields: Events
- Notifications
- Fields: Notifications
- Fields: Notification Conditions
- Creating Events and Notifications
- Plugin Event Service
- Handling Events
- System Events
- Securing the Connection with the Message Queue

Notifications

This section describes behavior of the **Notifications** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The Notifications API is new with *API version 3*, and is not available with older API versions. The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3`.

For more information, see Requesting Specific API Versions.

The **Fields: Notifications** reference contains the type and description of all fields in the **Notifications** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/notifications/	**Get All Notifications**	--	--	--
/rest/notifications/<id>	**Get Single Notification**	--	--	--
/rest/notifications/ignore	--	**Ignore All Notifications**	--	--

Resource	GET	PUT	POST	DELETE
/rest/notifications/<id>/ignore	--	Ignore Single Notification	--	--
/rest/notifications/unignore	--	Unignore All Notifications	--	--
/rest/notifications/<id>/unignore	--	Unignore Single Notification	--	--
/rest/notifications/dismiss	--	Dismiss All Notifications	--	--
/rest/notifications/<id>/dismiss	--	Dismiss Single Notification	--	--

This topic contains these sections:

- Getting Notifications
  - Get All Notifications
  - Get Single Notification
- Ignoring Notifications
  - Ignore All Notifications
  - Ignore Single Notification
- Unignoring Notifications
  - Unignore All Notifications
  - Unignore Single Notification
- Dismissing Notifications
  - Dismiss All Notifications
  - Dismiss Single Notification

Getting Notifications

The HTTP GET method is used to retrieve **Notification** information.

Quick reference

```
GET http://localhost:8080/mws/rest/notifications?api-version=3
GET http://localhost:8080/mws/rest/notifications/<id>?api-version=3
```

Get All Notifications

URLs and parameters

```
GET http://localhost:8080/mws/rest/notifications?api-version=3[&proxy-user=<username>]
[&query={"ignoredDate":null,"dismissedDate":null}][&sort={"observedDate":-1}]
```

Parameter	Required	Type	Description	Example
proxy-user	No	String	Perform the action as this user. Notifications cannot be created directly. Instead, they are automatically created for the current user or `proxy-user` specified in the request from non-expired notification conditions (see Notification Conditions). This is true no matter the `query` specified.	--
query	No	JSON	Query for specific results. It is possible to query notifications by one or more fields based on MongoDB query syntax. However, typically you will want to query on `{"ignoredDate":null,"dismissedDate":null}`.	`query={"ignoredDate":null,"dismissedDate":null}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"observedDate":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/notifications?api-version=3&proxy-
user=<username>&query={"ignoredDate":null,"dismissedDate":null}][&sort=
{"observedDate":-1}
-----------------------------------

{
    "totalCount": 2,
    "resultCount": 2,
    "results": [
        {
            "conditionId": "521bdea1e4b019cd33e29c86",
            "createdDate": "2013-08-26 23:02:56 UTC",
            "details": {},
            "dismissedDate": null,
            "ignoredDate": null,
            "message": "A health check failed for the 'ZeroMQ Message Queue'
connection, please see the MWS health details page for more information.",
            "objectId": "zmq",
            "objectType": "Health",
            "observedDate": "2013-09-05 17:57:00 UTC",
            "origin": "MWS/HealthNotificationJob",
            "user": "admin",
            "id": "5230ed82e4b065347016d62f"
        },
        {
            "conditionId": "521a1f18e4b0e3f9031f47f5",
            "createdDate": "2013-08-25 15:13:28 UTC",
            "details": {},
            "dismissedDate": null,
            "ignoredDate": null,
            "message": "A health check failed for the 'LDAP' connection, please see
the MWS health details page for more information.",
            "objectId": "ldap",
            "objectType": "Health",
            "observedDate": "2013-08-30 18:11:15 UTC",
            "origin": "MWS/HealthNotificationJob",
            "user": "admin",
            "id": "5230ed82e4b065347016d60d"
        }
    ]
}
```

Get Single Notification

URLs and parameters

```
GET http://localhost:8080/mws/rest/notifications/<id>?api-version=3
```

Parameter	Required	Type	Description
id	Yes	String	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d?api-
version=3
-----------------------------------

{
    "conditionId": "521a1f18e4b0e3f9031f47f5",
    "createdDate": "2013-08-25 15:13:28 UTC",
    "details": {},
    "dismissedDate": null,
    "ignoredDate": null,
    "message": "A health check failed for the 'LDAP' connection, please see the MWS
health details page for more information.",
    "objectId": "ldap",
    "objectType": "Health",
    "observedDate": "2013-08-30 18:11:15 UTC",
    "origin": "MWS/HealthNotificationJob",
    "user": "admin",
    "id": "5230ed82e4b065347016d60d"
}
```

Ignoring Notifications

The HTTP PUT method is used to ignore **Notifications**.

Quick reference

```
PUT http://localhost:8080/mws/rest/notifications/ignore?api-version=3
PUT http://localhost:8080/mws/rest/notifications/<id>/ignore?api-version=3
```

Ignore All Notifications

URLs and parameters

```
PUT http://localhost:8080/mws/rest/notifications/ignore?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-----------------------------------

{"messages":["Updated 10 Notification objects"]}
```

Ignore Single Notification

URLs and parameters

```
PUT http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d/ignore?api-
version=3
```

Parameter	Required	Type	Description
id	Yes	String	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
PUT http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d/ignore?api-
version=3
-----------------------------------

{
    "conditionId": "521a1f18e4b0e3f9031f47f5",
    "createdDate": "2013-08-25 15:13:28 UTC",
    "details": {},
    "dismissedDate": null,
    "ignoredDate": "2013-09-17 15:34:36 UTC",
    "message": "A health check failed for the 'LDAP' connection, please see the MWS
health details page for more information.",
    "objectId": "ldap",
    "objectType": "Health",
    "observedDate": "2013-08-30 18:11:15 UTC",
    "origin": "MWS/HealthNotificationJob",
    "user": "admin",
    "id": "5230ed82e4b065347016d60d"
}
```

Unignoring Notifications

The HTTP PUT method is used to unignore **Notifications**.

Quick reference

```
PUT http://localhost:8080/mws/rest/notifications/unignore?api-version=3
PUT http://localhost:8080/mws/rest/notifications/<id>/unignore?api-version=3
```

Unignore All Notifications

URLs and parameters

```
PUT http://localhost:8080/mws/rest/notifications/unignore?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-----------------------------------

{"messages":["Updated 10 Notification objects"]}
```

Unignore Single Notification

```
PUT
http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d/unignore?api-
version=3
```

Parameter	Required	Type	Description
id	Yes	String	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
PUT
http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d/unignore?api-
version=3
-----------------------------------

{
    "conditionId": "521a1f18e4b0e3f9031f47f5",
    "createdDate": "2013-08-25 15:13:28 UTC",
    "details": {},
    "dismissedDate": "null",
    "ignoredDate": null,
    "message": "A health check failed for the 'LDAP' connection, please see the MWS
health details page for more information.",
    "objectId": "ldap",
    "objectType": "Health",
    "observedDate": "2013-08-30 18:11:15 UTC",
    "origin": "MWS/HealthNotificationJob",
    "user": "admin",
    "id": "5230ed82e4b065347016d60d"
}
```

Dismissing Notifications

The HTTP PUT method is used to dismiss **Notifications**.

Quick reference

```
PUT http://localhost:8080/mws/rest/notifications/dismiss?api-version=3
PUT http://localhost:8080/mws/rest/notifications/<id>/dismiss?api-version=3
```

Dismiss All Notifications

URLs and parameters

```
PUT http://localhost:8080/mws/rest/notifications/dismiss?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-----------------------------------

{"messages":["Updated 10 Notification objects"]}
```

Dismiss Single Notification

URLs and parameters

```
PUT http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d/dismiss?api-
version=3
```

Parameter	Required	Type	Description
id	Yes	String	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
PUT http://localhost:8080/mws/rest/notifications/5230ed82e4b065347016d60d/dismiss?api-
version=3
-----------------------------------

{
    "conditionId": "521a1f18e4b0e3f9031f47f5",
    "createdDate": "2013-08-25 15:13:28 UTC",
    "details": {},
    "dismissedDate": "2013-09-17 15:34:36 UTC",
    "ignoredDate": null,
    "message": "A health check failed for the 'LDAP' connection, please see the MWS
health details page for more information.",
    "objectId": "ldap",
    "objectType": "Health",
    "observedDate": "2013-08-30 18:11:15 UTC",
    "origin": "MWS/HealthNotificationJob",
    "user": "admin",
    "id": "5230ed82e4b065347016d60d"
}
```

Related Topics

- Resources Introduction
- Events
- Fields: Events
- Notifications
- Fields: Notifications
- Creating Events and Notifications
- Plugin Event Service
- Handling Events

- System Events
- Securing the Connection with the Message Queue

Permissions

This section describes behavior of the **Permissions** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: User's Permissions** reference section contains the type and description of fields that all **Permissions** have in common.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/permissions	Get All Permissions	--	Create Single Permission	--
/rest/permissions/<id>	Get Single Permission	--	--	Delete Single Permission
/rest/permissions/users/<id>	Get a User's Permissions	--	--	--
/rest/permissions/users	Get a Current User's Permissions	--	--	--

This topic contains these sections:

- Getting Permissions
  - Get All Permissions
  - Get Single Permission
  - Get a User's Permissions
  - Get a Current User's Permissions
- Creating Permissions
  - Create Single Permission
- Deleting Permissions
  - Delete Single Permission

Getting Permissions

The HTTP GET method is used to retrieve **Permission** information. You can query all objects or a single object.

Quick reference

```
GET http://localhost:8080/mws/rest/permissions?api-version=3
GET http://localhost:8080/mws/rest/permissions/<id>?api-version=3
```

Get All Permissions

URLs and parameters

```
GET http://localhost:8080/mws/rest/permissions?api-version=3[&query={"field":"value"}
&sort={"field":<1|-1>}]
```

Parameter	Required	Type	Description	Example
query	No	JSON	Queries for specific results. It is possible to query permissions by one or more fields based on MongoDB query syntax.	`query={"type":"CUSTOM"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/permissions?api-
version=3&fields=resource,action,description
-----------------------------------

{
       "totalCount": 1,
       "resultCount": 1,
       "results": [{
               "resource" : "chart",
               "action" : "read",
               "description" : "The permission to view all charts."
               } ]
}
```

Sorting and querying

See the sorting and querying sections of Global URL Parameters.

Get Single Permission

URLs and parameters

```
GET http://localhost:8080/mws/rest/permissions/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the permission.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/permissions/<id>?api-version=3
-----------------------------------

{
        "action" : "create",
        "administrator": null,
        "description" : "The permission to create all charts.",
        "id" : "50296335e4b0011b0f8394ec",
        "label" : "Create Chart",
        "resource" : "chart",
        "resourceFilter" : null,
        "type" : "custom",
        "scope" : "NONE",
        "version" : 0
}
```

For permissions with type `"domain"`, scope must be `GLOBAL` or `TENANT`. All other permissions should have scope `NONE`.

Get a User's Permissions

URLs and parameters

```
GET http://localhost:8080/mws/rest/permissions/users/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the user.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/permissions/users/bob?api-version=3
-----------------------------------

[
         {
         "action": "read",
         "administrator": null,
         "description": "The permission to read all charts",
         "id": "5033b842e4b09cc61bedb818",
         "label": "",
         "resource": "chart",
         "resourceFilter": null,
         "type": "custom",
         "scope": "NONE",
         "version": 1
       },
         {
         "action": "read",
         "administrator": null,
         "description": "The permission to read all pages",
         "id": "5033b8a5e4b09cc61bedb82d",
         "label": "",
         "resource": "page",
         "resourceFilter": null,
         "type": "custom",
         "scope": "NONE",
         "version": 1
       },
         {
         "action": "update",
         "administrator": null,
         "description": "The permission to update all pages",
         "id": "5033b8a5e4b09cc61bedb82f",
         "label": "",
         "resource": "page",
         "resourceFilter": null,
         "type": "custom",
         "scope": "NONE",
         "version": 1
       }
]
```

Get a Current User's Permissions

URLs and parameters

```
GET http://localhost/mws/rest/permissions/users/?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost/mws/rest/permissions/users/?api-version=3
-------------------------------------

[
          {
          "action": "read",
          "administrator": null,
          "description": "The permission to read all charts",
          "id": "5033b842e4b09cc61bedb818",
          "label": "",
          "resource": "chart",
          "resourceFilter": null,
          "type": "custom",
          "scope": "NONE",
          "version": 1
        },
          {
          "action": "read",
          "administrator": null,
          "description": "The permission to read all pages",
          "id": "5033b8a5e4b09cc61bedb82d",
          "label": "",
          "resource": "page",
          "resourceFilter": null,
          "type": "custom",
          "scope": "NONE",
          "version": 1
        },
          {
          "action": "update",
          "administrator": null,
          "description": "The permission to update all pages",
          "id": "5033b8a5e4b09cc61bedb82f",
          "label": "",
          "resource": "page",
          "resourceFilter": null,
          "type": "custom",
          "scope": "NONE",
          "version": 1
        }
]
```

Creating Permissions

The HTTP POST method is used to create **Permissions**.

Quick reference

```
POST http://localhost:8080/mws/rest/permissions?api-version=3
```

Create Single Permission

URLs and parameters

```
POST http://localhost:8080/mws/rest/permissions?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

> The **resource**, **action**, and **type** are required on each permission.
>
> Api permissions are permissions with the type 'api' and are the only permissions enforced by MWS.
>
> Api permissions must map to a valid resource. For example, "services" is valid because there is a resource `/mws/rest/services`.
>
> Api permissions must have `create`, `read`, `update`, or `delete` as the action.

The following is an example request body to create a permission:

```
POST http://localhost:8080/mws/rest/permissions?api-version=3
-----------------------------------

{
                        "resource" : "Chart",
                        "action" : "read",
                        "administrator" : null,
                        "type" : "custom",
                        "scope" : "NONE",
                        "label" : "Read all charts",
                        "description" : "The permissions to view all charts."
}
```

Sample response

If the request was successful, the response body is the new permission that was created exactly as shown in **Get Single Permission**. On failure, the response is an error message.

Deleting Permissions

The HTTP DELETE method is used to delete **Permissions**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/permissions/<id>?api-version=3
```

Delete Single Permission

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/permission/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the permission.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{}
```

Related Topics

- Fields: User's Permissions
- Resources Introduction

Plugins

This section describes behavior of the **Plugins** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Plugins** reference contains the type and description of all fields in the **Plugin** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/plugins	Get All Plugins	--	Create Plugin	--
/rest/plugins/reporting-jobs/<jobName>?api-version=3	Get All Plugins Reporting Object	--	--	--
/rest/plugins/reporting-nodes/<nodeName>?api-version=3	Get All Plugins Reporting Object	--	--	--
/rest/plugins/reporting-vms/<vmName>?api-version=3	Get All Plugins Reporting Object	--	--	--
/rest/plugins/<id>	Get Single Plugin	Modify Plugin	--	Delete Plugin

Resource	GET	PUT	POST	DELETE
/rest/plugins/<id>/poll	--	--	Trigger Plugin Poll	--
/rest/plugins/<id>/services/<serviceName>	Access a Plugin Web Service	Access a Plugin Web Service	Access a Plugin Web Service	Access a Plugin Web Service

This topic contains these sections:

- Getting Plugins
  - Get All Plugins
  - Get All Plugins Reporting Object
  - Get Single Plugin
- Creating Plugins
  - Create Plugin
- Modifying Plugins
  - Modify Plugin
  - Trigger Plugin Poll
- Deleting Plugins
  - Delete Plugin
- Accessing Plugin Web Services
  - Access a Plugin Web Service

Getting Plugins

The HTTP GET method is used to retrieve **Plugin** information. Queries for all objects, a single object, and query by reported object are available.

Quick reference

```
GET http://localhost:8080/mws/rest/plugins?api-version=3
GET http://localhost:8080/mws/rest/plugins/<id>?api-version=3
GET http://localhost:8080/mws/rest/plugins/reporting-jobs/<jobName>?api-version=3
GET http://localhost:8080/mws/rest/plugins/reporting-nodes/<nodeName>?api-version=3
GET http://localhost:8080/mws/rest/plugins/reporting-vms/<vmName>?api-version=3
```

Get All Plugins

URLs and parameters

```
GET http://localhost:8080/mws/rest/plugins?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/plugins?api-version=3&fields=id
-------------------------------------

{
  "totalCount": 3,
  "resultCount": 3,
  "results":   [
    {"id": "plugin1"},
    {"id": "plugin2"},
    {"id": "plugin3"}
  ]
}
```

Get All Plugins Reporting Object

URLs and parameters

```
GET http://localhost:8080/mws/rest/plugins/reporting-jobs/<jobName>?api-version=3
GET http://localhost:8080/mws/rest/plugins/reporting-nodes/<nodeName>?api-version=3
GET http://localhost:8080/mws/rest/plugins/reporting-vms/<vmName>?api-version=3
```

Parameter	Required	Type	Valid values	Description
jobName	Yes	String	--	The name of the job to query by.
nodeName	Yes	String	--	The name of the node to query by.
vmName	Yes	String	--	The name of the VM to query by.

See Global URL Parameters for available URL parameters.

Sample response

This built-in query returns the same information as **Get All Plugins**, but filters the items to only plugins that are currently reporting the specified job, node, or VM (see Reporting State Data). The list is sorted ascending by the `precedence` field. In other words, the most authoritative plugin for the report is listed first. For more information, see Data Consolidation.

```
GET http://localhost:8080/mws/rest/plugins/reporting-nodes/node1?api-
version=3&fields=id
-------------------------------------

{
  "totalCount": 3,
  "resultCount": 3,
  "results":   [
    {"id": "plugin1"},
    {"id": "plugin2"},
    {"id": "plugin3"}
  ]
}
```

Get Single Plugin

URLs and parameters

```
GET http://localhost:8080/mws/rest/plugins/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{
  "id":"plugin1",
  "pluginType":"Native",
  "pollInterval":30,
  "autoStart":true,
  "config":{
    "getJobs":"exec:///opt/moab/tools/workload.query.pl"
  },
  "state":"STARTED",
  "nextPollDate":"2011-12-02 17:28:52 UTC",
  "lastPollDate":"2011-12-02 17:28:22 UTC"
}
```

Creating Plugins

The HTTP POST method is used to create **Plugins**.

Quick reference

```
POST http://localhost:8080/mws/rest/plugins?api-version=3
```

Create Plugin

URLs and parameters

```
POST http://localhost:8080/mws/rest/plugins?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

When creating a plugin, the `id` and `pluginType` fields are required. The request body below shows all fields that are available when creating a plugin, along with some sample values.

```
JSON request body
-------------------------------------

{
  "id":"plugin1",
  "pluginType":"Native",
  "pollInterval":30,
  "autoStart":true,
  "config":{
    "getJobs":"exec:///opt/moab/tools/workload.query.pl"
  }
}
```

Sample response

```
JSON response for successful POST
-------------------------------------

{"id": "plugin1"}
```

Restrictions

While it is possible to create a plugin with arbitrary nested configuration, such as:

```
…
"config":{
  "nestedObject":{
    "property1":"value1",
    "property2":"value2"
  },
  "nestedList:["listItem1", "listItem2"]
}
```

It is *not* recommended, because the user interface (see Plugin Management) does not support editing or viewing any configuration data values other than strings.

Modifying Plugins

The HTTP PUT method is used to modify **Plugins**. Additionally, the POST method may be used to trigger an immediate poll of a **Plugin**.

Quick reference

```
PUT http://localhost:8080/mws/rest/plugins/<id>?api-version=3
POST http://localhost:8080/mws/rest/plugins/<id>/poll?api-version=3
```

Modify Plugin

URLs and parameters

```
PUT http://localhost:8080/mws/rest/plugins/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Request body

The request body below shows all the fields that are available when modifying a **Plugin**, along with some sample values.

```
JSON request body for plugin modification
-----------------------------------

{
  "state":"STARTED",
  "pollInterval":30,
  "autoStart":true,
  "config":{
    "getJobs":"exec:///opt/moab/tools/workload.query.pl"
  },
  "state":"STARTED"
}
```

Sample response

```
JSON response
-----------------------------------

{"messages":["Plugin plugin1 updated", "Started Plugin 'plugin1'"]}
```

Trigger Plugin Poll

URLs and parameters

```
POST http://localhost:8080/mws/rest/plugins/<id>/poll?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Trigger poll

This resource call will trigger an immediate poll of the specified plugin. It is equivalent to the same operation on Monitoring and Lifecycle Controls.

Request body

No request body is required.

Sample response

```
JSON response
-------------------------------------

{"messages":["Polled Plugin with ID 'myPlugin'"]}
```

Deleting Plugins

The HTTP DELETE method is used to delete **Plugins**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/plugins/<id>?api-version=3
```

Delete Plugin

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/plugins/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response for successful DELETE
-------------------------------------

{}
```

Additional information about a successful DELETE can be found in the HTTP response header `X-MWS-Message`.

```
JSON response for an unsuccessful DELETE
-----------------------------------

{"messages":["Plugin plugin1 could not be deleted", "Error message describing the
problem"]}
```

Accessing Plugin Web Services

All HTTP methods can be used to access **Plugin Web Services**. However, some services only support specific methods. Check the specific plugin type documentation for more information.

Quick reference

```
GET http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
POST http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
PUT http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
DELETE http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
```

Access a Plugin Web Service

URLs and parameters

```
GET http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
POST http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
PUT http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
DELETE http://localhost:8080/mws/rest/plugins/<id>/services/<serviceName>
[/<objectId>]?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.
objectId	No	String	--	An arbitrary ID parameter that will be passed to the web service.
serviceName	Yes	String	--	The name of the web service, either in CamelCase or hyphenated.

See Global URL Parameters for available URL parameters.

Web service IDs

Translation is done to map CamelCase web service names to hyphenated names in the URL. For example, a web service method named `notifyEvent` on a plugin with a name of `notifications` may be called with the following URLs:

```
// CamelCase
/rest/plugins/notifications/services/notifyEvent

// Hyphenated
/rest/plugins/notifications/services/notify-event
```

HTTP method and request body

Because plugin custom web services do not need to distinguish which HTTP method is used (see Custom Web Services), it is recommended to use GET and POST when making requests to access web services unless documented otherwise. The request body and output may vary for each web service called. See Plugin Types for the requested plugin for available web services, request parameters, and expected output.

Related Topics

- Fields: Plugins
- Resources Introduction
- Plugin Types

Plugin Types

This section describes behavior of the **Plugin Type** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

> The **Fields: Plugin Types** reference section contains the type and description of all fields in the **Plugin Type** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/plugin-types	Get All Plugin Types	Creating or Updating Plugin Types	--	--
/rest/plugin-types/<id>	Get Single Plugin Type	--	--	--

This topic contains these sections:

- Getting Plugin Types
  - Get All Plugin Types
  - Get Single Plugin Type
- Creating or Updating Plugin Types
  - Update Plugin Type (File)
  - Update Plugin Type (JAR)

Getting Plugin Types

The HTTP GET method is used to retrieve **Plugin Type** information. Queries for all objects and a single object are available.

Quick reference

```
GET http://localhost:8080/mws/rest/plugin-types/<id>?api-version=3
```

Get All Plugin Types

URLs and parameters

```
GET http://localhost:8080/mws/rest/plugin-types?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/plugin-types?api-version=3&fields=id
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
    {"id": "vCenter"},
    {"id": "Native"}
  ]
}
```

Get Single Plugin Type

URLs and parameters

```
GET http://localhost:8080/mws/rest/plugin-types/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{
  "author": "Adaptive Computing Enterprises, Inc.",
  "commonsVersion": "0.9.3 > *",
  "description": "Polls a VMware® vCenter™ Server for information on the hypervisors
and virtual machines it manages.",
  "documentationLink": "",
  "email": "",
  "eventComponent": 1,
  "realizedEventComponent": 513,
  "id": "VCenter",
  "initialPlugins": { },
  "instances": [
    {"id":"vcenter"}
  ],
  "issueManagementLink": "",
  "license": "APACHE",
  "mwsVersion": "7.1.2 > *",
  "pollMethod": true,
  "scmLink": "",
  "title": "VCenter",
  "version": "1.0",
  "webServices": [ ],
  "website": "http://www.adaptivecomputing.com"
}
```

Creating or Updating Plugin Types

The HTTP PUT method is used to create or update **Plugin Types**. The `Content-Type` HTTP header is used to determine if the request contains a single class file as plaintext or the binary data of a JAR file. Each request is explained in the following sections.

Quick reference

```
PUT http://localhost:8080/mws/rest/plugin-types?api-version=3[&reload-plugins=false]
```

There is a known issue with dynamically updating plugin types with typed field injection. For more information, see Add or Update Plugin Types.

Update Plugin Type (File)

URLs and parameters

```
PUT http://localhost:8080/mws/rest/plugin-types?api-version=3[&reload-plugins=false]
```

Parameter	Required	Type	Valid values	Description
reload-plugins	No	String	`true` or `false`	Reloads all plugins of this type on successful update. Defaults to `true`.

See Global URL Parameters for available URL parameters.

Request body

This function is idempotent, meaning it will create the **Plugin Type** if it does not exist or update it if it does. The request body is the actual contents of the class file to upload. This web service is an exception to most as it *requires* a content type of `application/x-groovy` or `text/plain`.

If the `application/x-groovy` or `text/plain` content types are not used in the request, it will be interpreted as JSON, resulting in a failure.

```
Plaintext upload
-----------------------------------

package test

import com.adaptc.mws.plugins.*

class UploadPlugin {
       static author = "Adaptive Computing"
       static description = "A sample plugin class"
       String id

       public void configure() throws InvalidPluginConfigurationException {
               def myConfig = config
               def errors = []
               if (!myConfig.arbitraryKey)
                       errors << "Missing arbitraryKey!"
               if (errors)
                       throw new InvalidPluginConfigurationException(errors)
       }

       public def customService(Map params) {
               return params
       }
}
```

If using the curl library to perform plugin type uploading, the equivalent of the command-line option `--data-binary` must be used to send the request body. Otherwise compilation errors may be encountered when uploading the plugin type.

Sample response

The response of this task is the same as the **Get All Plugin Types** task. The reason that the return of this task is a list is to accommodate the possibility of uploading multiple plugin types in a single JAR file as explained in the next section.

Update Plugin Type (JAR)

URLs and parameters

```
PUT http://localhost:8080/mws/rest/plugin-types?api-version=3&jar-
filename=<filename.jar>[&reload-plugins=false]
```

Parameter	Required	Type	Valid values	Description
jar-filename	Yes	String	--	The filename of the JAR file that is being uploaded.
reload-plugins	No	String	`true` or `false`	Reloads all plugins of this type on successful update. Defaults to `true`.

See Global URL Parameters for available URL parameters.

Request body

This function is idempotent, meaning it will create the **Plugin Types** if they do not exist or update them if they do. The request body is the binary contents of the JAR file to upload. This web service is an exception to most as it *requires* a content type of `application/x-jar`.

If the `application/x-jar` content type is not used in the request, it will be interpreted as JSON, resulting in a failure.

If using the curl library to perform plugin type uploading, the equivalent of the command-line option `--data-binary` must be used to send the request body. Otherwise compilation errors may be encountered when uploading the plugin type.

Sample response

The response of this task is the same as the **Get All Plugin Types** task. Note that when using a JAR file, multiple plugin types may be uploaded in the same request.

Related Topics

- Fields: Plugin Types
- Resources Introduction
- Plugins

Policies

This section describes behavior of the **Policies** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

> The **Fields: Policies** reference section contains the type and description of fields of all **Policies**.

Supported policies

Name	ID
Automatic VM Migration	auto-vm-migration
Fairshare	fairshare
Hypervisor Allocation Overcommit	hv-allocation-overcommit
Migration Exclusion List	migration-exclusion-list
Node Allocation	node-allocation

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/policies	**Get All Policies**	--	--	--
/rest/policies/<id>	**Get Single Policy**	**Modify Policy**	--	--

This topic contains these sections:

- Getting Policies
  - Get All Policies
  - Get Single Policy
- Modifying Policies
  - Modify Policy

Getting Policies

The HTTP GET method is used to retrieve **Policies** information.

Quick reference

```
GET http://localhost:8080/mws/rest/policies?api-version=3
```

Get All Policies

URLs and parameters

```
GET http://localhost:8080/mws/rest/policies?api-version=3
```

Parameter	Required	Type	Description	Example
query	No	JSON	Query for specific results.	`query={"state":"DISABLED","conflicted":"false"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"id":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/policies?api-version=3&fields=id,state,conflicted
-----------------------------------

{
  "totalCount": 4,
  "resultCount": 4,
  "results": [  {
    "conflicted": false,
    "state": "DISABLED",
    "id": "auto-vm-migration"
  },{
    "conflicted": false,
    "state": "DISABLED",
    "id": "hv-allocation-overcommit"
  },{
    "conflicted": false,
    "state": "DISABLED",
    "id": "node-allocation"
  },{
    "conflicted": false,
    "state": "DISABLED",
    "id": "migration-exclusion-list"
  }]
}
```

Get Single Policy

URLs and parameters

```
GET http://localhost:8080/mws/rest/policies/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample responses

```
Auto VM Migration
-------------------------------------

{
  "conflicted": false,
  "description": "Controls how virtual machines are automatically migrated.",
  "id": "auto-vm-migration",
  "name": "Auto VM Migration",
  "potentialConflicts": [],
  "priority": 1,
  "state": "DISABLED",
  "tags": [],
  "types": [],
  "version": 0,
  "genericMetricThresholds":{
       "GMETRIC1":1.3
  },
  "processorUtilizationThreshold":0.5,
  "memoryUtilizationThreshold":0.4
}
```

```
Fairshare
-----------------------------------

{
    "conflicted": false,
    "decayFactor": 0.44,
    "depth": 4,
    "description": "Control job feasibility and priority decisions based on system
utilization targets for users, groups, accounts, classes, and QoS levels.",
    "intervalSeconds": 600,
    "name": "Fairshare",
    "potentialConflicts": [],
    "priority": 16,
    "state": "ENABLED",
    "tags": [],
    "types": [],
    "usageMetric": "DEDICATED_PROCESSOR_SECONDS_DELIVERED",
    "version": 3,
    "id": "fairshare"
}
```

```
Hypervisor Allocation Overcommit
-----------------------------------

{
  "conflicted": false,
  "description": "Controls how hypervisors are overallocated with regards to
processors and memory.",
  "id": "hv-allocation-overcommit",
  "name": "Hypervisor Allocation Overcommit",
  "potentialConflicts": [],
  "priority": 2,
  "state": "DISABLED",
  "tags": [],
  "types": [],
  "version": 0,
  "processorAllocationLimit":29.5,
  "memoryAllocationLimit":1.2
}
```

```
Migration Exclusion List
-----------------------------------

{
    "conflicted": false,
    "description": "Controls which machines are excluded from automatic live migration
operations.",
    "hvExclusionList": ["blade05", "blade02"],
    "name": "Migration Exclusion List",
    "potentialConflicts": [],
    "priority": 100,
    "state": "DISABLED",
    "tags": [],
    "types": [],
    "version": 1,
    "vmExclusionList": ["vm1", "vm5"],
    "id": "migration-exclusion-list"
}
```

```
Node Allocation
-----------------------------------

{
  "conflicted": false,
  "description": "Controls how nodes are selected for workload placement.",
  "id": "node-allocation",
  "name": "Node Allocation",
  "potentialConflicts": [],
  "priority": 3,
  "state": "DISABLED",
  "tags": [],
  "types": [],
  "version": 0,
  "nodeAllocationAlgorithm": "CustomPriority",
  "customPriorityFunction": "-100*GMETRIC[vmcount]"
}
```

Modifying Policies

The HTTP PUT method is used to modify **Policies**.

Quick reference

```
PUT http://localhost:8080/mws/rest/policies/<id>?api-version=3
```

Modify Policy

URLs and parameters

```
PUT http://localhost:8080/mws/rest/policies/<id>?api-version=3[&change-mode=set]
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Additional URL parameters

URL parameters for modifying a Migration Exclusion Lists Policy.

Migration Exclusion Lists parameter	Required	Type	Valid values	Description
change-mode	No	String	set (default) add remove	If **set**, replace existing exclusion list(s) with the given one. If **add**, add the given VMs/HVs to the existing exclusion list(s). If **remove**, remove the given VMs/HVs from the existing exclusion list(s).

Request body

In general, the fields shown in the **Fields: Policies** reference section are *not* available for modification. However, the `state` field may be modified to a valid `PolicyState`. All other fields listed in the specific policy type sections (i.e. `AutoVMMigrationPolicy`) may be modified unless documented otherwise.

- The request body below shows all the fields that are available when modifying a Auto VM Migration Policy, along with some sample values.

```
JSON request body for Auto VM Migration Policy
----------------------------------


{
    "genericMetricThresholds": {
        "GENERICTHRESHOLD": 5
    },
    "memoryUtilizationThreshold": 0.5,
    "processorUtilizationThreshold": 0.4
}
```

- The request body below shows all the fields that are available when modifying a Fairshare Policy, along with some sample values.

```
JSON request body for Fairshare Policy
-----------------------------------

{
    "decayFactor": 0.44,
    "depth": 4,
    "intervalSeconds": 600,
    "usageMetric": "DEDICATED_PROCESSOR_SECONDS_DELIVERED",
}
```

- The request body below shows all the fields that are available when modifying a Hypervisor Allocation Overcommit Policy, along with some sample values.

```
JSON request body for Hypervisor Allocation Overcommit Policy
-----------------------------------

{
    "processorAllocationLimit":29.5,
    "memoryAllocationLimit":1.2
}
```

- The request body below shows all the fields that are available when modifying a Migration Exclusion Lists Policy, along with some sample values.

```
JSON request body for Migration Exclusion Lists Policy
-----------------------------------

{
   "vmExclusionList" : ["vm1","vm3","vm5"],
   "hvExclusionList" : ["hv2","hv3","hv6"]
}
```

- The request body below shows all the fields that are available when modifying a Node Allocation Policy, along with some sample values.

```
JSON request body for Node Allocation Policy
-----------------------------------

{
   "nodeAllocationAlgorithm" : "CustomPriority",
   "customPriorityFunction" : "-100*GMETRIC[vmcount]"
}
```

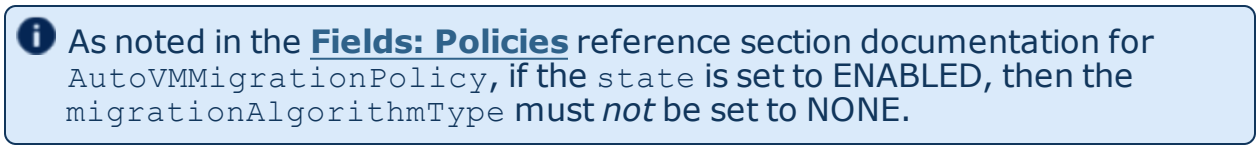
Sample response

```
JSON response
-----------------------------------

{
    "messages": ["Policy auto-vm-migration updated"]
}
```

Samples

Enable the Auto VM Migration Policy and set values.

```
PUT http://localhost:8080/mws/rest/policies/auto-vm-migration?api-version=3
-------------------------------------

{
  "state": "enabled",
  "migrationAlgorithmType": "overcommit",
  "processorUtilizationThreshold": 0.5,
  "memoryUtilizationThreshold": 0.4
}
```



Restrictions

All policies:

- Fields cannot be modified while the policy is disabled. Enable the policy to modify the field.

Auto VM Migration

- Arbitrary metrics can be added to **genericMetricThresholds**, but they cannot be removed once added.
- The **migrationAlgorithmType** field cannot be modified while the policy is disabled. Enable the policy to modify the field.
- Moab is configured with a default limit of 10 generic metrics. If this limit is reached, such as when arbitrary metrics are added to **genericMetricThresholds**, the metric will not be reported. To increase this limit, set the `MAXGMETRIC` property in the Moab configuration file.

Fairshare

- Updating the **usageMetric** field will clear all credential-based fairshare interval data.

Related Topics

- Fields: Policies
- Fairshare
- Resources Introduction

Fairshare

This section describes behavior of the **Fairshare** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from

MWS.

> The supported methods table below requires each resource to be accessed with a URL parameter of `api-version=3`.
>
> For more information, see Requesting Specific API Versions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/policies/fairshare	Get All Fairshare Interval Data	--	--	--
/rest/policies/fairshare/<credentialType>	Get all Fairshare Interval Data for a Single Credential Type	--	--	--
/rest/policies/fairshare/<credentialType>/<name>	Get all Fairshare Interval Data for a Single Credential	--	--	--

This topic contains these sections:

- Getting Credential-Based Fairshare Interval Data
  - Get All Fairshare Interval Data
  - Get all Fairshare Interval Data for a Single Credential Type
  - Get all Fairshare Interval Data for a Single Credential

Getting Credential-Based Fairshare Interval Data

The HTTP GET method is used to retrieve **Policies** information.

Quick reference

```
GET http://localhost:8080/mws/rest/policies/fairshare/credentials?api-version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/accounts?api-
version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/classes?api-
version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/groups?api-version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/qoses?api-version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/users?api-version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/accounts/<name>?api-
version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/classes/<name>?api-
version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/groups/<name>?api-
version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/qoses/<name>?api-
version=3
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/users/<name>?api-
version=3
```

Get All Fairshare Interval Data

URLs and parameters

```
GET http://localhost:8080/mws/rest/policies/fairshare/credentials?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/policies/fairshare/credentials?api-version=3
-----------------------------------

{
  "totalCount": 4,
  "resultCount": 4,
  "results":   [
    {
      "name": "jbethune",
      "target_type": null,
      "target": null,
      "interval_data":       [
        0,
        0,
        0,
        0
      ],
      "credential_type": "USER"
    },
    {
      "name": "jfoote",
      "target_type": null,
      "target": null,
      "interval_data":       [
        2104.16,
        2377.06,
        2240.1,
        2550
      ],
      "credential_type": "GROUP"
    },
    {
      "name": "NOGROUP",
      "target_type": null,
      "target": null,
      "interval_data":       [
        0,
        0,
        0,
        0
      ],
      "credential_type": "GROUP"
    },
    {
      "name": "DEFAULT",
      "target_type": null,
      "target": null,
      "interval_data":       [
        0,
        0,
        0,
        0
      ],
      "credential_type": "ACCOUNT"
    },
    {
      "name": "Administration",
      "target_type": null,
      "target": null,
      "interval_data":       [
        5256.28,
        6247.05,
        6048.27,
        6948.67
      ],
      "credential_type": "ACCOUNT"
    }
```

```
   ]
}
```

Get all Fairshare Interval Data for a Single Credential Type

URLs and parameters

```
GET
http://localhost:8080/mws/rest/policies/fairshare/credentials/<accounts|classes|groups
|qoses|users>?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample responses

```
GET http://localhost:8080/mws/rest/policies/fairshare/credentials/accounts?api-
version=3
-----------------------------------
{
  "totalCount": 6,
  "resultCount": 6,
  "results":   [
    {
      "name": "jbethune",
      "target_type": null,
      "target": null,
      "interval_data":       [
        0,
        0,
        0,
        0
      ],
      "credential_type": "ACCOUNT"
    },
    {
      "name": "Administration",
      "target_type": null,
      "target": null,
      "interval_data":       [
        5256.28,
        6247.05,
        6048.27,
        6948.67
      ],
      "credential_type": "ACCOUNT"
    },
    {
      "name": "Shared",
      "target_type": null,
      "target": null,
      "interval_data":       [
        4261.38,
        4951.09,
        4480.2,
        5000.54
      ],
      "credential_type": "ACCOUNT"
    },
    {
      "name": "Engineering",
      "target_type": null,
      "target": null,
      "interval_data":       [
        15034.64,
        17245.93,
        15008.67,
        17085
      ],
      "credential_type": "ACCOUNT"
    },
    {
      "name": "Test",
      "target_type": null,
      "target": null,
      "interval_data":       [
        1808.08,
        1873.96,
        1568.07,
        1757.33
      ],
      "credential_type": "ACCOUNT"
    },
```

```
    {
      "name": "Research",
      "target_type": null,
      "target": null,
      "interval_data":       [
        47606.8,
        52861.83,
        46370.07,
        52785
      ],
      "credential_type": "ACCOUNT"
    }
  ]
}
```

Get all Fairshare Interval Data for a Single Credential

URLs and parameters

```
GET
http://localhost:8080/mws/rest/policies/fairshare/credentials/<accounts|classes|groups
|qoses|users>/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The unique name of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
GET
http://localhost:8080/mws/rest/policies/fairshare/credentials/accounts/DEFAULT?api-
version=3
-----------------------------------
{
  "name": "DEFAULT",
  "target_type": null,
  "target": null,
  "interval_data":   [
    0,
    0,
    0,
    0
  ],
  "credential_type": "ACCOUNT"
}
```

Related Topics

- Policies
- Resources Introduction

Principals

This section describes behavior of the **Principal** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Principals** reference contains the type and description of all fields in the **Principal** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/principals	Get All Principals	--	Create Single Principal	--
/rest/principals/<id>	Get Single Principal	Modify Single Principal	--	Delete Single Principal
/rest/principals/<name>	Get Single Principal	Modify Single Principal	--	Delete Single Principal

This topic contains these sections:

- Getting Principals
  - Get All Principals
  - Get Single Principal
- Creating Principals
  - Create Single Principal
- Modifying Principals
  - Modify Single Principal
- Deleting Principals
  - Delete Single Principal

Getting Principals

The HTTP GET method is used to retrieve **Principal** information. You can query all objects or a single object.

Quick reference

```
GET http://localhost:8080/mws/rest/principals?api-version=3[&query={"field":"value"}
&sort={"field":<1|-1>}]
GET http://localhost:8080/mws/rest/principals/<id>?api-version=3
GET http://localhost:8080/mws/rest/principals/<name>?api-version=3
```

Get All Principals

URLs and parameters

```
GET http://localhost:8080/mws/rest/principals?api-version=3[&query={"field":"value"}
&sort={"field":<1|-1>}]
```

Parameter	Required	Type	Description	Example
query	No	JSON	Queries for specific results. It is possible to query principals by one or more fields based on MongoDB query syntax.	`query={"name":"Acme Principal"}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/principals?api-version=3&fields=name,group
-----------------------------------

{
  "totalCount": 2,
  "resultCount": 2,
  "results":    [
    {
      "groups": [      {
        "name": "CN=Engineering,CN=Users,DC=corp,DC=cloud,DC=dev",
        "type": "LDAPGROUP"
      }],
         "name": "Engineering-Principal"
    },
       {
         "groups": [   {
        "name": "CN=Marketing,CN=Users,DC=corp,DC=cloud,DC=dev",
        "type": "LDAPGROUP"
      }],
         "name": "Marketing-Principal"
    }
  ]
}
```

Sorting and Querying

See the sorting and querying sections of Global URL Parameters.

Get Single Principal

URLs and parameters

```
GET http://localhost:8080/mws/rest/principals/<id>?api-version=3
GET http://localhost:8080/mws/rest/principals/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the principal.
name	Yes	String	--	The name of the principal.

You must specify either id or name, but you do not have to specify both.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/principals/principal8?api-version=3
-------------------------------------

{
         "attachedRoles": [  {
               "description": "This is a role for normal users in the Acme BU Group.",
               "id": "5033b8eae4b09cc61bedb895",
               "name": "Acme-User-Role",
               "permissions":   [
                                 {
                        "action": "read",
                        "administrator": null,
                        "description": "The permission to read all nodes",
                        "id": "5033b842e4b09cc61bedb818",
                        "label": "",
                        "resource": "nodes",
                        "resourceFilter": null,
                        "type": "api",
                        "version": 1
                  },
               ],
               "version": 2
         }],
         "description": "Principal 8",
         "groups": [  {
               "name": "CN=Engineering,CN=Users,DC=corp,DC=cloud,DC=dev",
               "type": "LDAPGROUP"
         }],
         "id": "5033d33fe4b018b28745fecd",
         "name": "principal8",
         "users":   [
               {
                 "name": "jhammon",
                 "type": "LDAP"
               },
               {
                 "name": "bjones",
                 "type": "LDAP"
               }
         ],
        "version": 0
}
```

Creating Principals

The HTTP POST method is used to submit **Principals**.

Quick reference

```
POST http://localhost:8080/mws/rest/principals?api-version=3
```

Create Single Principal

URLs and parameters

```
POST http://localhost:8080/mws/rest/principals?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

> The **name** field is required and must contain only letters, digits, periods, dashes, and underscores.
>
> The **attachedRoles** field expects an array of Role IDs *or* names:

The following is an example request body to create a principal:

```
POST http://localhost:8080/mws/rest/principals?api-version=3
----------------------------------

{
       "name" : "Acme-Principal",
       "attachedRoles" : [{"name":"Acme-User-Role"}],
       "description" : "A cool principal",
       "groups" : [{"name": "CN=Engineering,CN=Users,DC=corp,DC=cloud,DC=dev",
"type":"LDAPGROUP"}],
       "users" : [{
               "name" : "john",
               "type" : "LDAP"
       } ]
}
```

Sample response

If the request was successful, the response body is the new principal that was created, exactly as shown in **Get Single Principal**. On failure, the response is an error message.

Modifying Principals

The HTTP PUT method is used to modify **Principals**.

Quick reference

```
PUT http://localhost:8080/mws/rest/principals/<id>?api-version=3
PUT http://localhost:8080/mws/rest/principals/<name>?api-version=3
```

Modify Single Principal

URLs and parameters

```
PUT http://localhost:8080/mws/rest/principals/<id>?api-version=3
PUT http://localhost:8080/mws/rest/principals/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Principal.
name	Yes	String	--	The name of the Principal. The **name** field must contain only letters, digits, periods, dashes, and underscores.
change-mode	Yes	String	add remove set (default)	If **add**, add the given objects (ldapGroups, ldapOUs, etc.) to the objects that already exist. If **remove**, delete the given objects from the objects that already exist. If **set**, add the given objects (ldapGroups, ldapOUs, etc.) and remove the objects that already exist.

See Global URL Parameters for available URL parameters.

You must specify either **id** or **name**, but you do not have to specify both. The **attachedRoles** field expects an array of Role IDs *or* names:

Example request

```
PUT http://locahost/mws/rest/principals/Acme-Principal?api-version=3
-----------------------------------
{
        "groups" : [ {
                "name" : "CN=Marketing,CN=Users,DC=mycompany,DC=com",
                "type" : "LDAPGROUP"
        },{
                "name" : "CN=Sales,CN=Users,DC=mycompany,DC=com",
                "type" : "LDAPGROUP"
        }],
        "users" : [ {
                "name" : "jhammon",
                "type" : "LDAP"
        }]
}
```

The **version** field contains the current version of the database entry. This field cannot be updated directly. However, if **version** is included in the modify request, it will be used to verify that another client did not update the object between the time that the data was retrieved and the modify request was delivered.

Sample response

If the request was successful, the response body is the modified principal as shown in **Get Single Principal**. On failure, the response is an error message.

Deleting Principals

The HTTP DELETE method is used to delete **Principals**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/principals/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/principals/<name>?api-version=3
```

Delete Single Principal

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/principals/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/principals/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the principal.
name	Yes	String	--	The name of the principal.

See Global URL Parameters for available URL parameters.

You must specify either **id** or **name**, but you do not have to specify both.

Sample response

```
JSON response
-------------------------------------

{}
```

Related Topics

- Fields: Principals
- Resources Introduction

Priority

This section describes behavior of the **priority** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/priority	Get All Priorities	Modify Priorities	--	--

This topic contains these sections:

- Getting Priorities
  - Get All Priorities
- Modifying Priorities
  - Modify Priorities

Getting Priorities

The HTTP GET method is used to retrieve **priority** information.

Quick reference

```
GET http://localhost:8080/mws/rest/priority?api-version=3
```

Get All Priorities

URLs and parameters

```
GET http://localhost:8080/mws/rest/priority?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
-------------------------------------
{
  "service":   {
    "weight": 1,
    "queue_time": 1,
    "x_factor": 0,
    "policy_violation": 0,
    "bypass": 0
  },
  "target":   {
    "weight": 1,
    "queue_time": 0,
    "x_factor": 0
  },
  "credential":   {
    "weight": 1,
    "user_credential": 0,
    "group_credential": 0,
    "account_credential": 0,
    "class_credential": 0,
    "qos_credential": 0
  },
  "attribute":   {
    "weight": 1,
    "attribute": 0,
    "state": 0
  },
  "fairshare":   {
    "weight": 1,
    "user_credential": 1000,
    "group_credential": 0,
    "account_credential": 0,
    "class_credential": 0,
    "qos_credential": 0,
    "jobs_per_user": 0,
    "processor_seconds_per_user": 0,
    "processors_per_user": 0
  },
  "resource":   {
    "weight": 1,
    "node": 0,
    "disk": 0,
    "memory": 0,
    "swap": 0,
    "processor_equivalent_seconds": 0,
    "walltime": 0
  },
  "usage":   {
    "weight": 1,
    "consumed": 0,
    "remaining": 0,
    "percentage_consumed": 0
  }
}
```

Modifying Priorities

The HTTP PUT method is used to update **priority** information.

Quick reference

```
PUT http://localhost:8080/mws/rest/priority?api-version=3
```

Modify Priorities

URLs and parameters

```
PUT http://localhost:8080/mws/rest/priority?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample body

```
PUT http://localhost:8080/mws/rest/priority?api-version=3
-----------------------------------
{
  "service":   {
    "weight": 2,
    "queue_time": 2,
    "x_factor": 1,
    "policy_violation": 1,
    "bypass": 1
  }
}
```

Related Topics

- Resources Introduction

Reports

This section describes behavior of the reporting framework in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Reports**, **Fields: Report Samples**, and **Fields: Report Datapoints** reference sections contain the type and description of all fields in the **Report**, **Sample**, and **Datapoint** objects. They also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/reports	**Get All Reports (No Data)**	--	**Create Report**	**Delete Report**
/rest/reports/<name>	**Get Single Report (With Data)**	--	--	--

Resource	GET	PUT	POST	DELETE
/rest/reports/<id>	Get Single Report (With Data)	--	--	--
/rest/reports/<name>/datapoints	Get Datapoints for Single Report	--	--	--
/rest/reports/<id>/datapoints	Get Datapoints for Single Report	--	--	--
/rest/reports/<name>/samples	Get Samples for Report	--	Create Samples for Report	--
/rest/reports/<id>/samples	Get Samples for Report	--	Create Samples for Report	--

This topic contains these sections:

- Getting Reports
  - Get All Reports (No Data)
  - Get Single Report (With Data)
  - Get Datapoints for Single Report
- Getting Samples for Reports
  - Get Samples for Report
- Creating Reports
  - Create Report
- Creating Samples
  - Create Samples for Report
- Deleting Reports
  - Delete Report

Getting Reports

The HTTP GET method is used to retrieve **Report** information. Queries for all reports with no attached data and a single report with associated data are available.

Quick reference

```
GET http://localhost:8080/mws/rest/reports?api-version=3[&query={"field":"value"}
&sort={"field":<1|-1>}]
GET http://localhost:8080/mws/rest/reports/<id>?api-version=3
GET http://localhost:8080/mws/rest/reports/<name>?api-version=3
```

Get All Reports (No Data)

URLs and parameters

```
GET http://localhost:8080/mws/rest/reports?api-version=3[&query={"field":"value"}
&sort={"field":<1|-1>}]
```

Parameter	Required	Type	Description	Example
query	No	JSON	Queries for specific results. It is possible to query reports by one or more fields based on MongoDB query syntax.	`query={"reportSize":4}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": [  {
    "id": "3efe5c670be86ba8560397ff",
    "name": "cpu-util"
       …
  }]
}
```

Samples

```
GET http://localhost:8080/mws/rest/reports?api-version=3&fields=id,name
-----------------------------------

{
  "totalCount": 3,
  "resultCount": 3,
  "results":   [
        {
      "id": "3efe5c670be86ba8560397ff",
      "name": "cpu-util"
    },
        {
      "id": "3efe5c670be86ba856039800",
      "name": "cpu-temp"
    },
        {
      "id": "3efe5c670be86ba856039801",
      "name": "cpu-load"
    }
  ]
}
```

Get Single Report (With Data)

URLs and parameters

```
GET http://localhost:8080/mws/rest/reports/<id>?api-version=3
GET http://localhost:8080/mws/rest/reports/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the report.
name	Yes	String	--	The name of the report.

Only one of **id** or **name** are required.

See Global URL Parameters for available URL parameters.

Sample response

In the example below, the first datapoint has a `null` data element, which means that the `minimumSampleSize` configured for the report was not met when consolidating the datapoint. The second datapoint contains actual data.

```
JSON response
-------------------------------------

{
  "consolidationFunction": "average",
  "datapointDuration": 15,
  "datapoints":   [
        {
      "endDate": "2011-12-02 17:28:22 UTC",
      "startDate": "2011-12-02 17:28:22 UTC",
      "firstSampleDate": null,
      "lastSampleDate": null,
      "data": null
    },
        {
      "endDate": "2011-12-02 17:28:23 UTC",
      "startDate": "2011-12-02 17:28:37 UTC",
      "firstSampleDate": "2011-12-02 17:28:23 UTC",
      "lastSampleDate": "2011-12-02 17:28:30 UTC",
      "data":       {
        "utilization": 99.89,
        "time": 27.433333333333337
      }
    }
  ],
  "description": "Example of CPU utilization reporting",
  "id": "3efe5c670be86ba8560397ff",
  "keepSamples": false,
  "minimumSampleSize": 1,
  "name": "cpu-util",
  "reportSize": 2
}
```

Get Datapoints for Single Report

URLs and parameters

```
GET http://localhost:8080/mws/rest/reports/<id>/datapoints?api-version=3[&query=
{"field":"value"}&sort={"field":<1|-1>}]
GET http://localhost:8080/mws/rest/reports/<name>/datapoints?api-version=3[&query=
{"field":"value"}&sort={"field":<1|-1>}]
```

Parameter	Required	Type	Description	Example
id	Yes	String	The unique identifier of the report.	--
name	Yes	String	The name of the report.	--
query	No	JSON	Queries for specific results.	`query={"reportS-ize":4}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

Only one of **id** or **name** are required.

It is possible to query reports by one or more fields based on MongoDB query syntax.

See Global URL Parameters for available URL parameters.

Sample response

This function is exactly the same as **Get Single Report (With Data)**. No report metadata (i.e. `description`, `minimumSampleSize`, etc.) is returned.

```
JSON response
-----------------------------------

{
  "resultCount":1,
  "totalCount":1,
  "results":[
        {
      "endDate": "2011-12-02 17:28:22 UTC",
      "startDate": "2011-12-02 17:28:22 UTC",
      "firstSampleDate": null,
      "lastSampleDate": null,
      "data": null
    },
        {
      "endDate": "2011-12-02 17:28:37 UTC",
      "startDate": "2011-12-02 17:28:37 UTC",
      "firstSampleDate": "2011-12-02 17:28:23 UTC",
      "lastSampleDate": "2011-12-02 17:28:23 UTC",
      "data":       {
        "utilization": 99.89,
        "time": 27.433333333333337
      }
    }
  ]
}
```

Getting Samples for Reports

The HTTP GET method is used to retrieve **Sample** information.

Quick reference

```
GET http://localhost:8080/mws/rest/reports/<id>/samples?api-version=3[&query=
{"field":"value"}&sort={"field":<1|-1>}]
GET http://localhost:8080/mws/rest/reports/<name>/samples?api-version=3[&query=
{"field":"value"}&sort={"field":<1|-1>}]
```

Get Samples for Report

URLs and parameters

```
GET http://localhost:8080/mws/rest/reports/<id>/samples?api-version=3[&query=
{"field":"value"}&sort={"field":<1|-1>}]
GET http://localhost:8080/mws/rest/reports/<name>/samples?api-version=3[&query=
{"field":"value"}&sort={"field":<1|-1>}]
```

Parameter	Required	Type	Description	Example
id	Yes	String	The unique identifier of the report.	--
name	Yes	String	The name of the report.	--
query	No	JSON	Queries for specific results.	`query={"reportS-ize":4}`
sort	No	JSON	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

Only one of **id** or **name** are required.

It is possible to query reports by one or more fields based on MongoDB query syntax.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": [  {
       "timestamp": "2011-12-02 17:28:37 UTC"
       "data":{
               "cpu1":2.3,
               "cpu2":1.2,
               "cpu3":0.0,
               "cpu4":12.1
       },
       …
  }]
}
```

Creating Reports

The HTTP POST method is used to create **Reports**. Operations are available to create reports with or without historical datapoints.

Quick reference

```
POST http://localhost:8080/mws/rest/reports?api-version=3
```

Create Report

URLs and parameters

```
POST http://localhost:8080/mws/rest/reports?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

To create a report, several fields are required as documented in Fields: Reports.

The request body below shows all the fields that are available during report creation.

```
JSON request body
-----------------------------------

{
        "name":"cpu-util",
        "description":"An example report on cpu utilization",
        "consolidationFunction":"average",
        "datapointDuration":15,
        "minimumSampleSize":1,
        "reportSize":2,
        "keepSamples":true,
        "reportDocumentSize":1024,
        "datapoints":[
                {
                        "startDate":"2011-12-01 19:16:57 UTC",
                        "endDate":"2011-12-01 19:16:57 UTC",
                        "data":{
                                "time":30,
                                "util":99.98
                        }
                }
        ]
}
```

Sample response

```
{
        "messages":["Report cpu-util created"],
        "id":"3efe5c670be86ba8560397ff",
        "name":"cpu-util"
}
```

Samples

```
POST http://localhost:8080/mws/rest/reports?api-version=3 (Minimal report without
datapoints)
-----------------------------------

{
        "name":"cpu-util",
        "datapointDuration":15,
        "reportSize":2
}
```

Creating Samples

The HTTP POST method is used to create samples for **Reports**.

Quick reference

```
POST http://localhost:8080/mws/rest/reports?api-version=3
```

Create Samples for Report

URLs and parameters

```
POST http://localhost:8080/mws/rest/reports/<id>/samples?api-version=3
POST http://localhost:8080/mws/rest/reports/<name>/samples?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the report.
name	Yes	String	--	The name of the report.

Only one of **id** or **name** are required.

See Global URL Parameters for available URL parameters.

Request body

To create samples for a report, simply send data and an optional timestamp to the URL above. The request body below shows all the fields that are available during sample creation. Note that the `data` field can contain arbitrary JSON.

```
JSON request body
-----------------------------------

{
       "timestamp":"2011-12-01 19:16:57 UTC",
       "agent":"my agent",
       "data":{
              "cpu1":2.3,
              "cpu2":1.2,
              "cpu3":0.0,
              "cpu4":12.1
       }
}
```

Sample response

```
{"messages":["1 sample(s) created for report cpu-util"]}
```

Deleting Reports

The HTTP DELETE method is used to delete **Reports**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/reports/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/reports/<name>?api-version=3
```

Delete Report

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/reports/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/reports/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the report.
name	Yes	String	--	The name of the report.

Only one of **id** or **name** are required.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-----------------------------------

{"messages":["Report cpu-util deleted"]}
```

Related Topics

- Fields: Reports
- Resources Introduction

Reservations

This section describes behavior of the **Reservations** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Reservations** reference contains the type and description of all fields in the **Reservations** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/reservations	Get All Reservations	--	Create Reservation	--
/rest/reservations/<id>	Get Single Reservation	Modify Reservation	--	Release Reservation

This topic contains these sections:

- Getting Reservations
  - Get All Reservations
  - Get Single Reservation
- Creating Reservations
  - Create Reservation
- Modifying Reservations
  - Modify Reservation
- Releasing Reservations
  - Release Reservation

Getting Reservations

The HTTP GET method is used to retrieve **Reservation** information. Queries for all objects and a single object are available.

Quick reference

```
GET http://localhost:8080/mws/rest/reservations/<id>?api-version=3
```

Restrictions

Only admin or user reservations are returned with this call.

Get All Reservations

URLs and parameters

```
GET http://localhost:8080/mws/rest/reservations?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/reservations?api-version=3&fields=id
-----------------------------------


{
  "totalCount": 3,
  "resultCount": 3,
  "results":   [
    {"id": "system.1"},
    {"id": "system.2"},
    {"id": "system.3"}
  ]
}
```

Get Single Reservation

URLs and parameters

```
GET http://localhost:8080/mws/rest/reservations/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{
  "accountingAccount": "",
  "accountingGroup": "",
  "accountingQOS": "",
  "accountingUser": "root",
  "aclRules": [  {
    "affinity": "NEUTRAL",
    "comparator": "LEXIGRAPHIC_EQUAL",
    "type": "RESERVATION_ID",
    "value": "system.43"
  }],
  "allocatedNodeCount": 1,
  "allocatedProcessorCount": 8,
  "allocatedTaskCount": 1,
  "allocatedNodes": [
       {"id":"node001"}
  ],
  "comments": "",
  "creationDate": null,
  "duration": 200000000,
  "endDate": "2018-03-17 16:49:10 UTC",
  "excludeJobs":   [
    "job1",
    "job2"
  ],
  "expireDate": null,
  "flags":   [
    "REQFULL",
    "ISACTIVE",
    "ISCLOSED"
  ],
  "globalId": "",
  "hostListExpression": "",
  "id": "system.43",
  "idPrefix": "",
  "isActive": true,
  "isTracked": false,
  "label": "",
  "maxTasks": 0,
  "messages": [],
  "owner":   {
    "name": "adaptive",
    "type": "USER"
  },
  "partitionId": "switchB",
  "profile": "",
  "requirements":   {
    "architecture": "",
    "featureList":     [
      "feature1",
      "feature2"
    ],
    "featureMode": "",
    "memory": 0,
    "nodeCount": 0,
    "nodeIds": ["node001:1"],
    "os": "",
    "taskCount": 1
  },
```

```
   "reservationGroup": "",
   "resources": {"PROCS": 0},
   "startDate": "2011-11-14 20:15:50 UTC",
   "statistics":   {
     "caps": 0,
     "cips": 2659.52,
     "taps": 0,
     "tips": 0
   },
   "subType": "Other",
   "taskCount": 0,
   "trigger": null,
   "triggerIds": [],
   "uniqueIndex": "",
   "variables": {}
}
```

Creating Reservations

The HTTP POST method is used to create **Reservations**.

Quick reference

```
POST http://localhost:8080/mws/rest/reservations?api-version=3
```

Create Reservation

URLs and parameters

```
POST http://localhost:8080/mws/rest/reservations?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

The request body below shows all the fields that are available when creating a **Reservation**, along with some sample values.

```
JSON request body
-----------------------------------

{
  "accountingAccount": "",
  "accountingGroup": "",
  "accountingQOS": "",
  "accountingUser": "root",
  "aclRules": [  {
    "affinity": "POSITIVE",
    "comparator": "LEXIGRAPHIC_EQUAL",
    "type": "GROUP",
    "value": "staff"
  }],
  "comments": "",
  "duration": 200000000,
  "endDate": "2018-03-17 16:49:10 UTC",
  "excludeJobs":   [
    "job1",
    "job2"
  ],
  "flags":   [
    "SPACEFLEX",
    "ACLOVERLAP",
    "SINGLEUSE"
  ],
  "hostListExpression": "",
  "idPrefix": "",
  "label": "myreservation",
  "owner":   {
    "name": "adaptive",
    "type": "USER"
  },
  "partitionId": "",
  "profile": "",
  "requirements":   {
    "architecture": "",
    "featureList":     [
      "feature1",
      "feature2"
    ],
    "memory": 0,
    "os": "",
    "taskCount": 1
  },
  "reservationGroup": "",
  "resources":   {
    "PROCS": 2,
    "MEM": 1024,
    "DISK": 1024,
    "SWAP": 1024,
    "other1": 17,
    "other2": 42
  },
  "startDate": "2011-11-14 20:15:50 UTC",
  "subType": "Other",
  "trigger": {
    "eventType":"START",
    "actionType":"EXEC",
    "action":"date"
  },
  "variables":   {
```

```
    "var1": "val1",
    "var2": "val2"
  }
}
```

This example is to create a reservation if no conflicting reservations are found. (This is the equivalent to `mrsvctl -c -h node01 -E`.)

```
JSON request body
-----------------------------------

{
  "flags":   [
    "DEDICATEDRESOURCE"
  ],
  "hostListExpression": "node01"
}
```

Sample response

```
JSON Response for successful POST
-----------------------------------

{"id": "system.44"}
```

Modifying Reservations

The HTTP PUT method is used to modify **Reservations**.

Quick reference

```
PUT http://localhost:8080/mws/rest/reservations/<id>?api-version=3&change-
mode=<add|remove|set>
```

Modify Reservation

URLs and parameters

```
PUT http://localhost:8080/mws/rest/reservations/<id>?api-version=3&change-
mode=<add|remove|set>
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.
change-mode	Yes	String	add remove set	If **add**, add the given variables to the variables that already exist. If **remove**, delete the given variables from the variables that already exist. If **set**, replace all existing variables with the given variables.

See Global URL Parameters for available URL parameters.

Request body

The request body below shows all the fields that are available when modifying a **Reservation**, along with some sample values.

```
JSON request body for reservation modify
------------------------------------

{
  "variables":   {
    "var1": "val1",
    "var2": "val2"
  }
}
```

Sample response

> This message may not match the message returned from Moab exactly, but is given as an example of the structure of the response.

```
JSON response
-------------------------------------

{"messages":["reservation 'system.43' attribute 'Variable' changed."]}
```

Restrictions

You can change the ACL Rules on a reservation, but not using this resource. See Create or Update ACL.

Releasing Reservations

The HTTP DELETE method is used to release **Reservations**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/reservations/<id>?api-version=3
```

Release Reservation

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/reservations/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON Response for successful DELETE
-----------------------------------

{}
```

Related Topics

- Fields: Reservations
- Resources Introduction

Resource Types

This section describes behavior of the **Resource Type** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

> The **Fields: Resource Types** reference contains the type and description of all fields in the **Resource Type** object.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/resource-types	Get All Resource Types	--	--	--

This topic contains these sections:

- Getting Resource Types
  - Get All Resource Types

Getting Resource Types

The HTTP GET method is used to retrieve **Resource Type** information.

Quick reference

```
GET http://localhost:8080/mws/rest/resource-types?api-version=3
```

Get All Resource Types

URLs and parameters

```
GET http://localhost:8080/mws/rest/resource-types?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/resource-types?api-version=3&fields=id
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results":   [
    {"id": "throttle_migrate"}
  ]
}
```

Related Topics

- Fields: Resource Types
- Resources Introduction

Roles

This section describes behavior of the **Role** resource in Moab Web Services. The role resource is used to control access to MWS resources based on the proxy-user. Each role is attached to a principal and contains a list of proxy-user permissions that the group can use in MWS. This section describes the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Roles** reference section contains the type and description of all fields in the **Role** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/roles	Get All Roles Get Default Permissions on Default Roles	--	Create Single Role	--
/rest/roles/<id>	Get Single Role	Modify Single Role Reset Role Permissions	--	Deleting Roles
/rest/roles/<name>	Get Single Role	Modify Single Role Reset Role Permissions	--	Delete Single Role

This topic contains these sections:

- Getting Roles
  - Get All Roles
  - Get Default Permissions on Default Roles
  - Get Single Role
- Creating Roles
  - Create Single Role
- Modifying Roles
  - Modify Single Role
  - Reset Role Permissions
- Deleting Roles
  - Delete Single Role

Getting Roles

The HTTP GET method is used to retrieve **Role** information. You can query all objects or a single object.

Quick reference

```
GET http://localhost:8080/mws/rest/roles?api-version=3[&query={"field":"value"}&sort=
{"field":<1|-1>}]
GET http://localhost:8080/mws/rest/roles/<id>?api-version=3
GET http://localhost:8080/mws/rest/roles/<name>?api-version=3
```

Get All Roles

URLs and parameters

```
GET http://localhost:8080/mws/rest/roles?api-version=3[&query={"field":"value"}&sort=
{"field":<1|-1>}]
```

Parameter	Required	Type	Valid values	Description	Example
query	No	JSON	--	Queries for specific results. It is possible to query roles by one or more fields based on MongoDB query syntax.	`query={"name":"Acme-User-Role"}`
sort	No	JSON	--	Sort the results. Use `1` for ascending and `-1` for descending.	`sort={"name":-1}`

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/roles?api-version=3&fields=id,name
-----------------------------------

{
  "totalCount": 1,
  "resultCount": 1,
  "results": [  {
    "id": "4fa197e68ca30fc605dd1cf0",
    "name": "Acme-User-Role"
  }]
}
```

Sorting and querying

See the sorting and querying sections of Global URL Parameters.

Get Default Permissions on Default Roles

The defaults parameter is used to list the default permissions that are attached to the default roles.

URLs and parameters

```
GET http://localhost:8080/mws/rest/roles?api-version=3&defaults=true
```

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
------------------------------------
{
  "totalCount": 2,
  "resultCount": 2,
  "results":   [
        {
      "name": "HPCUser",
      "description": "Basic user, with permission to create and manage their own
jobs",
      "scope": "GLOBAL",
      "permissions":       [
                {
          "action": "read",
          "administrator": false,
          "description": "Read nodes",
          "fieldPath": "*",
          "id": "5612b526e4b0b5b9bc0db341",
          "label": "read-nodes",
          "resource": "nodes",
          "resourceFilter": null,
          "scope": "GLOBAL",
          "type": "domain",
          "version": 0
        },
                {
          "action": "create",
          "administrator": false,
          "description": "Create jobs",
          "fieldPath": null,
          "id": "5612b526e4b0b5b9bc0db345",
          "label": "create-jobs",
          "resource": "jobs",
          "resourceFilter": null,
          "scope": "GLOBAL",
          "type": "domain",
          "version": 0
        },
                ...
      ]
    },
        {
      "name": "HPCAdmin",
      "description": "Administrative user, with privileges for all features and jobs",
      "scope": "GLOBAL",
      "permissions":       [
                {
          "action": "read",
          "administrator": false,
          "description": "Read nodes",
          "fieldPath": "*",
          "id": "5612b526e4b0b5b9bc0db341",
          "label": "read-nodes",
          "resource": "nodes",
          "resourceFilter": null,
          "scope": "GLOBAL",
          "type": "domain",
          "version": 0
        },
                {
          "action": "update",
          "administrator": false,
          "description": "Reprovision nodes",
          "fieldPath": "operatingSystem.image",
          "id": "5612b526e4b0b5b9bc0db342",
          "label": "update-nodes-image",
          "resource": "nodes",
```

```
          "resourceFilter": null,
          "scope": "GLOBAL",
          "type": "domain",
          "version": 0
        },
                ...
      ]
    }
  ]
}
```

Get Single Role

URLs and parameters

```
GET http://localhost:8080/mws/rest/roles/<id>?api-version=3
GET http://localhost:8080/mws/rest/roles/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Role.
name	Yes	String	--	The name of the Role.

You must specify either **id** or **name**, but you do not have to specify both.

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/roles/Acme-User-Role?api-version=3
-----------------------------------

{
       "description" : "This is a role for normal users in the Acme BU Group.",
       "id" : "5022e695e4b073f54e47c28d",
       "name" : "Acme-User-Role",
       "permissions" : [ {
              "action" : "create",
              "administrator" : null,
              "description" : "The permission to create all charts.",
              "id" : "5022e695e4b073f54e47c28e",
              "label" : "Create Chart",
              "resource" : "chart",
              "resourceFilter" : null,
              "type" : "custom",
              "scope" : "GLOBAL",
              "version" : 0
       }, {
              "action" : "read",
              "administrator" : null,
              "description" : "The permission to view all charts.",
              "id" : "5022e695e4b073f54e47c28f",
              "label" : "View Chart",
              "resource" : "chart",
              "resourceFilter" : null,
              "type" : "custom",
              "scope" : "GLOBAL",
              "version" : 0
       }, {
              "action" : "update",
              "administrator" : null,
              "description" : "The permission to modify the africa chart.",
              "id" : "5022e695e4b073f54e47c290",
              "label" : "Modify Africa Chart",
              "resource" : "chart",
              "resourceFilter" : {
                     "name" : "africa"
              },
              "type" : "custom",
              "scope" : "GLOBAL",
              "version" : 0
       }, {
              "action" : "read",
              "administrator" : null,
              "description" : "The permissions to view John's services.",
              "id" : "5022e695e4b073f54e47c291",
              "label" : "Read John's services",
              "resource" : "services",
              "resourceFilter" : {
                     "user":"john"
              },
              "type" : "api",
              "scope" : "GLOBAL",
              "version" : 0
       } ],
       "version" : 2
}
```

Creating Roles

The HTTP POST method is used to submit **Roles**.

Quick reference

```
POST http://localhost:8080/mws/rest/roles?api-version=3
```

Create Single Role

URLs and parameters

```
POST http://localhost:8080/mws/rest/roles?api-version=3
```

See Global URL Parameters for available URL parameters.

Request body

> The **name** field is required and must contain only letters, digits, periods, dashes, and underscores.

The following is an example of a request body to create a role:

```
POST http://localhost:8080/mws/rest/roles?api-version=3
-----------------------------------

{
        "name" : "Acme-User-Role",
        "description" : "This is a role for normal users in the Acme BU Group.",
        "permissions" :
        [
                {
                "id" : "4fa197e68ca30fc605dd1cf0"
                },
                {
                "id" : "4fa197e68ca30fc605dd1df2"
                }
        ]
}
```

Sample response

If the request was successful, the response body is the new role that was created, exactly as shown in **Get Single Role**. On failure, the response is an error message.

Samples

The **permissions** field only expects an array of permission IDs, as shown in the following example:

```
Example payload of role with 2 permissions
----------------------------------

{
        "name" : "Acme-User-Role",
        "description" : "This is a role for normal users in the Acme BU Group.",
        "permissions" :
        [
                {
                "id" : "4fa197e68ca30fc605dd1cf0"
                }
        ]
}
```

Modifying Roles

The HTTP PUT method is used to modify **Roles**.

Quick reference

```
PUT http://localhost:8080/mws/rest/roles/<id>?api-version=3
PUT http://localhost:8080/mws/rest/roles/<name>?api-version=3
```

Modify Single Role

URLs and parameters

```
PUT http://localhost:8080/mws/rest/roles/<id>?api-version=3
PUT http://localhost:8080/mws/rest/roles/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Role.
name	Yes	String	--	The name of the Role. The **name** field must contain only letters, digits, periods, dashes, and underscores.
change-mode	No	String	add remove set (default)	If **add**, adds the given permissions to the permissions that already exist. If **remove**, deletes the given permissions from the permissions that already exist. If **set**, adds the given permissions and deletes the permissions that already exist.

You must specify either **id** or **name**, but you do not have to specify both.

See Global URL Parameters for available URL parameters.

Example request

```
PUT http://locahost/mws/rest/role/Acme-User-Role?change-mode=add?api-version=3
-----------------------------------

{
       "permissions":[{"id":"4fa197e68ca30fc605dd1cf0"} ]
}
```

Sample response

If the request was successful, the response body is the modified role as shown in **Get Single Role**. On failure, the response is an error message.

Reset Role Permissions

The reset-permissions parameter is used to reset the permissions on a role to match the permissions of one of the default roles.

URLs and parameters

```
PUT http://localhost:8080/mws/rest/roles/<role>?api-version=3&reset-
permissions=<default-role>
```

Parameter	Required	Type	Valid values	Description
role	Yes	String	---	The role to be modified.
default-role	Yes	String	---	The name of the default role whose permissions will be applied to the <role>.

See Global URL Parameters for available URL parameters.

Request body

```
JSON request body (required)
-----------------------------------

{}
```

Sample response


```
{
  "description": "Basic user, with permission to create and manage their own jobs",
  "id": "5612b526e4b0b5b9bc0db389",
  "name": "HPCUser",
  "permissions":   [
        {
      "action": "read",
      "administrator": false,
      "description": "Read nodes",
      "fieldPath": "*",
      "id": "5612b526e4b0b5b9bc0db341",
      "label": "read-nodes",
      "resource": "nodes",
      "resourceFilter": null,
      "scope": "GLOBAL",
      "type": "domain",
      "version": 0
    },
        {
      "action": "create",
      "administrator": false,
      "description": "Create jobs",
      "fieldPath": null,
      "id": "5612b526e4b0b5b9bc0db345",
      "label": "create-jobs",
      "resource": "jobs",
      "resourceFilter": null,
      "scope": "GLOBAL",
      "type": "domain",
      "version": 0
    },
        ...
  ],
  "scope": "GLOBAL",
  "version": 2
}
```


Deleting Roles

The HTTP DELETE method is used to delete **Roles**.

Quick reference

```
DELETE http://localhost:8080/mws/rest/roles/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/roles/<name>?api-version=3
```

Delete Single Role

URLs and parameters

```
DELETE http://localhost:8080/mws/rest/roles/<id>?api-version=3
DELETE http://localhost:8080/mws/rest/roles/<name>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the Role.

Parameter	Required	Type	Valid values	Description
name	Yes	String	--	The name of the Role.

You must specify either **id** or **name**, but you do not have to specify both.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{}
```

Related Topics

- Fields: Roles
- Resources Introduction

Standing Reservations

This section describes behavior of the **Standing Reservation** object in Moab Web Services. It contains the URLs, request bodies, and responses delivered to and from MWS.

The **Fields: Standing Reservations** reference section contains the type and description of all fields in the **Standing Reservation** object. It also contains details regarding which fields are valid during PUT and POST actions.

Supported methods

Resource	GET	PUT	POST	DELETE
/rest/standing-reservations	Get All Standing Reservations	--	--	--
/rest/standing-reservations/<id>	Get Single Standing Reservation	--	--	--

This topic contains these sections:

- Getting Standing Reservations
  - Get All Standing Reservations
  - Get Single Standing Reservation

Getting Standing Reservations

The HTTP GET method is used to retrieve **Standing Reservation** information. Queries for all objects and a single object are available.

Quick reference

```
GET http://localhost:8080/mws/rest/standing-reservations/<id>?api-version=3
```

Get All Standing Reservations

URLs and parameters

```
GET http://localhost:8080/mws/rest/standing-reservations?api-version=3
```

See Global URL Parameters for available URL parameters.

Sample response

```
GET http://localhost:8080/mws/rest/standing-reservations?api-version=3&fields=id
-----------------------------------

{
  "totalCount": 3,
  "resultCount": 3,
  "results":   [
    {"id": "sr1"},
    {"id": "sr2"},
    {"id": "sr3"}
  ]
}
```

Get Single Standing Reservation

URLs and parameters

```
GET http://localhost:8080/mws/rest/standing-reservations/<id>?api-version=3
```

Parameter	Required	Type	Valid values	Description
id	Yes	String	--	The unique identifier of the object.

See Global URL Parameters for available URL parameters.

Sample response

```
JSON response
-------------------------------------

{
  "access": "DEDICATED",
  "accounts": ["account1"],
  "aclRules": [  {
    "affinity": "POSITIVE",
    "comparator": "EQUAL",
    "type": "USER",
    "value": "adaptive",
  }],
  "chargeAccount": "account2",
  "chargeUser": "user2",
  "classes": ["class1"],
  "clusters": ["cluster1"],
  "comment": "comment",
  "days": ["Monday"],
  "depth": 2,
  "disabled": false,
  "endOffset": 86415,
  "flags": ["ALLOWJOBOVERLAP"],
  "groups": ["group1"],
  "hosts": ["host1"],
  "id": "fast",
  "jobAttributes": ["TEMPLATESAPPLIED"],
  "maxJob": 2,
  "maxTime": 0,
  "messages": ["message1"],
  "nodeFeatures": ["feature1"],
  "os": "Ubuntu 10.04.3",
  "owner":   {
    "name": "root",
    "type": "USER"
  },
  "partition": "ALL",
  "period": "DAY",
  "procLimit":   {
    "qualifier": "<=",
    "value": 5
  },
  "psLimit":   {
    "qualifier": "<=",
    "value": 60
  },
  "qoses": ["qos1"],
  "reservationAccessList": [],
  "reservationGroup": "group2",
  "resources":   {
    "PROCS": -1,
    "tapes": 1
  },
  "rollbackOffset": 43200,
  "startOffset": 347040,
  "taskCount": 0,
  "tasksPerNode": 0,
  "timeLimit": -1,
  "triggers": [],
  "type": "type1",
  "users": ["user1"]
}
```

Related Topics

- Fields: Standing Reservations
- Resources Introduction

Chapter 5 Overview of Reporting Framework

The reporting framework is a set of tools to make time-based reports from numerical data. The following sections will (1) provide an overview of the framework and the concepts related to it, and (2) work through an example report (CPU Utilization) with details regarding which web services to use and with what data.

The REST API reference is located in the Report resource section (see Reports).

Concepts

The reporting framework uses 3 core concepts: reports, datapoints, and samples.

- Reports (see Fields: Reports): A report is a time-based view of numerical data.
- Report Datapoints (see Fields: Report Datapoints): A datapoint is a consolidated set of data for a certain time period.
- Report Samples (see Fields: Report Samples): A sample is a snapshot of a certain set of data at a particular point in time.

To illustrate, consider the memory utilization of a virtual machine: at any given point in time, you can get the memory utilization by using your operating system's performance utilities (top for Linux, Task Manager for Windows):

```
2400/12040MB
```

By recording the memory utilization and time constantly for 1 minute, you could gather the following data:

Time	Memory utilization
3:53:55 PM	2400/12040 MB
3:54:13 PM	2410/12040 MB
3:54:27 PM	2406/12040 MB
3:54:39 PM	2402/12040 MB
3:54:50 PM	2409/12040 MB

Each of the rows in the table above represent a **sample** of data. By averaging the rows we can consolidate them into one or more **datapoints**:

Start time	End time	Memory utilization
3:53:30 PM	3:54:00 PM	2400/12040 MB
3:54:00 PM	3:54:30 PM	2408/12040 MB
3:54:30 PM	3:55:00 PM	2406/12040 MB

Note that each datapoint covers exactly the same amount of time, and averages all samples within that period of time.

A **report**, then, is simply a list of datapoints with some additional configuration information:

Field	Value
Name	Memory Utilization Report
Datapoint Duration	30 seconds
Report Size	3 datapoints

Datapoints:

Start time	End time	Memory utilization
3:53:30 PM	3:54:00 PM	2400/12040 MB
3:54:00 PM	3:54:30 PM	2408/12040 MB
3:54:30 PM	3:55:00 PM	2406/12040 MB

Capabilities

While storing simple information like memory utilization is nice, the reporting framework is built to automatically handle much more complex information.

Consolidating Samples

Samples are JSON documents which are pushed into the report using the Samples API (see Creating Samples). Samples are then stored until the consolidation operation creates a datapoint out of them. The table below shows how different data types are handled in this operation:

Type	Consolidation function handling
Numbers	Numerical data is averaged.
Strings	Strings are aggregated into an array.
Objects	The consolidation function recursively consolidates sub-objects.
Lists	Lists are combined into a single flat list containing all elements.
Mixed	If samples have different types of data for the same field, the values are aggregated into an array.
Null	These values will be ignored unless all values for a sample field are set to null, resulting in a null result.

If the mixed data types contains at least one number, it will be treated as numerical data. The non-numerical data will be ignored and the result will be averaged.

Below is an example of how the consolidation function works:

- Samples:

Time	NumberEx	StringEx	ListEx	MixedEx	MixedNumberEx
3:53:55 PM	2400	"str1"	["elem1"]	"str1"	"str1"
3:54:13 PM	2410	"str2"	["elem2", "elem3"]	["elem1"]	["elem1"]
3:54:27 PM	2405	"str3"	["elem4"]	null	5

- Resulting Datapoint after consolidation:

Time	NumberEx	StringEx	ListEx	MixedEx	MixedNumberEx
3:55:00 PM	2405	["str1", "str2", "str3"]	["elem1", "elem2", "elem3", "elem4"]	["str1", "elem1"]	5

Minimum number of samples

If your dataset is highly variable (i.e. values contained in samples are not very close together), converting a single sample into a datapoint may provide misleading information. It may be better to have a datapoint with an "Unknown" value. This can be accomplished by setting the minimum number of samples for a datapoint in the report.

The `minimumSampleSize` field in the Reports reference section (see Reports) explains that if the specified size of samples is not met when the consolidation function is performed, the datapoint is considered "null" and no data is available for it. When this occurs, the sample data is discarded and the `data` field of the datapoint is set to "null".

For information on how to set this option, see the REST API Report Resource section (see Reports).

Report size

Reports have a predetermined number of datapoints, or size, which sets a limit on the amount of data that can be stored. After the report size has been reached, as newly created datapoints are pushed into the report, the oldest datapoints will automatically be deleted. This is to aid in managing the storage capacity of the server hosting MWS.

> **Warning:** On report creation, a Mongo collection will be initialized that is the configured report document size multiplied by the report size. Be careful in setting a large report size or report document size as this may quickly allocate the entire disk. See the `reportDocumentSize` and `reportSize` fields in Fields: Reports for more information.

Related Topics

- Example Report (CPU Utilization)

Example Report (CPU Utilization)

To understand how the behavior and usage of the reporting framework, a sample report covering CPU Utilization will be shown in this section. It will not cover how to gather or display data for reports, but will cover some basic operations that are available with Moab Web Services to facilitate reporting.

Creating a Report

Before any data is sent to Moab Web Services, a report must first be created. A JSON request body with a HTTP method of POST must be used to do this.

```
POST /rest/reports
-----------------------------------

{
  "name":"cpu-util",
  "description":"An example report for cpu utilization",
  "consolidationFunction":"average",
  "datapointDuration":600,
  "reportSize":288
}
```

This will result in a report being created which can then be retrieved by sending a GET request to `/rest/reports/cpu-util`. The `datapointDuration` of `600` signifies that the datapoint consolidation should occur once every 10 minutes, while the `reportSize` (i.e. number of the datapoints) shows that the report will retain up to 2 days worth of the latest datapoints.

```
GET /rest/reports/cpu-util
-----------------------------------

{
    "consolidationFunction": "average",
    "datapointDuration": 600,
    "datapoints": [],
    "description": "An example report for cpu utilization",
    "id": "aef6f6a3a0bz7bf6449537c9d",
    "keepSamples": false,
    "minimumSampleSize": 1,
    "name": "cpu-util",
    "reportSize": 288,
    "version": 0
}
```

(Note that an ID has been automatically generated and that no datapoints are associated with the report.)

Adding Samples

Until samples are added and associated with the report, datapoint consolidation will generate datapoints with a `data` field equal to `null`. Once samples are added, however, they will be averaged and inserted into the next datapoint.

Create samples for the `cpu-util` by sending a POST request as follows:

```
POST /rest/reports/cpu-util/samples
-----------------------------------

[
  {
    "agent": "cpu-monitor",
    "timestamp":"2012-01-01 12:00:00 UTC",
    "data": {
      "minutes1": 0.5,
      "minutes5": 0,
      "minutes15": 0
    }
  },
  {
    "agent": "cpu-monitor",
    "timestamp":"2012-01-01 12:01:00 UTC",
    "data": {
      "minutes1": 1,
      "minutes5": 0.5,
      "minutes15": 0.05
    }
  },
  {
    "agent": "cpu-monitor",
    "timestamp":"2012-01-01 12:02:00 UTC",
    "data": {
      "minutes1": 1,
      "minutes5": 0.5,
      "minutes15": 0.1
    }
  },
  {
    "agent": "cpu-monitor",
    "timestamp":"2012-01-01 12:03:00 UTC",
    "data": {
      "minutes1": 0.75,
      "minutes5": 1,
      "minutes15": 0.25
    }
  },
  {
    "agent": "cpu-monitor",
    "timestamp":"2012-01-01 12:04:00 UTC",
    "data": {
      "minutes1": 0,
      "minutes5": 1,
      "minutes15": 0.85
    }
  }
]
```

This sample data contains average load for the last 1, 5, and 15 minute intervals. The samples were recorded at one-minute intervals starting at noon on January 1st, 2012.

Consolidating Data

A consolidation function must run to generate datapoints from the given samples. This scheduled consolidation will occur at intervals of `datapointDuration` seconds. For each field in the `data` object in samples, all

values will be averaged. If non-numeric values are included, the following strategies will be followed:

1. All fields which contain a single numeric value in any included sample will be averaged and the non-numeric or null values will be ignored.
2. All fields which contain a list will be consolidated into a single, flat list.
3. All fields which contain only non-numeric or null values will be consolidated into a single, flat list.

If no historical datapoints are provided in the creation of a report as in this example, the next consolidation will be scheduled for the current time plus the `datapointDuration`. In this example, the scheduled consolidation is at 10 minutes from the creation date. If historical datapoints are included in the report creation, the latest datapoint's `endDate` plus the `datapointDuration` will be used as the scheduled time. If this date was in the past, the next scheduled consolidation will occur at the appropriate interval from the last `endDate`.

Retrieving Report Data

To retrieve the consolidated datapoints, simply perform a GET request on the report once again. Alternatively, the GET for a report's datapoints (see Get Datapoints for Single Report) may be used.

```
GET /rest/reports/cpu-util
----------------------------------

{
    "consolidationFunction": "average",
    "datapointDuration": 600,
    "datapoints": [
        {
            "firstSampleDate": null,
            "lastSampleDate": null,
            "data": null,
            "startDate": "2012-01-01 11:49:00 UTC",
            "endDate": "2012-01-01 11:59:00 UTC"
        },
        {
            "firstSampleDate": "2012-01-01 12:00:00 UTC",
            "lastSampleDate": "2012-01-01 12:04:00 UTC",
            "data": {
                "minutes1": 0.65,
                "minutes15": 0.25,
                "minutes5": 0.6
            },
            "startDate": "2012-01-01 11:59:00 UTC",
            "endDate": "2012-01-01 12:09:00 UTC"
        }
    ],
    "description": "An example report for cpu utilization",
    "id": "aef6f6a3a0bz7bf6449537c9d",
    "keepSamples": false,
    "minimumSampleSize": 1,
    "name": "cpu-util",
    "reportSize": 288,
    "version": 0
}
```

Note that of the two datapoints above, only the second actually contains data, while the other is set to `null`. Only samples lying within the datapoint's duration, or from the `startDate` to the `endDate`, are included in the consolidation. Therefore the first datapoint, which covered the 10 minute period just before the samples' recorded timestamps, contained no data. The second, which covers the 10 minute period matching that of the samples, contains the averaged sample data. This data could be used to display consolidated report data in a custom interface.

Possible Configurations

Configuration options may be changed to affect the process of report generation. These are documented in Fields: Reports and Fields: Report Samples.

Related Topics

- Overview of Reporting Framework

Chapter 6 About Moab Web Services Plugins

This chapter describes MWS plugins, their use, and their creation in Moab Workload Manager. The sections in this chapter provide you with the following information:

- An introduction to the concept of MWS plugins (see Plugin Introduction).
- A description of the plugin lifecycle (see Lifecycle States).
- How plugins are driven by events (Handling Events).
- How to expose web services from a plugin (Exposing Web Services).
- How plugin utility services may be used (Utility Services).
- How data report collisions between plugins are consolidated (Data Consolidation).
- How calls from Moab are routed to MWS plugins (Routing).

It contains the following sections:

- Plugin Overview
- Plugin Developer's Guide
- Plugin Type Management
- Plugin Management
- Plugin Services

Related Topics

- Configuring Moab Web Services

Plugin Overview

This section provides an overview of the plugin layer in web services. It contains these topics:

- Plugin Introduction
- Lifecycle States
- Events
- Custom Web Services
- Utility Services
- Data Consolidation
- Routing

Related Topics

- About Moab Web Services Plugins

Plugin Introduction

Moab Web Services plugins provide a highly extensible interface to interact with Moab, MWS, and external resources. Plugins can perform some of the same functions as Moab resource managers (RMs), while also providing many other features not available to RMs. This section will discuss the main features of plugins, some basic terminology, and how MWS plugins can interact with Moab.

Features

Plugins can:

- Be created, modified, and deleted without restarting Moab Workload Manageror MWS.
- Be defined in Groovy and uploaded to MWS without restarting.
- Have individual data storage space and configuration.
- Access MWS configuration and RESTful web services.
- Log to a standard location configured in MWS.
- Be polled at a regular interval (configured on a per-plugin basis).
- Be informed of important system events.
- Be individually stopped, started, paused, and resumed.
- Expose secured and unsecured custom web services for external use.
- Be manipulated via a full RESTful API (for more information, see Resources Introduction).
- Be manipulated via a full user interface in a web browser.

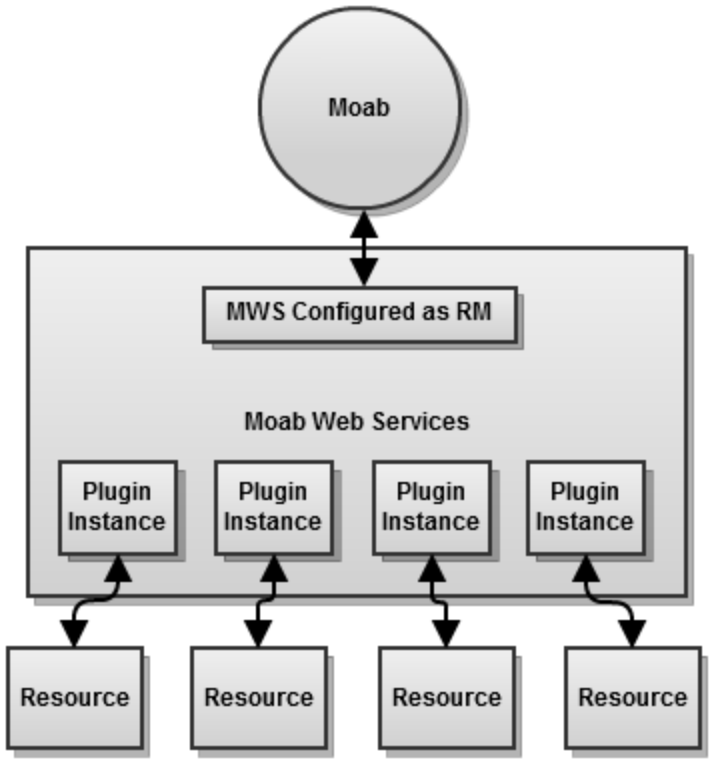
Terminology

There are two distinct terms in the plugin layer: **plugin types** and **plugins (instances)**.

Term	Description
plugin types	Plugin types can be considered plugin templates with built-in logic. In object-oriented programming languages, this relates to the concept of a class. They possess certain abilities, or methods, that can be called by Moab Web Services to query or update information about certain resources. They also can define methods which will be exposed to external clients as web services. They do not contain any configuration or current data, but they are often tied to a *type* of component, such as components that communicate with Moab's WIKI Protocol, or those that are built on a certain product. They can define several types of methods: • **Instance methods** that return information about the current plugin, such as `getState`. (While these are defined in the plugin type, the plugin type itself does not have a state.) • The **poll event method** that is called at a configured interval. • **Lifecycle event methods** of plugins created from the plugin type, such as `beforeStart` and `afterStart`. • **RM event methods** that are called by Moab when certain events occur. • **Web service methods** that expose custom functionality as public web services. Some examples of plugin types include the Native and vCenter plugin types.
plugins (instances)	Plugins (also called plugin instances) are created from plugin types. They contain current data or configuration and use the plugin type methods to interact with resources.

Interactions with Moab as a resource manager

The plugin layer in MWS is integrated with Moab Workload Manager via the Native Resource Manager (RM) interface. When utilizing plugins, MWS is configured as a RM in Moab, as explained in the next section. Events from Moab are pushed through the RM interface to MWS, which is then pushed to each plugin in turn. The relationship between MWS, Moab, and plugins is shown in the following image:



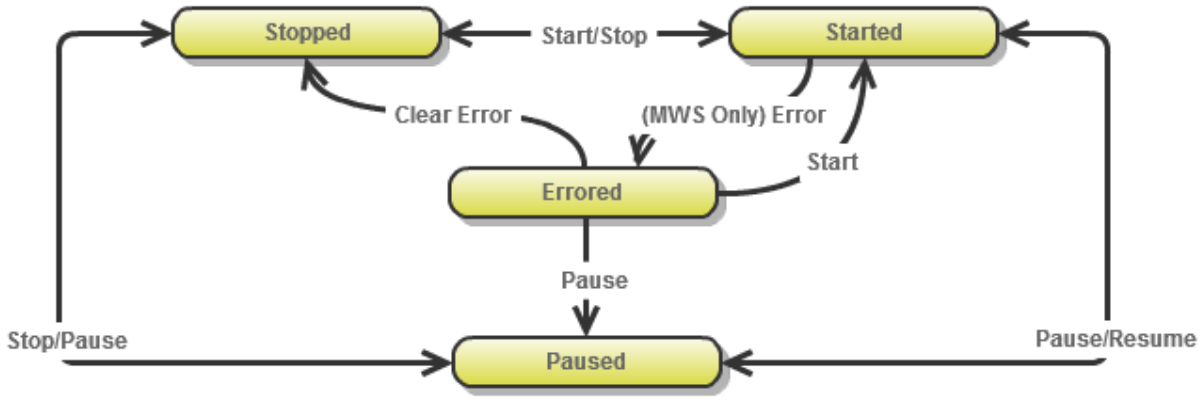
For more information, see Data Consolidation and Reporting State Data.

Related Topics

- About Moab Web Services Plugins

Lifecycle States

During the course of a plugin's use, the state of the plugin may change many times. Plugins have four possible states: **Stopped**, **Started**, **Paused**, and **Errored**. For the descriptions of each state, see the **Fields: Plugins** reference section. The flow of a plugin through the states is shown in the following image:



You can see Handling Events for information about the events that occur during lifecycle state changes.

Related Topics

- Plugin Introduction

Events

Plugins use an event-based model, meaning that methods are called on the plugin when certain criteria are met or situations arise. Events currently exist for polling, lifecycle state changes, and RM events from Moab. For more information, see Handling Events.

Related Topics

- Handling Events
- Plugin Introduction

Custom Web Services

Although the events interface typically serves most cases, there are some instances where an event is not supported that is desired. This is especially true when an external resource is the source of the event. To address these issues, plugins can expose custom web services to external resources. These web services may be named freely and do anything they wish within the plugin framework.

For example, suppose a resource needs to notify a plugin that provisioning of a virtual machine has been completed. Instead of having the plugin poll the

resource to verify that the provisioning was finished, the plugin could expose a custom web service to handle notification from the resource itself.

```
Sample custom web service
---------------------------------


def vmProvisionFinished(Map params) {
        // Handle event
        return [messages:["Event successfully processed"]]
}
```

Additionally, plugin types may define web services which are unsecured, meaning that a user or application account is not required to access it. A full explanation of the syntax and creation of custom secured and unsecured web services may be seen on Exposing Web Services.

For information how resources can access plugin web services, see Accessing Plugin Web Services.

Related Topics

- Plugin Introduction

Utility Services

Several features of plugins are only available by utilizing bundled services. These include:

- Accessing the individual datastore (see Individual Datastore).
- Reporting state data to Moab through the Resource Manager interface (see Reporting State Data).
- Manipulating other plugins and controlling their lifecycle (see Controlling Lifecycle).
- Accessing REST resources from Moab Web Services (Accessing MWS REST Resources).

It may also be necessary or desired to create additional utility services when creating new plugin types. The easiest way to do this is to create a utility service which is called by convention a translator (see Using Translators), because it can typically "translate" from a specific resource or API to data which can be used by the plugin type.

Finally, custom components (see Registering Custom Components) may be used to fulfill use cases not covered by bundled services or custom translators.

Related Topics

- Plugin Introduction

Data Consolidation

At times, plugins can report differing or even contradictory data for nodes, virtual machines, and jobs. This is called a data "collision". The act of resolving these collisions is called "Consolidation." Plugins also have the concept of "precedence," where the plugins with the lowest precedence value are considered more authoritative than the greater precedence values plugins. For example, a plugin with a precedence value of 1 has a higher precedence and is considered more authoritative than a plugin with a precedence value of 5. If no precedence is provided when creating plugins, the plugin is automatically assigned to the lowest precedence, or 1 greater than the highest precedence value. The precedence value may not be less than 1.

When data from one plugin "collides" with another, the data from the highest precedence plugin will be considered the authoritative source for information. If multiple sets of data (reports) are provided by the same plugin, the latest set of data will take precedence. Additionally, MWS supports the concept of treating node and virtual machine data with state information **optimistically**, **pessimistically**, or **neither**. This is configured using the `plugins.stateConsolidationPolicy` configuration property in the MWS configuration file. If this property is set to `optimistic` and *any* plugin reports the state for a node or VM as "Up," the consolidated state will be "Up." Inversely, if the property is set to `pessimistic` and *any* plugin reports the state as "Down," the consolidated state will be "Down." If it is set to `null` (neither), consolidation will occur for the state field just as with any other field, with higher precedence and later reports being considered authoritative.

When MWS is upgraded to a version that supports plugin precedence from an older version, existing plugins will not have the precedence field set. The administrator should assign precedence to each plugin manually through the API (see Modifying Plugins) or through the user interface (see Modifying a Plugin) to ensure that the consolidation will occur as expected. By default, data from a plugin without a precedence defaults to a precedence of 1, or the highest precedence.

Consolidation examples

Suppose two plugins exist, `pluginA` and `pluginB`. Plugin "A" has a precedence of 1, and plugin "B" has a precedence of 2, meaning that plugin "A" is more authoritative. These plugins both report data for a node with an ID of `node1`. However, each reports a different node power state. Plugin "A" reports the power as `ON`, while plugin "B" reports the power as `OFF`. The data collision that occurs due to these two contradictory reports is resolved by the precedence of the plugins. Since plugin A has a higher precedence (lower number), it is considered authoritative and the node will be reported as `ON`.

Now suppose that the plugins also report differing node state for `node1`. In this case, the node state would depend on the `plugins.stateConsolidationPolicy` property. The different combinations

of report values compared to the state consolidation policy and the final reported state are shown in the table below.

Plugin "A" node state	Plugin "B" node state	State consolidation policy	Consolidated node state
ON	OFF	null (neither)	ON
OFF	ON	null (neither)	OFF
ON	OFF	optimistic	ON
OFF	ON	optimistic	ON
ON	OFF	pessimistic	OFF
OFF	ON	pessimistic	OFF

In general, it is recommended that no two plugins report the same resource or that they report different properties of the same resource. For example, if plugin "A" only modified the power state and plugin "B" only modified the available disk resource, these two plugins would work in harmony to provide a consistent view of the node resource.

For more information, see Reporting State Data and Resource Manager Queries.

Related Topics

- Plugin Introduction

Routing

Plugin routing is currently in *Beta*. Interfaces may change significantly in future releases.

Because Moab Web Services is configured as a Resource Manager (RM) in Moab Workload Manager, events are sometimes triggered by Moab through the RM interface. These actions could be migrating a virtual machine, starting a job, submitting a job, modifying a node, and so forth. The decisions regarding which plugins are affected and notified is termed **routing**.

Currently all plugins receive all commands from Moab. This means that each plugin will receive the command to start a job if sent from Moab, even if that plugin does not handle the job. This means that plugins must ensure they handle actions or commands only for resources which they report or handle.

Related Topics

- Plugin Introduction

Plugin Developer's Guide

Plugin types comprise the methods by which Moab may communicate with resource managers or other external components. They define all operations that can be performed for a "type" or "class" of plugins, hence the name "plugin type."

Several plugin types are provided with Moab Web Services, but it is easy to create additional plugin types and add their functionality to web services. This involves using Groovy, which is based on the Java programming language. This section describes the general guidelines and specifics of implementing new plugin types.

API classes and interfaces

There are several packages and classes available to assist in creating plugin types. These can all be found in the API documentation.

This section contains these topics:

- Requirements
- Dynamic Methods
- Logging
- i18n Messaging
- Configuration
- Configuration Constraints
- Individual Datastore
- Exposing Web Services
- Reporting State Data
- Controlling Lifecycle
- Accessing MWS REST Resources
- Creating Events and Notifications
- Handling Events
- Handling Exceptions
- Managing SSL Connections
- Utilizing Services or Custom "Helper" Classes

- Packaging Plugins
- Example Plugin Types

Related Topics

- About Moab Web Services Plugins

Requirements

This section discusses the requirements to create a basic functional plugin. The `com.adaptc.mws.plugins` package contains the abstract class AbstractPlugin that should form the basis of any new plugin type. However, this class need not be extended to create a functional plugin type. Only two requirements must be fulfilled for this:

1. The class name must end in `Plugin`.
2. There must exist `id` field getter and setter methods:

```
* public String getId();
* public void setId(String id);
```

The `id` field may be stored in whichever way desired as long as the getter and setter are available as shown above, but will most likely be implemented as follows:

```
class BasicPlugin {
       String id
}
```

In this case, `String id` will be expanded by the Groovy compiler to the full getter and setter method definitions given above. In other words, no explicit method definitions are actually needed. Note that the `BasicPlugin` shown above is able to be uploaded as a plugin type to MWS, but does not actually do anything.

It must also be noted that the `AbstractPlugin` class already implements an `id` field. Therefore, a plugin type that extends this class does not need to define the field as shown in the following example.

```
import com.adaptc.mws.plugins.AbstractPlugin

class BasicPlugin extends AbstractPlugin {
       // No ID field is needed since it exists in AbstractPlugin
}
```

Related Topics

- Plugin Developer's Guide

Dynamic Methods

These methods are currently in *Beta*. Interfaces may change significantly in future releases.

Several methods are dynamically inserted onto each plugin. These methods do not need to be included in the plugin class, and will be overwritten if included. Additionally, a logger is inserted into each plugin as discussed in the next section. The inserted methods are shown below (full definitions can be found in AbstractPlugin and AbstractPluginInfo):

- `public void start() throws PluginStartException;` (Equivalent to the start method in the Plugin Control Service.)
- `public void stop() throws PluginStopException;` (Equivalent to the stop method in the Plugin Control Service.)
- `public Log getLog();` (See Logging.)
- `public ConfigObject getAppConfig();` (See Configuration.)
- `public String message(Map parameters);` (See i18n Messaging.)
- `public String getPluginType();`
- `public PluginState getState();`
- `public Integer getPollInterval();`
- `public Boolean getAutoStart();`
- `public Map<String, Object> getConfig();` (See Configuration.)

Many of these methods are provided for convenience and are discussed in the linked pages or the following sections.

Related Topics

- Plugin Developer's Guide

Logging

Logging in plugin types uses the Apache Commons Logging and log4j libraries. Each plugin is injected with a method called `getLog` which can be used to access the configured logger. It returns an instance of org.apache.commons.logging.Log. Examples of using the logger are shown below.

The logger may used to register messages to the MWS log at several levels (in order of severity):

1. trace
2. debug
3. info
4. warn
5. error
6. fatal

Each of these levels is available as a method on the logger, for example:

```
public void poll() {
       getLog().debug("getLog() is equivalent to just using 'log' in Groovy")
       log.debug("This is a debug message and is used for debugging purposes only")
       log.info("This is a informational message")
       log.warn("This is a warning")
       log.error("This is an error message")
}
```

Logger name

Each logger in the MWS logging configuration has a name. In the case of plugins, it is comprised of the full class name, including the package, prepended by "`plugins.`". For example, a plugin class of "example.LoggingPlugin" will have access to a logger configured as `plugins.example.LoggingPlugin`.

Logging configuration

The logging configuration is done through the MWS configuration file. For more information on configuring loggers, see Configuring Moab Web Services. A good configuration for developing plugin types may be to add "plugins" at the debug level. Be sure to set the log level threshold down for the desired appender.

```
log4j = {
  …
  // Appender configuration
  ...
  debug "plugins"
}
```

Related Topics

- Plugin Developer's Guide

i18n Messaging

Plugins, translators, and custom components all have access to i18n messages. Utilizing messages requires the two following steps:

1. Including a file (or multiple files) that ends in "`messages.properties`" in the plugin JAR file.
2. Using the `message` method on a plugin type, translator, or custom component.

Including messages in plugin JAR file

Messages are defined using property files. These may be named anything as long as they end with "`messages.properties`" and must be placed at the root or top level of the plugin JAR file. If they are present, they will be loaded automatically. Multiple property files may be used within a single plugin JAR file.

Each property file consists of an arbitrary amount of lines that define a message property (also called a code) with letters, numbers, and periods, associated with a human-readable message that can span multiple lines, have quotes, or contain arguments. These are demonstrated in the following example.

```
first.message.code=This is the first message
second.message=This message can span multiple lines, \\
  and will not show the linebreaks when retrieved
message.with.arguments=This message has arguments: first - {0}, second - {1}, third -
{2}, etc.
message.with.quotes=This message uses single quotes around ''this phrase''.
```

It is recommended to namespace the messages by using the property definitions and multiple property files if necessary. For example, suppose a plugin JAR existed which actually contained two plugin types: `Message1Plugin` and `Message2Plugin`. The first suggestion is to namespace the messages for each plugin by the property definition, such as the following:

```
message1Plugin.first.message=This is a message for Message1Plugin
message2Plugin.first.message=This is a message for Message2Plugin
```

These messages could be stored in a file named "`messages.properties`" in the root of the plugin JAR file. If there are many messages contained for each plugin type, it may be necessary to split each plugin type's messages into a separate file, such as "`message1-messages.properties`" and "`message2-messages.properties`". Note that it is essential that each property file ends with "`messages.properties`" so that it is registered correctly.

It is important that no two message codes are identical within a single plugin JAR file, even if they are defined in separate property files. If this is done, a conflict will exist with the messages and behavior is undefined.

Using the message method

Each plugin, translator, and custom component is injected with a method named `message`. This method takes a Map as its parameter, which can contain one or several of the following properties:

Parameter	Type	Description
code	String	The message property definition (everything before the equals sign in the property file for a single message), for example, `first.message.code`.
args	List<Object>	A list of arguments to insert into the message.
default	String	A default message to be used when the message code cannot be resolved.
error	org.springframework.context.MessageSourceResolvable	An object that represents a hierarchy of message codes. This is typically used to display errors.

The most utilized parameters are `code` and `args`, as these combined provide great flexibility in generating messages. If a message cannot be resolved, or in other words the message definition does not exist, the code will simply be returned as the resolved message. Below are several examples of messages resolved using the property files given above. While these are contained in the polling method, the `message` may be used anywhere within a plugin type.

```
package example
import com.adaptc.mws.plugins.AbstractPlugin

class MessagingPlugin extends AbstractPlugin {
       def poll() {
               assert message(code:"first.message.code")=="This is the first message"
               assert message(code:"message.with.arguments", args:[
                               "1st", 2, true
                       ])=="This message has arguments: first - 1st, second - 2, third - true, etc."
               assert message(code:"message.with.quotes")=="This message uses single quotes around
'this phrase'."
               assert message(code:"invalid.message.code")=="invalid.message.code"
       }
}
```

Related Topics

- Plugin Developer's Guide

Configuration

Plugin types can access two different kinds of configuration: an individual plugin's configuration, and the global MWS application configuration.

Individual plugin configuration

The individual plugin configuration is separate for each instance of a plugin. This may be used to store current configuration information such as access information for linked resources. It should not be used to store cached information or non-configuration related data. The individual datastore should be used instead for these cases (for more information, see Individual Datastore).

It is accessed by using the `getConfig` method discussed in Dynamic Methods.

```
public void poll() {
  def configFromMethod = getConfig()
  // OR an even simpler method…
  def configFromMethod = config
}
```

A common case is to retrieve the configuration in the `configure` method, verify that it matches predetermined criteria, and utilize it perform initial setup of the plugin (e.g. initialize libraries needed to communicate with external resources). For example, to verify that the configuration contains the keys "username" and "password," the following code may be used.

```
public void configure() throws InvalidPluginConfigurationException {
  def myConfig = config
  // This checks to make sure the key exists in the configuration Map and that the
value is not empty or null
  if (!myConfig.containsKey("username") || !myConfig.username)
       throw new InvalidPluginConfigurationException("The username configuration parameter
must be provided")
  if (!myConfig.containsKey("password") || !myConfig.password)
    throw new InvalidPluginConfigurationException("The password configuration
parameter must be provided")
}
```

Access MWS configuration

The MWS application configuration can also be accessed in plugin types. This configuration is global for the entire application and can be modified by the administrator as shown in Configuring Moab Web Services.

It is accessed by using the `getAppConfig` method discussed in Dynamic Methods. This is demonstrated below:

```
public void poll() {
  // Retrieve the current MWS_HOME location
  def mwsHome = appConfig.mws.home.location
  // OR an even simpler method…
  def mwsHome = getAppConfig().mws.home.location
}
```

Any of the properties shown in the **Configuration** reference may be accessed. Custom properties may also be registered and accessed:

```
mws-config.groovy
-----------------------------------

plugins.custom.property = "This is my custom property"
```

```
CustomAppPropertyPlugin
-----------------------------------

public void poll() {
  assert appConfig.plugins.custom.property=="This is my custom property"
}
```

Related Topics

- Plugin Developer's Guide

Configuration Constraints

Plugin types can optionally define validation constraints for the polling interval and plugin configuration. These parameters are then checked against the defined constraints during the creation of a new plugin. If the validation fails, meaning the configuration provided does not pass the constraints defined by the plugin type, the plugin will fail to be created with error messages based on the parameters and constraints defined.

Defining constraints

To define constraints for a plugin type and therefore for all plugins created using it, use the following syntax:

```
import com.adaptc.mws.plugins.*
class ConstrainedPlugin extends AbstractPlugin {
       static constraints = {
               // Set plugin's default polling interval
               pollInterval defaultValue:60
               // The "myParam" configuration parameter is automatically required and cannot be
blank
               myParam blank:false
               // The "myEnum" configuration parameter is not required and must set to one of the
values in the list
               myEnum required:false, inList:["val1", "val2", "val3]
               // Insert additional constraints here…
       }
}
```

In the table below, all available constraints are shown, as well as the expected value type, an example, the default message code, and the message suffix. The message columns are described in greater detail in the **Messaging** section below.

Constraint	Default value	Type	Example value	Default message code	Message suffix	Description
blank	--	Boolean	`true`	default.blank.message	blank	If false, the parameter (if present) cannot be a blank string.
creditCard	--	Boolean	`true`	default.invalid.creditCard.message	creditCard.invalid	If **true**, uses `org.apache.commons.validator.CreditCardValidator` to determine if the parameter (if present) is a valid credit card number.
defaultValue	--	Object or Closure	`60`	--	--	If the parameter is not present, it will be set to this default value. Does not return any error messages. See **Default value** below for more information.
email	--	Boolean	`true`	default.invalid.email.message	email.invalid	If **true**, the parameter (if present) must be a valid email address.
inList	--	List	`["first", "second"]`	default.not.inlist.message	not.inList	The parameter (if present) must be set to one of the values specified.
matches	--	String	`"[a-z][A-Z]+"`	default.doesnt.match.message	matches.invalid	The parameter (if present) must match the specified regular expression.

Constraint	Default value	Type	Example value	Default message code	Message suffix	Description
max	--	Integer	`10`	default.invalid.max.message	max.exceeded	The parameter (if present) must not be greater than the defined value.
***maxSize**	--	Integer	`10`	default.invalid.max.size.message	maxSize.exceeded	The parameter's (if present) size must not be greater than the defined value.
min	--	Integer	`1`	default.invalid.min.message	min.notmet	The parameter (if present) must not be less than the defined value.
***minSize**	--	Integer	`1`	default.invalid.min.size.message	minSize.notmet	The parameter's (if present) size must not be less than the defined value.
notEqual	--	Object	`"Invalid Value"`	default.not.equal.message	notEqual	The parameter (if present) must *not* be set to the defined value.
nullable	true	Boolean	`false`	default.null.message	nullable	If **true**, the parameter (if present) must be non-null value. See **required** for how to enforce the parameter to be present.
password	--	Boolean	`true`	--	--	If **true**, the parameter (if present) is hidden from the user both on input and display when managing plugin configuration. It is not, however, hidden in the REST API. Does not return any error messages.

Constraint	Default value	Type	Example value	Default message code	Message suffix	Description
range	--	Range	`1..10`	default.invalid.range.message	range.toosmall/range.toobig	Uses a groovy range to validate that the value is within a specified range.
required	true	Boolean	`false`	default.required.message	required	If **true**, the parameter must be present and non-null for the plugin to be created successfully. Implies the `nullable:false` constraint.
scale	--	Integer	`2`	--	--	Only valid for Double parameters. Rounds the parameter (if present) to the specified number of digits. Does not return any error messages.
***size**	--	Range	`2`	default.invalid.size.message	size.toosmall/size.toobig	Uses a groovy range to restrict the size of a collection, string, or a number.
***type**	--	Class	`Integer.class`	typeMismatch	typeMismatch	See **Type inferencing and conversion** below.
url	--	Boolean	`true`	default.invalid.url.message	url.invalid	If **true**, uses `org.apache.commons.validator.UrlValidator` to determine if the parameter (if present) is a valid URL. Does not support `exec` or `file` scheme URLs.
scriptableUrl	--	Boolean	`true`	default.invalid.scriptable.url.mess	scriptableUrl.invalid	Identical to the **url** validator, but adds support for `exec` and `file` scheme URLs.

Constraint	Default value	Type	Example value	Default message code	Message suffix	Description
validator	--	Closure	(See Custom validator)	default.invalid.validator.message	validator.error	See **Custom validator** below.
widget	--	String	"textarea"	--	--	By default, all strings render as a text field when creating or editing plugins. Setting this to "textarea" causes it to render as a text area with multi-line support. This is only valid for string configuration parameters.

* The user interface (see Plugin Management) does not support parameters whose type is a subclass of Collection (a List, for example). Such parameters are therefore not recommended.

The polling interval constraints must always apply to Integer types. If this specification is violated, the plugin type cannot be added or updated.

Messaging

When defined constraints are violated for a plugin, error messages are retrieved based on the configuration parameters and the applied constraints using i18n Messaging codes (see i18n Messaging). First, the most specific error message will be attempted to be resolved from a message code generated from the plugin type name, the configuration parameter, and the constraint. This code takes the format of `pluginTypeName.parameterName.suffix` where the plugin type's name has a lowercase first letter and the suffix is shown in the table above. If this message code is not defined, the default message code (as shown in the table above) will be used.

For example, if the **url** constraint validation failed for the "ExamplePlugin" plugin type's "endpoint" configuration parameter, the following message codes would be resolved in order:

- `examplePlugin.endpoint.url.invalid`
- `default.invalid.url.message`

> Plugin types that have two or more uppercase letters at the start of the name will not be converted to have a lowercase first letter for error message codes. In other words, for the example just given using "VCenterPlugin" instead of "ExamplePlugin", the following message codes would be resolved in order:
>
> `VCenterPlugin.endpoint.url.invalid`
>
> `default.invalid.url.message`

Default messages

Default messages may be contained in any `messages.properties` file included in the plugin JAR file as explained in i18n Messaging (see i18n Messaging). Arguments for each constraint vary, but they always include these argument indices:

- `{0}`: The configuration parameter name (for example, `endpoint`).
- `{1}`: The plugin type class name (for example, `my.package.ExamplePlugin`).
- `{2}`: The value of the configuration parameter.

If default messages are not defined in the plugin project, the following messages will be used:

```
default.doesnt.match.message=The ''{0}'' configuration parameter value ({2}) does not
match the required pattern ''{3}''
default.invalid.url.message=The ''{0}'' configuration parameter value ({2}) is not a
valid URL
default.invalid.scriptable.url.message=The ''{0}'' configuration parameter value ({2})
is not a valid scriptable URL
default.invalid.creditCard.message=The ''{0}'' configuration parameter value ({2}) is
not a valid credit card number
default.invalid.email.message=The ''{0}'' configuration parameter value ({2}) is not a
valid e-mail address
default.invalid.range.message=The ''{0}'' configuration parameter value ({2}) does not
fall within the valid range from {3} to {4}
default.invalid.size.message=The ''{0}'' configuration parameter value ({2}) does not
fall within the valid size range from {3} to {4}
default.invalid.max.message=The ''{0}'' configuration parameter value ({2}) is greater
than the maximum value of {3}
default.invalid.min.message=The ''{0}'' configuration parameter value ({2}) is less
than the minimum value of {3}
default.invalid.max.size.message=The ''{0}'' configuration parameter value ({2})
exceeds the maximum size of {3}
default.invalid.min.size.message=The ''{0}'' configuration parameter value ({2}) is
less than the minimum size of {3}
default.invalid.validator.message=The ''{0}'' configuration parameter value ({2}) does
not pass custom validation
default.not.inlist.message=The ''{0}'' configuration parameter value ({2}) is not
contained within the list [{3}]
default.blank.message=The ''{0}'' configuration parameter cannot be blank
default.not.equal.message=The ''{0}'' configuration parameter value ({2}) cannot be
equal to ''{3}''
default.null.message=The ''{0}'' configuration parameter cannot be null
default.required.message=The ''{0}'' configuration parameter is required and cannot be
null
typeMismatch=The ''{0}'' configuration parameter value ({2}) does not match the
required type ''{3}''
```

Labels and help messages

Message codes may also be provided for configuration parameters to aid the admin user with human readable property labels and help messages. Similar to the validation error message codes, labels and help message codes may be defined using the `pluginTypeName.parameterName.label` and `pluginTypeName.parameterName.help` message codes. These values are used only in plugin type management (see Plugin Type Management) and are not exposed through the REST API.

Type inferencing and conversion

Due to the dynamic nature of configuration parameters, the expected type or class of values for each parameter are inferred from constraints. The following rules govern how type is inferred, in priority order:

- If the ***type** constraint is applied to a parameter, the constraint value will be used as the expected type.

> Only the `String`, `Date`, `Double`, `Integer`, and `Boolean` classes are supported for the ***type** constraint. If `Float` or `Long` is desired, use `Double` and `Integer` respectively as the type.

- If the **inList** or **range** constraints are applied to a parameter, the class of the first element in the constraint value array is used as the expected type.
- If the ***minSize** or ***maxSize** constraints are applied to a parameter, `java.lang.Collection` is used as the expected type.
- If the **max**, **min**, or **notEqual** constraints are applied to a parameter, the class of the constraint value is used as the expected type.
- If none of the above apply, `java.lang.String` is used as the expected type.

If the configuration parameter values can be converted to the expected types, this will occur automatically. Otherwise, the ***type** constraint is violated and the applicable error messages will be generated.

Custom validator

In cases where the built-in constraints prove inadequate for validation, custom validators may be used. The **validator** constraint expects a Groovy Closure parameter which has one or (optionally) two arguments: the value of the configuration parameter and the plugin object. With these parameters, complex validation logic may be defined. Additionally, custom message codes and arguments may be defined by validator constraints and these will be used in generating error messages when validation fails.

For example, suppose that the parameter "user" cannot be set to the same value as parameter "creator." Additionally, the "creator" parameter must not be equal to either "bob" or "joe." The existing constraints are inadequate to fulfill this use case, but the following code using validators would perform exactly as expected:

```
import com.adaptc.mws.plugins.*
class ConstrainedPlugin extends AbstractPlugin {
       static constraints = {
               user validator:{ val, obj ->
                       if (val==obj.config.creator)
                               return "invalid.equal.to.creator"
               }
               creator validator:{ val ->
                       if ("val"=="joe")
                               return ["invalid.equal", "joe"]
                       if (val=="bob")
                               return ["invalid.equal", "bob"]
               }
       }
}
```

In the examples above, the message codes and output on validation failure is shown below:

```
Message codes
-----------------------------------

constrainedPlugin.user.invalid.equal.to.creator=The user configuration parameter value
({2}) must not be equal to the creator parameter.
constrainedPlugin.creator.invalid.equal=The creator configuration parameter must not
be equal to {3}.
```

```
Output error messages
-----------------------------------

For user = "jill", creator = "jill"
"The user configuration parameter value (jill) must not be equal to the creator
parameter."
For user = "jill", creator = "bob"
"The creator configuration parameter must not be equal to bob."

For user = "jill", creator = "joe"
"The creator configuration parameter must not be equal to joe."
```

The validator Closure may return:

- Nothing (`null`) or `true` if the validation succeeded without errors.
- `false` if a validation error occurred (in this case the default validator message suffix would be used).
- A string which will be used as the message code suffix in the `pluginTypeName.propertyName.suffix` format.
- A list with the first element being the message code suffix, and all other elements being arguments for the message indexed starting at 3 (as shown in the example above).

All validator constraints automatically have the `appConfig` property available, which contains the application configuration as discussed in the Configuration section (see Configuration). The `suite` property contains the value of the configured MWS suite. Additionally, services may be retrieved as explained in the next section.

Retrieving services

At times it may be necessary to use Bundled Services in custom validators. A method named getService which takes a single string parameter of the name of the service (as used during injection) is provided to be used in these cases. For example, if a plugin needs a valid server certificate file, the SSL Service may be used as follows:

```
import com.adaptc.mws.plugins.*
class ConstrainedPlugin extends AbstractPlugin {
       static constraints = {
               certificateFile validator:{ val ->
                       ISslService sslService = getService("sslService")
                       try {
                               sslService.getSocketFactory(val)
                       } catch(Exception e) {
                               // Certificate file is invalid, return an error
                               return ["invalid", e.message]
                       }
               }
       }
}
```

The `getService` method does not work with translators, custom components, RM services, or the Individual Datastore.

Default value

The default value for a configuration parameter might depend on the MWS configuration or other properties. Therefore, the **defaultValue** constraint can be set to a closure. The `defaultValue` closure does not take any parameters and must return the object to be used as the default value.

For example, if the default value of a parameter must be true if and only if MWS is configured for the Cloud suite, then the following constraints would satisfy these conditions:

```
import com.adaptc.mws.plugins.*
class ConstrainedPlugin extends AbstractPlugin {
       static constraints = {
               myParameter required: true, type: Boolean, defaultValue: {
                       return suite == Suite.CLOUD
               }
       }
}
```

As with **validator** closures, `defaultValue` closures have access to `appConfig`, `suite`, and `getService`.

Related Topics

- Plugin Developer's Guide

Individual Datastore

Each plugin has access to an individual, persistent datastore which may be used for a variety of reasons. The datastore is not designed to store Moab data such as nodes, jobs, or virtual machines, but custom, arbitrary data pertinent only to the individual plugin. This may include storing objects in a persistent cache, state information for currently running processes, or any other arbitrary data. The individual datastore has the following properties:

- Data is persisted to the Mongo database and will be available even if the plugin or MWS is restarted.
- The data must be stored in groups of data called **collections**. These correspond directly to MongoDB collections.
- Each plugin may have an arbitrary number of collections.
- Collections are guaranteed not to collide if there are identically named collections between two plugin types or even two plugin instances.
- Each collection contains multiple objects or **entries**. These correspond directly to MongoDB documents.
- The values of entries may be any object which can be serialized to MongoDB: simple types (int or Integer), Maps, and Lists.
- A collection is automatically created whenever an entry is added to it, it does not need to be specifically initialized.

To utilize the datastore, the **Plugin Datastore Service** must be used. Operations are provided to add, query, and remove data from each collection.

> Simple key/value storage is not currently provided with the datastore. It may easily be done, however, by storing data in the format of `{name:"key", value:"value"}` and then retrieving this entry later by querying on name equals "key."

Example

The example below demonstrates two web services (see Exposing Web Services). The first adds multiple entries containing various types of data to an arbitrarily named collection. The second retrieves the data and returns it to the user.

```
package example
import com.adaptc.mws.plugins.*

class DatastorePlugin extends AbstractPlugin {
       IPluginDatastoreService pluginDatastoreService

       def storeData(Map params) {
              def collectionName = params.collectionName
              def data = [[boolVal:true], [stringVal:"String"], [intVal:1], [nullVal:null]]
              if (pluginDatastoreService.addData(collectionName, data))
                     log.info("Data successfully added")
              else
                     log.info("There was an error adding the data")
              return [success:true]
       }

       def retrieveData(Map params) {
              def collectionName = params.collectionName
              return pluginDatastoreService.getCollection(collectionName)
       }
}
```

Related Topics

- Plugin Developer's Guide

Exposing Web Services

Any number of methods may be exposed as public, custom web services by satisfying several criteria:

- The method must declare that it returns `Object` or `def`.
- The method must define a single argument of type `Map`.
- The method must actually return a `List` or `Map`.
- The method must not be declared as private or protected; only public or unscoped methods will be recognized as web services.

Parameters and request body

The `Map` argument will contain all parameters passed into the web service by the client. See Accessing Plugin Web Services for additional details.

Parameters may be passed into the web service call as normal URL parameters such as `?param=value&param2=value2`, as key-value pairs in the POST body of a request, or as JSON in the body.

For the first two cases, the parameters will be available on the Map argument passed into the web service call as key value pairs matching those of the request. Note that in these cases all keys and values will be interpreted as strings. However, the parameters object has several helper methods to convert from Strings to simple types, such as Booleans, integers, doubles, floats, and lists. If the value is not a valid simple type, null is returned.

Finally, note that the client may optionally include an `objectId` as the last part of the URL. When this is done, the **id** field will be set to this value in the `Map` argument to the web service.

```
GET <webServiceUrl>?key=value&key2=true&key3=5&list=1&list=2

def serviceMethod(Map params) {
       assert params.key=="value"
       assert params.key2=="true"
       assert params.bool('key2')==true
       assert params.key3=="5"
       assert params.int('key3')==5
       assert params.list('list')==[1, 2]

       // Null is returned if the conversion is invalid
       assert params.int('key')==null
}
```

When the body possesses JSON, the parsed JSON object or array will be available within a parameter called **body** in the `Map` argument. In this scenario, the types of the values are preserved by the JSON format.

```
POST <webServiceUrl> with JSON body of
{"key":"value","key2":true,"key3":5}

def serviceMethod(Map params) {
       assert params.body.key=="value"
       assert params.body.key2==true
       assert params.body.key3==5
}
```

Unsecured web services

There are times when it is desirable to create a plugin with a publicly available web service that does not require a valid application account in order to access it (for details, see Access Control). In these cases, the Unsecured annotation may be used on the plugin web service method. No authentication will be performed on Unsecured web services. An example of using the annotation is given below.

```
Sample unsecured custom web service
-----------------------------------

@Unsecured
def retrievePublicData(Map params) {
       return [data:["data item 1", "data item 2"]]
}
```

> **Be cautious in using this annotation as it may potentially present a security risk if sensitive data is returned from the web service.**

Returning errors

In order to signify an error occurred or invalid data was provided, the WebServiceException class may be thrown from any custom web service. This exception contains constructors and fields for a list of messages and a HTTP response code. For example, suppose that the user provided inadequate information. The web service could use the following code to notify the user and prompt them to take action with custom messages.

```
def service(Map params) {
       // Handle invalid input
       if (!params.int('a'))
              throw new WebServiceException("Invalid parameter 'a' specified, please specify an
integer!", 400)
       // Use params.a correctly …
}
```

For the example above, a 400 response code (bad request) would be returned with a response body as follows:

```
{
  "messages":[
       "Invalid parameter 'a' specified, please specify an integer!"
  ]
}
```

If any other exception is thrown from a web service (ie Exception, IllegalArgumentException, etc.), a 500 response code will be returned with the following response body:

```
{
  "messages":[
       "A problem occurred while processing the request",
       "Message provided in the exception constructor"
  ]
}
```

See Responses and Return Codes for more information on error formats in MWS.

Accessing the HTTP Request Method

The HTTP method used for the request is available from the Map parameters argument. The key used to access it is stored as a static field in PluginConstants called `WEB_SERVICES_METHOD`. The value is a string which can be `GET`, `POST`, `PUT`, or `DELETE`. The following example demonstrates how this could be used with the `WebServiceException` to create a REST API with a plugin.

```
def serviceMethod(Map params) {
       // Check to make sure that this request used the HTTP GET method
       // Throw a 405 error (method not supported) if not
       if (params[PluginConstants.WEB_SERVICES_METHOD]!="GET")
              throw new WebServiceException("Method is not supported", 405)
}
```

Related Topics

- Plugin Developer's Guide

Reporting State Data

As long as Moab Workload Manager is configured with MWS as a Resource Manager (RM), plugins may report state information on jobs, nodes, storage, and virtual machines to Moab. This is done through **Reports** that are generated by the plugin and passed to the bundled RM services (**Job RM Service**, **Node RM Service**, **Storage RM Service**, and **Virtual Machine RM Service**). Each report is for a specific type of object: job, node, storage, or virtual machine. Each contains current state information on the specific attributes of the type it is for.

Note that storage is a sub-type of node, meaning that it is a specialized node.

Generating reports

To generate a report, simply create a new instance of a report depending on the type of object to be reported:

Object type	Report type
Job	JobReport
Node	NodeReport
Storage	StorageReport
Virtual Machine	VirtualMachineReport

Each report has a single required parameter for creating a new instance—the ID of the object which is being reported. Once the report instance has been created, any property may be modified as shown in the API documentation links in the table above. The following example shows the creation of a simple node report and modification of a few properties:

```
public void poll() {
       NodeReport node = new NodeReport("node1")
       node.timestamp = new Date()
       node.image = "centos-6.6-stateless"
       … // Set other properties and persist the report
}
```

Master and slave reports

At times, you may want to report some additional attributes on objects *only if* the objects are being reported by other plugins. For example, you may want to report the power state of a VM, but sometimes the plugin reporting this data can receive data even after the VM has been destroyed. In this case, you can set the `slaveReport` field on any report to `true`, signifying that the report should only be used if another plugin is reporting on the same object (in other words, creating "master" reports).

> If all reports for an object are "slave" reports, and no "master" reports exist, then the object will not report to Moab Workload Manager.

Special cases in field values

All complex types, such as Lists, Maps, and objects (not including Enumerated values such as NodeReportState and JobReportState) have default values set for them and are not required to be instantiated before use. For example, the metrics property of a node report may be modified as follows:

```
public void poll() {
       NodeReport node = new NodeReport("node1")
       // The following assignments are equivalent in their functionality
       node.features.add("FEAT1")
       node.features << "FEAT2"
       // The following assignments are equivalent in their functionality
       node.metrics.METRIC1 = 4d
       node.metrics["METRIC2"] = 125.5
       … // Set other properties and persist the report
}
```

For the `resources` and `requirements` (jobs only) properties, assignments may be made easily without checking for previously existing values or null objects. For example, resources may be added to the `resources` property simply by accessing it as a Map:

```
public void poll() {
       NodeReport node = new NodeReport("node1")
       node.resources.RES1.total = 10
       node.resources.RES1.available = 3
       node.resources["RES2"].total = 10
       node.resources["RES2"].available = 10
       … // Set other properties and persist the report
}
```

The job report's `requirements` property has some additional handling to allow it to be accessed as a single JobReportRequirement object, such as in the following example:

```
public void poll() {
       JobReport job = new JobReport("job.1")
       job.nodeCountMinimum = 4
       job.processorCountMinimum = 2
       job.requiredNodeFeatures << "FEAT1"
       job.preferredNodeFeatures << "FEAT2"
       … // Set other properties and persist the report
}
```

Although multiple requirements may be added to the `requirements` list to provide consistency with the MWS Job resource (see Jobs), only the first requirement object's properties will be reported to Moab through the RM interface.

Managing images for nodes

In order to have Moab Workload Manager recognize a node as a virtual machine hypervisor, it must have a valid associated Image (see Images). In particular, the `image` property on a node report must set to a valid image name. The image's `hypervisorType` and `virtualizedImages` properties are then used to report the correct hypervisor type and supported virtual machine images to Moab.

If the `image` is invalid, it will be ignored and the node will not be recognized as a hypervisor. If the `image` is valid, but no `hypervisorType` value is present, the `extensions.xcat.hvType` field value will be used. If that is also not present,

the configuration parameter for default hypervisor type (see Configuration) will be used instead.

Persisting a Report

After a report has been generated and all desired fields have been updated, the report must be sent to one of the three bundled RM services for persisting. If this is not done, the report will be discarded and will not be considered when reporting state information to Moab. The RM services are shown below according to the object type that they handle:

Object type	RM service
Job	**Job RM Service**
Node	**Node RM Service**
Storage	**Storage RM Service**
Virtual Machine	**Virtual Machine RM Service**

Each service has two methods: `save` and `update`. The difference between these is that the `save` method first removes all previous reports from the plugin calling the method, and then persists the new reports, thereby only persisting the latest reports, while the `update` method does not remove any reports before persisting the new reports. Typically, the `save` method will be used while a plugin is being polled, while the `update` method will be used in incremental event based reporting. An example of using the `save` method is shown below.

```
INodeRMService nodeRMService

public void poll() {
       NodeReport node = new NodeReport("node1")
       // Change the state
       node.state = NodeReportState.BUSY
       // Persist
       nodeRMService.save([node])
}
```

Once this is done, the reports will be persisted to MongoDB and will be included in RM queries (see Resource Manager Queries) from Moab Workload Manager or users.

Related Topics

- Plugin Developer's Guide

Controlling Lifecycle

Plugin control is currently in Beta. Interfaces may change significantly in future releases.

At times a plugin developer may wish to modify the current state of a plugin or even create plugins programatically. This may be done with the **Plugin Control Service**. Operations exist on the service to:

- create plugin instances dynamically with specific configuration.
- retrieve plugin instances by ID or based on configuration properties.
- start or stop plugin instances.
- verify plugin instance configuration.

Creating plugins

Several methods are provided to allow on-the-fly creation of new plugins. Generally, they allow a plugin with a specific ID and plugin type (as a string or as a Groovy Class) to be created with optional configuration properties. These properties should match the fields in Plugins.

If any configuration properties are omitted, the defaults will be used as described in Setting Default Plugin Configuration. A boolean value is also returned indicating whether the creation succeeded or not.

Note that the `createPlugin` methods will initialize the plugin for retrieval or usage and attempt to start the plugin if the `autoStart` property is true.

Retrieving plugins

Plugins may be retrieved by using an ID, querying by plugin type, or even querying based on configuration parameters. Several methods are provided to perform these functions as shown on Plugin Control Service.

Starting and stopping plugins

Plugins may also be started or stopped on demand. These two methods are exposed directly as `start` and `stop` on the plugin control service. Although each method does not return any data, exceptions are thrown if errors are encountered.

Verifying plugin configuration

Finally, the plugin control service may be used to verify plugin configuration at any point instead of just when the plugin is started or modified. This may be useful to attempt to modify plugin configuration directly through the `setConfig` dynamic method (see Dynamic Methods) and then verify that the new configuration is valid for the plugin. Exceptions are thrown if the plugin or the configuration is invalid.

Examples

If an error state is detected it may be necessary to stop the current plugin instance until corrective action can be taken. This may be done using the following code:

```
package example

import com.adaptc.mws.plugins.*

class ErrorPlugin {
       IPluginControlService pluginControlService

       public void poll() {
               // Error is detected, stop plugin instance!
               try {
                       log.warn("An error was detected, trying to stop the plugin ${id}")
                       pluginControlService.stop(id)
                       log.warn("The plugin was successfully stopped")
               } catch(PluginStopException e) {
                       log.error("Plugin instance ${id} could not be stopped", e)
               }
       }
}
```

Related Topics

- Plugin Developer's Guide

Accessing MWS REST Resources

Often a plugin type may need to access existing MWS REST Resources in order to extend or complement default MWS functionality. This may be done with the **Moab REST Service**, which allows a plugin type developer to utilize the existing Resources documentation see Resources Introduction) to perform these tasks.

All accesses to resources require a HTTP method to use (such as GET, POST, PUT, or DELETE) and a relative URL (such as `/rest/jobs`). Although it mimics the REST resource interface, no actual requests are made and no data is transmitted through the network.

Authentication

All resources are available to the Moab REST Service, and no authentication or Application Accounts are needed.

> Caution must be used when developing plugin types, as there are no restrictions to what may be done with the Moab REST Service. This is especially true when not utilizing hooks as discussed below.

Hooks

If pre and post-processing hooks are utilized in MWS (Pre- and Post-Processing Hooks), the plugin type developer may choose whether or not they are executed when performing a "request" through the Moab REST service. This is done through the hooks option as documented in Moab REST Service.

Verifying API version support

The Moab REST Service provides a method for easily determining which API versions are supported by the current version of MWS. This method includes checks to make sure that the API version will work as expected, including verifying any configuration or external services are running.

```
moabRestService.isAPIVersionSupported(1)
moabRestService.isAPIVersionSupported(2)
```

Converting string dates

Because the Moab REST Service returns data exactly as given to an external consumer of MWS, including dates converted to strings, the service provides a method for converting MWS date strings to actual Date objects.

```
moabRestService.convertDateString("2011-11-08 13:18:47 MST")
```

URL parameters

URL parameters, such as `query`, `sort`, `proxy-user`, and others should be not be appended directly to the URL. Instead, these may be specified with the `params` option:

```
// Query images that are hypervisors
moabRestService.get("/rest/images", params:[query:'{"hypervisor":true}'])
// Sort images by osType
moabRestService.get("/rest/images", params:[sort:'{"osType":1}'])
```

Examples

This code retrieves a list of all nodes, and is equivalent to the **Get All Nodes** task.

```
package example

import com.adaptc.mws.plugins.*
import net.sf.json.*

class RestPlugin {
       IMoabRestService moabRestService

       public void poll() {
               def result = moabRestService.get("/rest/nodes")
               // OR with the hook enabled…
               def result = moabRestService.get("/rest/nodes", hooks:true)

               assert result instanceof MoabRestResponse
               assert nodes instanceof List

               log.debug("Nodes list:")
               nodes.each { JSON node ->
                       log.debug(node.id)
               }
       }
}
```

This code adds a flag to a job, and is equivalent to the **Modify Job Attributes** task. This request also enables the hook (if one is configured) for the "request" and uses a URL parameter. This is the equivalent of making a call to `/rest/jobs/job.1?proxy-user=adaptive`.

```
package example

import com.adaptc.mws.plugins.*
import net.sf.json.*

class RestPlugin {
       IMoabRestService moabRestService

       public void poll() {
               def jobId = "job.1"
               def result = moabRestService.put("/rest/jobs/"+jobId, hooks:true, params:['proxy-
user':'adaptive']) {
                       [flags:["RESTARTABLE"]]
               }
               assert result.isSuccess()
       }
}
```

Related Topics

- Plugin Developer's Guide

Creating Events and Notifications

Plugins may easily create new events and create or update notification conditions using the **Plugin Event Service**. Previously, this was only possible by utilizing the MWS REST resources. The event service eases this burden from

plugin developers. There are several operations that are available using the service:

- Create a event with or without specifying an event date.
- Create an event from a enumeration annotated with `EventEnumeration` (see Plugin Event Service) with or without specifying an event date.
- Create or update a notification condition with or without specifying an observed date or expiration duration.

Creating Events

Events are composed of several properties such as arguments, associated objects, origin, message, severity, escalation level, and a unique event code. The plugin event service removes the need for magic strings such as those for event severity ("INFO", "WARN", "FATAL") and also handles creating unique event codes. In other words, no bitwise manipulation is required to create new events.

The event code is comprised of several elements:

Code element	Description
Severity	If the event is informational, a warning, an error, or fatal.
Escalation level	Who cares about the event, or who should act on the event.
Component code	Internally made up of the MWS component code (stored internally) and the plugin event component code (see Plugin event component code).
Entry code	The code representing a unique event for the component (for each plugin event component code).

The plugin event service handles the severity, escalation level, and entry code portions of the code by the values passed as parameters to the `createEvent` method. The plugin event component code is described in the next section.

Plugin event component code

The plugin event component code should be a unique number across all plugin types or projects from `1-254`. This number is combined with the MWS component code to represent each plugin as a unique component code across all Adaptive Computing products. `0` is reserved for MWS itself and should not be used. `255` is reserved for plugin types that do not define an event component code and represents an "unknown" plugin component. Additionally, codes `1-150` are reserved for Adaptive Computing plugins, while `151-254` are reserved for Professional Services and/or customer-specific plugins.

This code may be specified by setting an `eventComponent` property (see Fields: Plugin Types) on the plugin project file or as a static property on the plugin type. As with all other project properties, the plugin type value overrides the project value. For example:

```
class MyExampleProject {
       …
       Integer eventComponent = 2
       …
}
ExamplePlugin {
       static final eventComponent = 1
       …
}
Example2Plugin {
       // no eventComponent property
       …
}
```

In this case, the plugin type `ExamplePlugin` has a plugin event component code of `1`, while the `Example2Plugin` has a code of `2` since it inherits it from the project properties.

Origin suffix

The origin of an event created through the plugin event service is automatically set by the plugin framework to `MWS/plugins/<plugin type>/<plugin id>`. For example, an event created by the plugin created from the "ExamplePlugin" plugin type with an ID of "plugin1" would generate events with an origin of `MWS/plugins/Example/plugin1`.

While this origin is sufficient for an administrator to determine the plugin where the event came from, the plugin developer may want this to be more specific to a class name or method name. This may be done using the optional `originSuffix` parameter to the `createEvent` method. The origin suffix, as its name implies, is appended to the end of the generated origin. For the example above, suppose the plugin developer passed `myMethod/switch1` as the origin suffix parameter when creating a new event. The event would then have an origin of `MWS/plugins/Example/plugin1/myMethod/switch1`.

Event enumerations

While creating events using the plugin event service is quite simple, often there are related events that have properties in common, such as the event type prefix or the origin suffix. Additionally, i18n messages (see i18n Messaging) are typically used for the event's message. Using the `EventEnumeration` annotation (see Plugin Event Service) in combination with a enumeration simplifies this process. When this is done, each message is pulled from the `messages.properties` files using a standard convention, and the event type prefix and the origin suffix may optionally added as static properties on the enumeration. Using `EventEnumeration` requires:

- The annotated element is an `enum`, not a `class` or `interface`.
- Each enumeration value must use the constructor with three arguments: the event name, the severity, and the escalation level.
- If an event type prefix is specified, it must be defined as "static String EVENT_TYPE_PREFIX = ...", otherwise the property should not be defined.
- If an origin suffix is specified, it must be defined as "static String ORIGIN_SUFFIX = ...", otherwise the property should not be defined.

If any of these conditions are not fulfilled, using the `EventEnumeration` annotation will result in compilation errors.

Enumeration values are automatically marked as implementing the `IPluginEvent` interface and may be used as the first parameter of the `createEvent` method on the plugin event service. For example:

```
package example

import com.adaptc.mws.plugins.EventEnumeration
import com.adaptc.mws.plugins.IPluginEventService.AssociatedObject
import static com.adaptc.mws.plugins.IPluginEventService.Severity.*
import static com.adaptc.mws.plugins.IPluginEventService.EscalationLevel.*

public class ExamplePlugin {
       void poll() {
               // Event 1 takes no arguments
               pluginEventService.createEvent(ExampleEvents.EVENT1, null, null)
               // Event 2 takes one argument and has an associated object
               pluginEventService.createEvent(ExampleEvents.EVENT2, ["arg1"], [new AssociatedObject
(type:"type1", id:"id1")])
       }
}

@EventEnumeration
enum ExampleEvents {
       EVENT1("Example One", INFO, USER),      // Entry code is 0
       EVENT2("Example Two", INFO, USER)       // Entry code is 1
}
```

It may be noted that several key properties of events are missing from the enumeration definition and create event call parameters:

- Message: retrieved automatically from i18n messages (see Messages for event enumerations)
- Event type: generated from the enumeration constructor and optional event type prefix property (see Event type for event enumerations)
- Entry code: generated from the return value of `ordinal()` on the enumeration value; in other words, this is generated from the order of the enumeration values

Messages for event enumerations

The message for events created from enumerations is generated using i18n messages (see i18n Messaging) with codes in the following format:

- `<enumeration type name>.<enumeration value name>.message`
- `<enumeration type name>.<enumeration value name>.comment`

Considering the example in the section above, the message for `ExampleEvents.EVENT1` would be generated using the argument list passed to the `createEvent` method with the "ExampleEvents.EVENT1.message" message from `messages.properties`. This message should contain arguments if needed, such as "My example with ID {0} was created" and is used as the "message" property in the created event. The comment, on the other hand, is not persisted with the event and should be text (typically in paragraph format) describing why the event typically occurs or what actions should be taken when it does occur. Consider the message to contain instance specific information for the event (passed as arguments to the message) and the comment to be general documentation concerning the event.

As a best practice, name event enumeration values using the number and short name of each argument to the message. This makes it easy for the consumer to know which arguments are expected and what each means. For example, if an event is for connection errors and needs two arguments to the message, the URL and the error message, the enumeration value should be named "CONNECT_FAILURE_1URL_2ERROR" or even "CONNECT_TO_1URL_FAILURE_2ERROR". In this way, the consumer knows that the first argument represents the URL and the second is the error message.

Event type for event enumerations

As described above, the static string field `EVENT_TYPE_PREFIX` may be defined on the enumeration. This value is optional and, when present, is prepended with a space to the event name parameter from the constructor to generate the event type. For example, consider the following enumeration:

```
package example

import com.adaptc.mws.plugins.EventEnumeration
import static com.adaptc.mws.plugins.IPluginEventService.Severity.*
import static com.adaptc.mws.plugins.IPluginEventService.EscalationLevel.*

@EventEnumeration
enum MyPluginEvents {
       CONNECT("Connect", INFO, ADMIN),
       DISCONNECT("Disconnect", INFO, ADMIN)

       static String EVENT_TYPE_PREFIX = "My Plugin"
}
```

If `MyPluginEvents.CONNECT` and `MyPluginEvents.DISCONNECT` were used with the plugin event service, the generated event types would be "My Plugin Connect" and "My Plugin Disconnect" respectively.

Origin for event enumerations

The origin for event enumeration values automatically contains more information than those for non-enumerated events, such as those described above. The enumeration type name and value are appended to the origin. For example, consider the following enumeration and plugin fragment:

```
…
class ExamplePlugin {
        …
        assert id=="example1"   // plugin ID is example1
        pluginEventService.createEvent(ExampleEvents.EVENT1, null, null)
        …
}
…
@EventEnumeration
enum ExampleEvents {
        EVENT1("Event One", INFO, ADMIN)
        ...
```

The origin generated for the created event would be `MWS/plugins/Example/example1/ExampleEvents/EVENT1`. The static string field `ORIGIN_SUFFIX` may also be defined on the enumeration. This value is optional and, when present, is appended to the end of the generated origin as described above with the origin suffix parameter to the `createEvent` method.

Example

In order to understand all interactions when event enumerations are used, the following is a complete example.

```
Plugin type
-----------------------------------

package example
import com.adaptc.mws.plugins.*

class ConnectPlugin extends AbstractPlugin {
        static eventComponent = 1

        IPluginEventService pluginEventService

        void poll() {
                def errorMessage = connect()
                if (errorMessage)
                        pluginEventService.createEvent(ConnectEvents.CONNECT_TO_1URL_FAILURE_2ERROR,
[config.url, errorMessage], null)
                else
                        pluginEventService.createEvent(ConnectEvents.CONNECT_SUCCESS, null, null)
        }

        // Returns the error message or null/empty on success
        private String connect() {
                String errorMessage
                …
                return errorMessage
        }
}
```

```
Event enumeration
-----------------------------------

package example
import com.adaptc.mws.plugins.EventEnumeration
import static com.adaptc.mws.plugins.IPluginEventService.Severity.*
import static com.adaptc.mws.plugins.IPluginEventService.EscalationLevel.*

@EventEnumeration
enum ConnectEvents {
       CONNECT_SUCCESS("Success", INFO, ADMIN),
       CONNECT_TO_1URL_FAILURE_2ERROR("Failure", ERROR, ADMIN)

       static String EVENT_TYPE_PREFIX = "Connect"
}
```

```
messages.properties
-----------------------------------

ConnectEvents.CONNECT_SUCCESS.message=The plugin was successfully connected!
ConnectEvents.CONNECT_SUCCESS.comment=This occurs when the plugin successfully
connects to the configured URL and
       is informational only.
ConnectEvents.CONNECT_TO_1URL_FAILURE_2ERROR.message=The plugin failed to connect to
{0}: {1}
ConnectEvents.CONNECT_TO_1URL_FAILURE_2ERROR.comment=This occurs when the plugin fails
to connect to the configured
       URL for any reason.  The most common reason is that the service is not running and
needs to be started.
```

The following are examples of the events created in MWS:

```
Created events
-----------------------------------

{"totalCount": 2, "resultCount": 2, "results": [
       {
              "arguments": ["http://localhost:1000", "The service is not running!"],
              "code": 570523649,
              "eventDate": "2013-06-12 19:16:50 UTC",
              "eventType": "Connect Failure",
              "message": "The plugin failed to connect to http://localhost:1000: The service is
not running!",
              "origin": "MWS/plugins/Connect/connect/ConnectEvents/CONNECT_TO_1URL_FAILURE_
2ERROR",
              "severity": "ERROR",
              "id": "51b8c922a816c6a04af2401d",
              "associatedObjects": []
       },
       {
              "arguments": [],
              "code": 33652736,
              "eventDate": "2013-06-12 19:18:07 UTC",
              "eventType": "Connect Success",
              "message": "The plugin was successfully connected!",
              "origin": "MWS/plugins/Connect/connect/ConnectEvents/CONNECT_SUCCESS",
              "severity": "INFO",
              "id": "51b8c96fa816c6a04af24021",
              "associatedObjects": []
       }
]}
```

Unique event codes

The last topic that must be covered in creating events from plugins is that all efforts should be made to make sure that event codes are unique throughout all Adaptive Computing product suites. Additionally, the codes should be static, meaning they do not change once established. In order to do this, adhere the following recommendations:

- Use a unique (across all plugin types) plugin event component code for each plugin type.
- Follow the guidelines for plugin event component codes established above (see Plugin event component code) and ensure it is a number 1-254.
- Use event enumerations where possible, otherwise ensure (through testing if possible) that all entry codes are unique for each plugin type.
- Ensure (through testing if possible) that the ordinal value of the event enumeration values do not change.

Creating or Updating Notification Conditions

The plugin event service also makes it easy to create or update notification conditions. Simply use the `updateNotificationCondition` method. Just as the MWS notification condition resource, this is an idempotent operation, meaning it can be called multiple times with the same result. If the notification condition does not exist, it will be created automatically. If it does exist, the observed date and details will be updated accordingly.

Examples

Examples are available on Plugin Event Service.

Related Topics

- Resources Introduction
- Events
- Notifications
- Notification Conditions
- Plugin Developer's Guide
- Fields: Events
- Plugin Event Service
- Handling Events
- System Events
- Securing the Connection with the Message Queue

Handling Events

Plugin events (excepting the poll event) are currently in Beta. Interfaces may change significantly in future releases.

Plugin types may handle specific events by containing methods defined by the conventions below. All events are optional.

The polling event

To maintain current information, each plugin is polled at a specified time interval. The following method definition is required to utilize the polling event.

```
void poll() { … }
```

Typically this polling method is used to report node and virtual machine information. By default, the polling interval is set to 30 seconds, but can be modified for all or individual plugins as explained in Plugin Management.

When a polling event occurs, the `poll` method on the target plugin is called. This method may perform any function desired and should typically make calls to the **Node RM Service**, the **Virtual Machine RM Service**, and the **Job RM Service** services to report the current state of nodes and virtual machines. For example, the `poll` method in the Native plugin type is implemented as follows:

This is an extremely simplified version of what is actually implemented in the Native plugin type.

```
INodeRMService nodeRMService;
IVirtualMachineRMService virtualMachineRMService;

public void poll() {
       nodeRMService.save(getNodes());
       virtualMachineRMService.save(getVirtualMachines());
}
```

This simple poll method calls two other helper methods called `getNodes` and `getVirtualMachines` to retrieve node and virtual machine reports. These reports are then sent to the appropriate RM service. See Reporting State Data for more information on the RM services; however, the objective of this example is to demonstrate one possible use of the poll event handler. Other plugin types, on the other hand, may use the poll event to update internal data from pertinent resources or make calls to external APIs.

Lifecycle events

Events are also triggered for certain lifecycle state changes. The following method definitions are required to receive lifecycle events.

```
public void configure() throws InvalidPluginConfigurationException { … }
public void beforeStart() { … }
public void afterStart() { … }
public void beforeStop() { … }
public void afterStop() { … }
```

Each event is described in the table below with the associated state change when the event is triggered.

State change	Event	Description
configure	Configure	Triggered before `beforeStart` and after the plugin has been configured. May be used to verify configuration and perform any setup needed any time configuration is loaded or modified.
beforeStart	Start	Triggered just before starting a plugin.
afterStart	Start	Triggered just after a plugin has been started.
beforeStop	Stop	Triggered just before stopping a plugin.
afterStop	Stop	Triggered just after stopping a plugin.

Currently, no events are triggered for pausing, resuming, erroring, or clearing errors for plugins.

RM events

When MWS is configured as a Moab Resource Manager (see Moab Workload Manager Resource Manager Integration, and more specifically, Configuring Moab Workload Manager), RM events are sent from Moab to each plugin according to the routing specification (see Routing). The following method definitions are required to receive these events.

```
public boolean jobCancel(String jobName) { … }
public boolean jobModify(String jobName, Map<String, Object> attributes, ModifyMode
modifyMode) { … }
public boolean jobRequeue(String jobName) { … }
public boolean jobResume(String jobName) { … }
public boolean jobStart(String jobName, List<String> nodes, String username) { … }
public boolean jobSubmit(Map<String, Object> job, String submissionString, String
submissionFlags) { … }
public boolean jobSuspend(String jobName) { … }
public boolean nodeModify(List<String> nodes, Map<String, String> attributes,
ModifyMode modifyMode) { … }
public boolean nodePower(List<String> nodes, NodeReportPower state) { … }
public boolean virtualMachinePower(List<String> virtualMachines, NodeReportPower
state) { … }
```

Related Topics

- Events
- Notifications
- Notification Conditions
- Plugin Developer's Guide
- Fields: Events
- Resources Introduction
- Plugin Event Service
- Creating Events and Notifications

Handling Exceptions

Plugin exceptions are currently in Beta. Interfaces may change significantly in future releases.

The `com.adaptc.mws.plugins` package contains several exceptions that may be used and in some cases, should be caught. All exceptions end with "Exception", as in PluginStartException.

There are several specific cases where Exceptions should or can be used:

- The `reload` method on the **Plugin Control Service** can throw the InvalidPluginConfigurationException to signify that the configuration contains errors.
- Various methods on the **Plugin Control Service** throw plugin exceptions which must be caught to diagnose errors when creating plugin types.
- Any exception (including the Exception class) can be thrown from a custom web service to display a 500 Internal Server Error to the client requesting the service with the given error message.

Related Topics

- Plugin Developer's Guide

Managing SSL Connections

At times it is desirable to load and use self-signed certificates, certificates generated from a single trusted certificate authority (CA), or even simple server certificates. It may also be necessary to use client certificates to communicate with external resources. To ease this process, the **SSL Service** may be utilized. This service provides methods to load client and server certificates from the filesystem. Methods are also present to aid in creating connections which automatically trust all server certificates and connections.

Several points should be noted when using the SSL Service:

- Certificate files may be in the PEM file format and do not need to be in the DER format (as is typical of Java security).
- Each method returns an instance of SSLSocketFactory, which may then be used to create simple sockets or, in combination with another client library of choice, create a connection.
- If the client certificate password is non-null, it will be used to decrypt the protected client certificate.
- This service is *not* needed when performing SSL communications with trusted certificates, such as those for HTTPS enabled websites that do not have a self-signed certificate.
- If the file name of the certificate file (client or server) is relative (no leading '/' character), it will be loaded from the `mws.certificates.location` configuration parameter (see Configuration).
    - The default value of `mws.certificates.location is MWS_HOME/etc/ssl.crt`.
- Both the client certificate alias and password may be `null`. In this case, the client certificate must not be encrypted and the client certificate's default alias (the first subject CN) will be used.
- The lenient socket factory and hostname verifier automatically trust all server certificates. Because of this, they present a large security hole. Only use these methods in development or in fully trusted environments.

Example

To create a socket to a server that requires a client certificate, the following code may be used.

```
package example

import com.adaptc.mws.plugins.*

class SSLConnectionPlugin extends AbstractPlugin {
       ISslService sslService

       public void poll() {
               // This certificate is not encrypted and will be the only certificate presented to
the
               // connecting end of the socket.
               // This file will be loaded from MWS_HOME + mws.certificates.location + my-cert.pem.
               String clientCert = "my-cert.pem"

               def socketFactory = sslService.getSocketFactory(clientCert, null, null)
               def socket = socketFactory.createSocket("hostname.com", 443)
               // Write and read from the socket as desired…
       }
}
```

To create a HTTPS URL connection to a server that has a self-signed certificate, the following code may be used. Note that this is very typical of client libraries –

they have a method to set the SSL socket factory used when creating connections.

```
package example

import com.adaptc.mws.plugins.*

class SSLConnectionPlugin extends AbstractPlugin {
       ISslService sslService

       public void poll() {
                // This certificate represents either the server public certificate or the CA's
certificate.
                // Since the path is absolute it will not be loaded from the MWS_HOME directory.
                String serverCert = "/etc/ssl/certs/server-cert.pem"

                def socketFactory = sslService.getSocketFactory(serverCert)

                // Open connection to URL
                HttpsURLConnection conn = "https://hostname.com:443/test".toURL().openConnection()
                conn.setSSLSocketFactory(socketFactory)

                // Retrieve page content and do with as desired…
                def pageContent = conn.getInputStream().text
       }
}
```

Related Topics

- Plugin Developer's Guide

Utilizing Services or Custom "Helper" Classes

There are three general types of services available for use in plugins:

- Bundled services such as the **Moab REST Service**.
- Custom built translators loaded by convention of their name.
- Other custom built helper classes registered with Annotations.

These will each be described in this section.

Bundled Services

Bundled services are utility classes that are included and injected by default onto all plugin types. It is not required to use any of these services, but they enable several core features of plugin types as discussed in Utility Services.

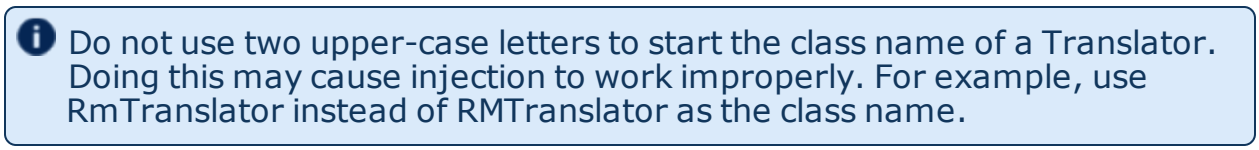
More information may be found on each bundled service in Plugin Services.

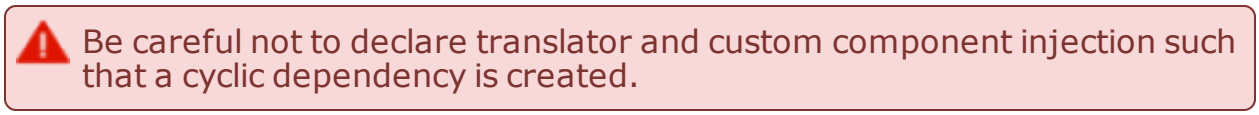
Using Translators

Often a plugin type class file becomes so complex that it is desirable to split some of its logic into separate utility service classes. The most typical use case

for this is to split out the logic for "translating" from a specific resource API to a format of data that the plugin type can natively understand and utilize. For this reason, there is a convention defined to easily add these helper classes called "Translators."

Simply end any class name with "Translator," and it will be automatically injected just as bundled services onto plugin types, other translators, or even custom registered components. The injection occurs only if a field exists on the class matching the name of the translator with the first letter lower-cased. For example, a translator class called "MyTranslator" would be injected on plugin types, other translators, and custom components that define a field called "myTranslator" as `def myTranslator` or `MyTranslator myTranslator`.





Logging in translators

All translators automatically have a "getLog" method injected on them which can be used to access the configured logger. It returns an instance of org.apache.commons.logging.Log.

```
package example

class ExampleTranslator {
       public void myMethod() {
               // log will be translated to getLog() by the groovy compiler
               log.info("Starting my method")
       }
}
```

See Logging for more information on logging configuration and usage.

Example

Suppose that a translator needs to be created to handle a connection to access an external REST resource. The translator could be defined as follows:

```
package example

class ExampleTranslator {
       public int getExternalNumber() {
               def number = … // Make call to external resource
               return number
       }
}
```

A plugin type can then use the translator by defining a field called "exampleTranslator". Note that an instance does not need to be explicitly created.

```
package example

class ExamplePlugin {
       def exampleTranslator
       // OR …
       //ExampleTranslator exampleTranslator

       public void poll() {
               // Use the translator
               log.info("The current number is "+exampleTranslator.getExternalNumber())
       }
}
```

To extend the example, the translator may also be injected into another translator:

```
package example

class AnotherTranslator {
       def exampleTranslator

       public int modifyNumber(int number) {
               return number + exampleTranslator.getExternalNumber()
       }
}
```

This translator may be used in the plugin type just as the other translator.

Registering Custom Components

There are cases where the concept of a "Translator" does not fit the desired use of a utility class. In these cases, it is possible to register any arbitrary class as a component to be injected just as a translator would be. This is done using the Spring Framework's annotation `org.springframework.stereotype.Component`. When this annotation is used, the class is automatically registered to be injected just as translators onto plugin types and translators.

> All annotations are available in the dependencies declared by the plugins-commons artifact.

> Do not use two upper-case letters to start the class name of a custom component. Doing this may cause injection to work improperly. For example, use RmUtility instead of RMUtility as the class name.

Changing scope

By default, when a custom component is injected, only a single instance is created for all classes which inject it. This is referred to as the 'singleton' scope.

Another scope that is available is 'prototype', which creates a new instance every time it is injected. This is useful when the class contains state data or fields that are modified by multiple methods. To change the scope, use the `org.springframework.context.annotation`.Scope on the class with a single String parameter specifying "singleton" or "prototype."

Injecting translators or components

The need may arise to inject translators or other custom components onto custom components. This is done using the `org.springframework.beans.factory.annotation.Autowired` or `javax.annotation.Resource` annotations. The `Autowired` annotation is used to inject class instances by the type (i.e. `MyTranslator myTranslator`) while the `Resource` annotation is used to inject class instances by the name (i.e. `def myTranslator`). Add the desired annotation to the field that needs to be injected.

Note that using the `Autowired` annotation does injection by type which differs from translator and plugin type injection. These are done by name just as the `Resource` annotation allows. Due to this fact, a type of "def" cannot be used when doing injection onto custom components using the `Autowired` annotation. See the example below.

Injection of custom components *onto* translators and plugin types are still done by name, only fields injected using the `Autowired` annotation are affected.

Be careful not to declare translator and custom component injection such that a cyclic dependency is created.

Logging in custom components

Unlike plugins and translators, custom components do *not* automatically have a "getLog" method injected on them. In order to log with custom components, you must use the Apache Commons Logging classes to retrieve a new log. The `PluginConstants` class contains the value of the logger prefix that is used for all plugins and translators. The following is an example of how to retrieve and use a logger correctly in a custom component.

```
package example

import com.adaptc.mws.plugins.PluginConstants
import org.apache.commons.logging.Log
import org.apache.commons.logging.LogFactory
import org.springframework.stereotype.Component

@Component
class ExampleComponent {
    private static final Log log = LogFactory.getLog(PluginConstants.LOGGER_
PREFIX+this.name)

       public void myMethod() {
               log.info("Starting my method")
       }
}
```

See Logging for more information on logging configuration and usage.

Example

Suppose that a custom utility class is needed to perform complex logic. A custom component could be defined as follows (notice the optional use of the `Scope` annotation):

```
package example

import org.springframework.stereotype.Component
import org.springframework.context.annotation.Scope

@Component
@Scope("prototype")
class ComplexLogicHandler {
       def handleLogic() {
               … // Perform complex logic and return
       }
}
```

A plugin type or translator could then be defined to inject this component:

```
package example

class CustomPlugin {
       def complexLogicHandler

       public void poll() {
               complexLogicHandler.handleLogic()
       }
}
```

Now suppose another custom component needs to use the ComplexLogicHandler in its code. It can inject it using the `Autowired` annotation:

```
package example

import org.springframework.stereotype.Component
import org.springframework.beans.factory.annotation.Autowired

@Component
class AnotherHandler {
       // Note that this is injected by type, so 'def' may not be used
       @Autowired
       ComplexLogicHandler complexLogicHandler

       def wrapLogic() {
              complexLogicHandler.handleLogic()
       }
}
```

To perform the same injection but by name (as translators and plugin types are injected), use the `Resource` annotation:

```
package example

import org.springframework.stereotype.Component
import javax.annotation.Resource

@Component
class AnotherHandler {
       // Note that this is injected by name based solely on the name defined in
       //      the annotation.  The name of the field itself does not affect the injection.
       @Resource(name="complexLogicHandler")
       def complexLogicHandler

       def wrapLogic() {
              complexLogicHandler.handleLogic()
       }
}
```

Related Topics

- Plugin Developer's Guide

Packaging Plugins

Plugin types may be packaged in two different ways to upload to MWS:

- A simple Groovy file containing a single plugin type definition.
- A JAR file containing one or more plugin types, translators, and custom components.

While each may be uploaded to MWS using the REST API or the User Interface as described in Add or Update Plugin Types, using a JAR file is recommended. Using a simple Groovy file is useful for testing and generating proof of concept work, but does not allow the use of several features of plugins.

The principles of packaging a plugin type or set of plugin types in a JAR file are very simple. Simply compile the classes and package in a typical JAR file. All

classes ending in "Plugin" are automatically attempted to be loaded as a plugin type, all classes ending in "Translator" are attempted to be loaded as a translator, and all classes annotated as a custom component will be attempted to be loaded. It is recommended that a build framework is used to help with compiling and packaging the JAR file, such as Gradle. This makes it easy to declare a dependency on the necessary JAR files used in plugin development and to debug, compile, and test plugin code.

In addition to using utility services such as translators, packaging plugin types in JAR files allows the creation of a single project for multiple related plugin types and bundling of external dependencies. These two features are discussed in the following sections.

Plugin Projects and Metadata

Each plugin type has information attached to it, called metadata, which describes the origin and purpose of the plugin type. Additionally, a JAR file may also contain a project file which defines default metadata attributes for all plugin types in the JAR. Initial plugins, or plugins that will be created on loading of the JAR file if they do not exist, are also able to be defined on a project file. In all cases, metadata declared on a plugin type will override the metadata defined on the project file.

To define a project file, simply add a class to JAR file that ends in "Project." This file will attempted to be loaded as the project file. Every field on a project file, and even the file itself, is optional. All available fields are shown in the example below.

```
class SampleProject {
       // Plugin information
       String title = "Sample"
       String description = "Sample plugin types"
       String author = "Our Company."
       String website = "http://example.com"
       String email = "sample@example.com"
       Integer eventComponent = 1
       // Versioning properties
       String version = "0.1"
       String mwsVersion = "7.1 > *"
       String commonsVersion = "0.9 > *"
       String license = "APACHE"

       // Documentation properties
       String issueManagementLink = "http://example.com/ticket-system/sample-plugins"
       String documentationLink = "http://example.com/docs/sample-plugins"
       String scmLink = "http://example.com/git/sample-plugins"

       // Plugins that are to be created with these properties only when they do NOT exist
       // This does not override any existing plugin instance configuration
       def initialPlugins = {
               /*
               // Multiple instances of plugins may be defined here.
               // In this case, 'sample' is the id of the plugin
               sample {
                               pluginType = "Sample"
                               // All properties except for "pluginType" are optional
                               pollInterval = 30
                               autoStart = true
                               // Although it is possible to set plugin precedence, it is not recomm
this precedence
                               // may already be taken and plugin creation will fail in this case
                               precedence = 5
                               config {
                                       configParam = "value"
                               }
                       }
               }
               // Another plugin with an ID of 'sample2'
               sample2 {
                       …
               */
       }
```

As can be seen, metadata information about the plugin type(s), versions, and documentation are available. These are displayed when viewing plugin information in the User Interface or through the REST API.

Any of these properties except for `initialPlugins`, `mwsVersion`, and `commonsVersion` may be overwritten by the plugin type class itself by using static properties. A simple example is shown below.

```
package example

class SamplePlugin {
       // Properties may be typed, untyped, final, or otherwise,
       // but they MUST be static
       static version = "0.2"
       static title = "Sample plugin"
       static description = "This sample plugin is used to demonstrate metadata information"
       static author = "Separate Division"
       static eventComponent = 1

       … // Rest of the plugin type definition
}
```

Event component

The `eventComponent` field is explored in Creating Events and Notifications.

MWS and commons versions

The `mwsVersion` and `commonsVersion` fields are used to restrict the versions of MWS and plugin framework with which the plugin project may be used. Each field is of the format `FIRST_VERSION > LAST_VERSION`, where `FIRST_VERSION` is the first supported MWS or plugin framework version (inclusive), and `LAST_VERSION` is the last supported MWS or plugin framework version (inclusive). Each version must take the format of #.# or #.#.#, as in 7.1, or 7.1.2. An asterisk (*) is used to denote any version, and may be used for the first or the last version.

Although support for restricting both the MWS and commons versions are provided, it is recommended to use the commons version restriction always and the MWS version restriction where necessary. Restrictions on the commons version prevent plugin loading errors while restrictions on the MWS version prevent runtime errors such as missing support for certain MWS API versions.

Typically the `mwsVersion` and `commonsVersion` fields are set as shown above, with the first version set to a specific number, and the last version set to any (an asterisk). This is the recommended approach for setting both fields. It is not recommended to use any version (asterisk) for the first version. Some examples of `mwsVersion` and `commonsVersion` values are shown below with explanations of how they behave.

```
String mwsVersion = "7.1 > *" // Any MWS version 7.1.0 and greater is supported
(including 7.2, etc)
String mwsVersion = "7.1.3 > *" // Any MWS version 7.1.3 and greater is supported
(including 7.2, etc)
String mwsVersion = "7.1 > 7.1.3" // Any MWS version between 7.1.0 and 7.1.3 is
supported
String mwsVersion = "* > *" // Any MWS version is supported (not recommended!)
String mwsVersion = "* > 7.2" // Any MWS version up to 7.2 is supported (not
recommended!)

String commonsVersion = "0.9 > *" // Any framework version 0.9.0 and greater is
supported (including 1.0, etc)
String commonsVersion = "0.9.3 > *" // Any framework version 0.9.3 and greater is
supported (including 1.0, etc)
String commonsVersion = "0.9 > 0.9.3" // Any framework version between 0.9.0 and 0.9.3
is supported
String commonsVersion = "* > *" // Any framework version is supported (not
recommended!)
String commonsVersion = "* > 1.0" // Any framework version up to 1.0 is supported (not
recommended!)
```

If the `mwsVersion` or `commonsVersion` fields are formatted incorrectly, the plugin project will fail to load. If a plugin project is uploaded to MWS and the version check fails, the project will fail to load with an error message about the `mwsVersion` or `commonsVersion`.

The `mwsVersion` and `commonsVersion` fields cannot be overridden by a single plugin type, but can be set only at the plugin project level. This prevents mixing of MWS and commons version requirements within a single project.

Initial plugins

The initial plugins closure provides the flexibility to insert plugin instances when the JAR is loaded. This occurs at two points: when the plugin JAR is first uploaded to MWS, and when MWS is restarted. As shown in the example above, the ID, pluginType, and other properties may be configured for multiple plugins.

The nature of Groovy closures means that programmatic definition of initial plugins is possible. This may even be based on the MWS application configuration. Two properties are automatically available in the initialPlugins closure:

- `appConfig` – Contains the MWS application configuration. Any configuration parameter is available for access as documented on Configuration.
- `suite` – Contains the currently configured suite that MWS is running in. This is equivalent to the `mws.suite` configuration parameter, and is an instance of Suite.

Native plugin case study

The Native JAR file utilizes many of the features discussed above. In the root of the JAR file, a compiled class called NativeProject exists which defines all of the metadata fields, including `initialPlugins`. Trying to create an initial plugin presents two distinct problems:

- The plugin should be initialized only if the suite is CLOUD.
- The plugin type configuration must contain an entry referencing the configured `mws.home.location` parameter, or the configured MWS_HOME location.

The `initialPlugins` closure is defined as follows:

```
import com.adaptc.mws.plugins.Suite

class NativeProject {
       … // Metadata fields

       def initialPlugins = {
               // Initialize the cloud-native plugin only if the suite is CLOUD
               if (suite==Suite.CLOUD) {
                       'cloud-native' {
                               pluginType = "Native"
                               pollInterval = 30
                               config {
                                       // Use the appConfig property to retrieve the current MWS_HOME
                                       getCluster = "file://${appConfig.mws.home.location}/etc/nodes.t
                               }
                       }
               }
       }
}
```

Managing External Dependencies

External dependencies (e.g. JAR files) may be included and referenced in JAR files. Certain rules must also be followed in order to have the dependencies loaded from the JAR file correctly:

The plugin type must bundle all external dependency JARs in the root of the plugin type JAR file.

An entry must be included in the `MANIFEST.MF` file that references each of these bundled JAR files as a space separated list:

```
Class-Path: dependency1.jar dependency2.jar dependency3.jar
```

Assuming that these rules are followed and that the plugin type is uploaded using the REST API or the User Interface, the dependent JARs will first be loaded and then the new plugin type and associated files will be loaded.

Documenting Plugin Types

Documentation may also be included in JAR files by placing one or more Markdown formatted files in the root of the project JAR file. These files will be processed dynamically by MWS and presented as documentation pages for the respective plugin types within the MWS plugin user interface pages. Markdown is a simple text-to-HTML format used in some of the most popular open-source repositories such as GitHub and BitBucket. To help provide plugin developers use a single place or file for documentation, the conventional use of "README.md" as documentation was followed within MWS.

Documentation file naming

Each documentation filename must start with "README" and end with ".md". If only one documentation file is needed for bundled plugin type(s), it is recommended to call the file "README.md". For multiple plugin types, the file name must contain the plugin type name without the "Plugin" suffix in the format of "README-<PluginName>.md". For example, if a plugin project JAR file contained the plugin type classes "MyPlugin", "ABTestPlugin", and "ImportantPlugin", the documentation files would be located in the root of the JAR file and would be called "README-My.md", "README-ABTest.md", and "README-Important.md" respectively. If a "README" file does not exist for a certain plugin type, the main "README.md" file (if provided) will be used as documentation for that plugin type.

Markdown syntax

The Markdown syntax supported by MWS is very close to GitHub Flavored Markdown. Internally, the pegdown Markdown processor is used to generate the HTML with the `TABLES`, `ABBREVIATIONS`, `FENCED_CODE_BLOCKS`, `SMARTYPANTS`, `DEFINITIONS`, and `QUOTES` extensions enabled. HTML tags may also be used directly in order to create more refined formatting of the documentation, but this is discouraged with the exception of inserting the configuration reference table discussed below.

For example, the `TABLES` extension may be used to easily create HTML tables:

```
Name | Notes
------ | -------
Bob    | Knows how to use MWS plugins but has never created one
George | Writes MWS plugins in his spare time
```

The only main difference from standard Markdown processors is that block quotes (marked by lines prepended with '> ') are shown as highlighted information boxes when displayed in MWS. This may be used to draw more attention to informational or warning messages without writing custom HTML.

```
 > **Warning:** The use of this plugin type requires that MWS and MWM are configured
correctly as described in
 > the MWS user guide.
```

Configuration reference table

A table of available configuration parameters is often constructed in documentation for each plugin type. To ease the burden on the plugin developer of maintaining this documentation and the constraints on the plugin type, a table generated from the constraints (see Configuration Constraints) and included messages is available by using the following HTML in the README file(s):

```
<div class="configuration-table">This section will be replaced by MWS with the
configuration parameters table</div>
```

The text within the `div` container may be anything, but should state something helpful such as that it is placeholder in cases where the documentation may be viewed within other contexts such as on GitHub.

The generated table includes the following columns for each configuration parameter listed in the constraints: name, key, required, type, description. The "name" and "description" values are retrieved from the "help" and "label" messages bundled in the plugin JAR (see the labels and help messages section in Configuration Constraints for more information).

Web services reference sections

Documentation for exposed web services (see Exposing Web Services) is also able to be generated automatically. Instead of a single table as done with configuration parameters, a section with several tables (possible URL access points, URL parameters, and response fields) and additional information is generated for each exposed web service. This is available by using the following HTML in the README file(s):

```
<div class="webservice-sections">This section will be replaced by MWS with the web
service documentation</div>
```

The text within the `div` container may be anything, but should state something helpful such as that it is placeholder in cases where the documentation may be viewed within other contexts such as on GitHub.

Changing heading sizes

The generated sections each begin with an `<h2>` heading with the name of the web service. If a different heading size (`h3`, `h4`, etc.) is desired, this may be done with the following HTML:

```
<div class="webservice-sections" data-level="3">This section will be replaced by MWS
with the web service documentation</div>
```

Notice the `data-level` attribute, which contains the number used in the HTML `h` tag.

Message codes

Just as with the configuration table, the data for the content is generated automatically from the web service method name and from i18n messages (see i18n Messaging) bundled in the plugin JAR file. Message codes are available to customize the label and description of the web service. Codes are also available to define an arbitrary number of URL parameters and response fields. These do not need to be defined, but are helpful. The following table defines each message used in generating the documentation for web services.

Name	Message code	Description
Web Service Label	<pluginType>.webServices.<webServiceMethod>.label	The label used as the heading for the section, defaults to the naturally capitalized method name if not present.
Web Service Description	<pluginType>.webServices.<webServiceMethod>.help	Paragraph text describing the web service and its functionality, outputs, etc.
Parameter Key	<pluginType>.webServices.<webServiceMethod>.parameter<n>.key	The `n`th URL parameter, starting at 1 (example: `id`).
Parameter Label	<pluginType>.webServices.<webServiceMethod>.parameter<n>.label	The label for the `n`th URL parameter, defaults to the naturally capitalized key if not present.
Parameter Type	<pluginType>.webServices.<webServiceMethod>.parameter<n>.type	The type for the `n`th URL parameter, defaults to `String` if not present.
Parameter Description	<pluginType>.webServices.<webServiceMethod>.parameter<n>.help	The description or help text for the `n`th URL parameter.
Response Field Key	<pluginType>.webServices.<webServiceMethod>.return<n>.key	The nth response field, starting at 1 (example: `success`).

Name	Message code	Description
Response Field Label	<pluginType>.webServices.<webServiceMethod>.return<n>.label	The label for the nth response field, defaults to the naturally capitalized key if not present.
Response Field Type	<pluginType>.webServices.<webServiceMethod>.return<n>.type	The type for the nth response field, defaults to `String` if not present.
Response Field Description	<pluginType>.webServices.<webServiceMethod>.return<n>.help	The description or help text for the nth response field.

As an example, suppose that a web service method called "doSomething" exists on a plugin type named "MyExamplePlugin". This web service expects two URL parameters: `id`, an integer, and `action`, a string. The response body consists of a JSON object with two fields: `success`, a boolean value, and `messages`, a list of strings. The following messages would serve to generate helpful documentation:

```
messages.properties
-----------------------------------

# web service messages
myExamplePlugin.webServices.doSomething.label=Do Something Important
myExamplePlugin.webServices.doSomething.help=This web service does something important
with the input parameters.
# parameters
myExamplePlugin.webServices.doSomething.parameter1.key=id
myExamplePlugin.webServices.doSomething.parameter1.label=ID
myExamplePlugin.webServices.doSomething.parameter1.type=Integer
myExamplePlugin.webServices.doSomething.parameter1.help=The identifier of an object
myExamplePlugin.webServices.doSomething.parameter2.key=action
myExamplePlugin.webServices.doSomething.parameter2.label=Action # same as the default
would be
myExamplePlugin.webServices.doSomething.parameter2.type=String # same as the default
would be
myExamplePlugin.webServices.doSomething.parameter2.help=The action to perform

# response fields
myExamplePlugin.webServices.doSomething.return1.key=success
myExamplePlugin.webServices.doSomething.return1.label=Success # same as the default
would be
myExamplePlugin.webServices.doSomething.return1.type=Boolean
myExamplePlugin.webServices.doSomething.return1.help=True if the request succeeded,
false otherwise
myExamplePlugin.webServices.doSomething.return1.key=messages
myExamplePlugin.webServices.doSomething.return1.label=Error Messages
myExamplePlugin.webServices.doSomething.return1.type=List of Strings
myExamplePlugin.webServices.doSomething.return1.help=Error messages describing the
reason why success is false.
```

Note that if the *first* URL parameter key is `id`, the listed resource URLs will include the optional URL with the `id` parameter inline, such as `/rest/plugins/<pluginId>/services/<webService>/<id>`. Therefore, it is recommended to use `id` as parameter `1` if the web service expects a parameter with that key.

Related Topics

- Plugin Developer's Guide

Example Plugin Types

Several plugin types are provided by Adaptive Computing for use in Moab Web Services. Examples of these include the Native and vCenter plugin types.

A sample plugin type in Groovy would resemble the following:

```
package sample

import com.adaptc.mws.plugins.*

class SamplePlugin extends AbstractPlugin {
       static author = "Adaptive Computing"
       static description = "A simple plugin in groovy"
       static version = "0.1"

       INodeRMService nodeRMService

       public void configure() throws InvalidPluginConfigurationException {
               def myConfig = config   // "config" is equivalent to getConfig() in groovy
               def errors = []
               if (!myConfig.arbitraryKey)
                       errors << "Missing arbitraryKey!"
               if (errors)
                       throw new InvalidPluginConfigurationException(errors)
       }

       public void poll() {
               NodeReport node = new NodeReport("node1")
               node.resources.RES1.total = 5
               node.resources.RES1.available = 5
               node.state = NodeReportState.IDLE
               nodeRMService.save([node])
       }

       // Access at /rest/plugins/<id>/services/example-service
       public def exampleService(Map params) {
               return [success:true]
       }
}
```

Related Topics

- Plugin Developer's Guide

Moab Workload Manager Resource Manager Integration

Moab Workload Manager possesses the concept of Resource Managers (RMs). While plugins can be related to RMs, they often provide greater functionality and serve more purposes than a typical RM. MWS must be represented in Moab as a RM to enable certain plugin features such as state reporting and handling RM events. This section describes the process of configuring Moab and additional details of its queries to MWS. It includes the following topics:

- Configuring Moab Workload Manager
- Resource Manager Queries

Related Topics

- About Moab Web Services Plugins

Configuring Moab Workload Manager

Moab Workload Manager must be configured to use MWS as a resource manager. Do the following:

- First, the following lines must be in the Moab Workload Manager configuration file or one of its included files:

```
RMCFG[mws]                          TYPE=MWS
RMCFG[mws]                          FLAGS=UserSpaceIsSeparate
RMCFG[mws]                          BASEURL=http://localhost:8080/mws
```

*The **BASEURL** must match the configured URL of MWS.*

- The next step is to edit the MWS credential information in the Moab private configuration file (`/opt/moab/etc/moab-private.cfg`, by default). Here are the default values:

```
CLIENTCFG[RM:mws] USERNAME=moab-admin PASSWORD=changeme!
```

USERNAME and **PASSWORD** must match the values of `auth.defaultUser.username` and `auth.defaultUser.password`, respectively, found in the MWS configuration file. The MWS RM contacts MWS directly using the base URL, username, and password configured.

Optionally, the **USERNAME** and **PASSWORD** configuration values may be specified directly in the Moab configuration file, though this is not recommended. Likewise, the **BASEURL** configuration value can be specified in the Moab private configuration file.

- Lastly, to enable such actions as submitting jobs as different users, the **ENABLEPROXY**=*TRUE* option must be present in the **ADMINCFG** configuration line, and the **OSCREDLOOKUP** option must be set to *NEVER*, as follows:

```
ADMINCFG[1]             USERS=root      ENABLEPROXY=TRUE
OSCREDLOOKUP            NEVER
```

- You may also want to configure SSL by using the following options (in either the RMCFG or CLIENTCFG section):
  - **SSLCACERT**: Lets you specify the absolute path to your SSL CA certificate. (This also enables the use of self-signed certificates, if desired.) It is recommended that you set this option in the Moab private configuration file. For example:

```
CLIENTCFG[RM:mws]    SSLCACERT=/path/to/cert.pem
```

  - **SSLNOHOSTCHECK**: Lets you disable the SSL check to make sure that the actual server name matches the certificate's server name. For example:

```
#In moab-private.cfg
CLIENTCFG[RM:mws]    SSLNOHOSTCHECK=TRUE

#Or in moab.cfg
RMCFG[mws]           SSLNOHOSTCHECK=TRUE
```

WARNING: This setting could compromise the security of the system and should not be used in production environments.

- **SSLNOPEERCHECK**: Lets you disable the SSL check to make sure that the certificate is valid.

```
#In moab-private.cfg
CLIENTCFG[RM:mws]    SSLNOPEERCHECK=TRUE

#Or in moab.cfg
RMCFG[mws]           SSLNOPEERCHECK=TRUE
```

WARNING: This setting could compromise the security of the system and should not be used in production environments.

Related Topics

- Moab Workload Manager Resource Manager Integration
- Configuring Moab Workload Manager

Resource Manager Queries

During each iteration of Moab Workload Manager's cycle, it will query MWS through the RM interface to access current node, virtual machine, and job information. At this point, all reports are loaded from the database and consolidated into a single report of each object as explained in Data Consolidation.

All unset (or null) values for properties on reports are ignored.

In some cases it may be desired to query MWS directly for the current consolidated node, storage, virtual machine, and job reports. This may be done using the following URLs which return data in a format that is a subset of the API version 3 interface for each object (i.e. `/rest/nodes?api-version=3`, `/rest/vms?api-version=3`, `/rest/jobs?api-version=3`).

Query	Description
`/rest/plugins/all/rm/cluster-query?api-version=3`	Retrieves consolidated node, storage, and virtual machine reports from all plugins.

Query	Description
`/rest/plugins/<ID>/rm/cluster-query?api-version=3`	Retrieves consolidated node, storage, and virtual machine reports for the specified plugin ID.
`/rest/plugins/all/rm/workload-query?api-version=3`	Retrieves consolidated job reports from all plugins.
`/rest/plugins/<ID>/rm/workload-query?api-version=3`	Retrieves consolidated job reports for the specified plugin ID.

These queries have no effect on the data itself. In other words, reports are not removed or manipulated when RM queries are performed. These are manipulated only the RM services as described in Reporting State Data.

Examples

The following example uses cURL (see curl Samples) to perform the query.

```
$ curl -u moab-admin:changeme! http://localhost:8080/mws/rest/plugins/all/rm/cluster-
query?api-version=3&pretty=true
{
    "nodes": {
        "n1.test": {
            "states": {
                "state": "IDLE"
            },
            "lastUpdatedDate": 1382386344,
            "resources": {
                "processors": {
                    "configured": 4
                },
                "memory": {
                    "configured": 8191,
                    "available": 7206
                },
                "gres1": {
                    "configured": 100
                }
            },
            "metrics": {
                "cpuLoad": 0.008233333333333334,
                "vmcount": 0,
                "cpuUtilization": 0.2008333333333333
            },
            "featuresReported": [
                "feature1"
            ],
            "ipAddress": "10.0.8.69",
            "operatingSystem": {
                "hypervisorType": "esx",
                "image": "vcenter-vcenter-esx-4.x",
                "virtualMachineImages": [
                    "centos6-v7"
                ]
            },
            "variables": {
                "VCENTER_DATASTORE_REMOTE1": "datastore-448",
                "VCENTER_DATASTORE_LOCAL1": "datastore-411"
            },
            "attributes": {
                "MOAB_DATACENTER": {
                    "value": "vcenter-datacenter-401",
                    "displayValue": "vcenter-vcenter - adaptive data center"
                }
            }
        },
        "n2.test": {
            "states": {
                "state": "IDLE"
            },
            "lastUpdatedDate": 1382386344,
            "resources": {
                "processors": {
                    "configured": 4
                },
                "memory": {
                    "configured": 10239,
                    "available": 9227
                },
                "gres1": {
```

```
                    "configured": 100
                }
            },
            "metrics": {
                "cpuLoad": 0.00805,
                "vmcount": 0,
                "cpuUtilization": 0.19666666666666666
            },
            "featuresReported": [
                "feature1",
                "feature2"
            ],
            "ipAddress": "10.0.8.76",
            "operatingSystem": {
                "hypervisorType": "esx",
                "image": "vcenter-vcenter-esx-5.0",
                "virtualMachineImages": [
                    "centos6-v7",
                    "centos6",
                    "win2008"
                ]
            },
            "variables": {
                "VCENTER_DATASTORE_REMOTE1": "datastore-448",
                "VCENTER_DATASTORE_LOCAL1": "datastore-415"
            },
            "attributes": {
                "MOAB_DATACENTER": {
                    "value": "vcenter-datacenter-401",
                    "displayValue": "vcenter-vcenter - adaptive data center"
                }
            }
        },
        "n3.test": {
            "states": {
                "state": "IDLE"
            },
            "lastUpdatedDate": 1382386344,
            "resources": {
                "processors": {
                    "configured": 4
                },
                "memory": {
                    "configured": 10239,
                    "available": 9229
                },
                "gres1": {
                    "configured": 100
                }
            },
            "metrics": {
                "cpuLoad": 0.0097,
                "vmcount": 0,
                "cpuUtilization": 0.2375
            },
            "featuresReported": [
                "feature1"
            ],
            "ipAddress": "10.0.8.72",
            "operatingSystem": {
                "hypervisorType": "esx",
                "image": "vcenter-vcenter-esx-5.0",
```

```
                "virtualMachineImages": [
                    "centos6-v7",
                    "centos6",

                    "win2008"
                ]
            },
            "variables": {
                "VCENTER_DATASTORE_REMOTE1": "datastore-448",
                "VCENTER_DATASTORE_LOCAL1": "datastore-416"
            },
            "attributes": {
                "MOAB_DATACENTER": {
                    "value": "vcenter-datacenter-401",
                    "displayValue": "vcenter-vcenter - adaptive data center"
                }
            }
        },
        "n4.test": {
            "states": {
                "state": "IDLE"
            },
            "lastUpdatedDate": 1382386344,
            "resources": {
                "processors": {
                    "configured": 4
                },
                "memory": {
                    "configured": 10239,
                    "available": 9229
                },
                "gres1": {
                    "configured": 100
                }
            },
            "metrics": {
                "cpuLoad": 0.007883333333333334,
                "vmcount": 0,
                "cpuUtilization": 0.1925
            },
            "featuresReported": [
                "feature2"
            ],
            "ipAddress": "10.0.8.77",
            "operatingSystem": {
                "hypervisorType": "esx",
                "image": "vcenter-vcenter-esx-5.0",
                "virtualMachineImages": [
                    "centos6-v7",
                    "centos6",
                    "win2008"
                ]
            },
            "variables": {
                "VCENTER_DATASTORE_REMOTE1": "datastore-448",
                "VCENTER_DATASTORE_LOCAL1": "datastore-958"
            },
            "attributes": {
                "MOAB_DATACENTER": {
                    "value": "vcenter-datacenter-401",
                    "displayValue": "vcenter-vcenter - adaptive data center"
                }
```

```
            }
        }
    },
    "vms": {
        "vm1": {
            "states": {
                "state": "DOWN",
                "powerState": "OFF"
            },
            "host": {
                "name": "n1.test"
            },
            "lastUpdatedDate": 1382386344,
            "resources": {
                "processors": {
                    "configured": 4
                },
                "memory": {
                    "configured": 12288
                }
            },
            "metrics": {
                "vmcount": 1
            }
        }
    },
    "storage": {}
}
```


Related Topics

- Moab Workload Manager Resource Manager Integration
- Resource Manager Queries

Plugin Type Management

Plugin types may be managed and accessed with Moab Web Services dynamically, even while running. Operations are provided to upload (add or update) plugin types and to list or show current plugin types. The available fields that are displayed with plugin types are given in Fields: Plugin Types. For more information on how these fields are set, see Plugin Projects and Metadata.

Plugin Type JAR or groovy files should never be manually copied into the `MWS_HOME/plugins` directory. They must be managed using the methods shown in this section or through the REST API (see Plugin Types).

Bundled plugin types are included automatically in Moab Web Services releases and may be utilized immediately after startup. See Plugin Management for more information on how to utilize these plugin types.

> The plugin type documentation is now located in the plugin type management pages. See Plugin Type Documentation for more information.

This section contains these topics:

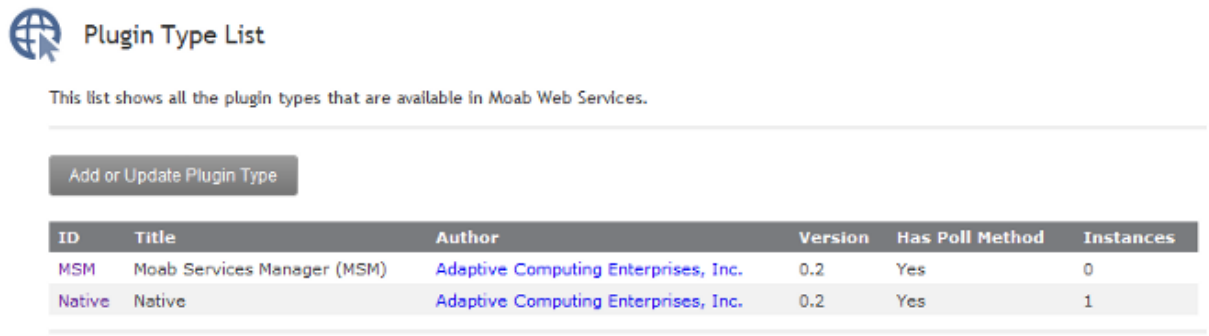
- Listing Plugin Types
- Displaying Plugin Types
- Plugin Type Documentation
- Add or Update Plugin Types

Related Topics

- About Moab Web Services Plugins

Listing Plugin Types

To list all plugin types, browse to the MWS home page (for example, `https://servername/mws`). Log in as the admin user, then click **Plugins > Plugin Types**.

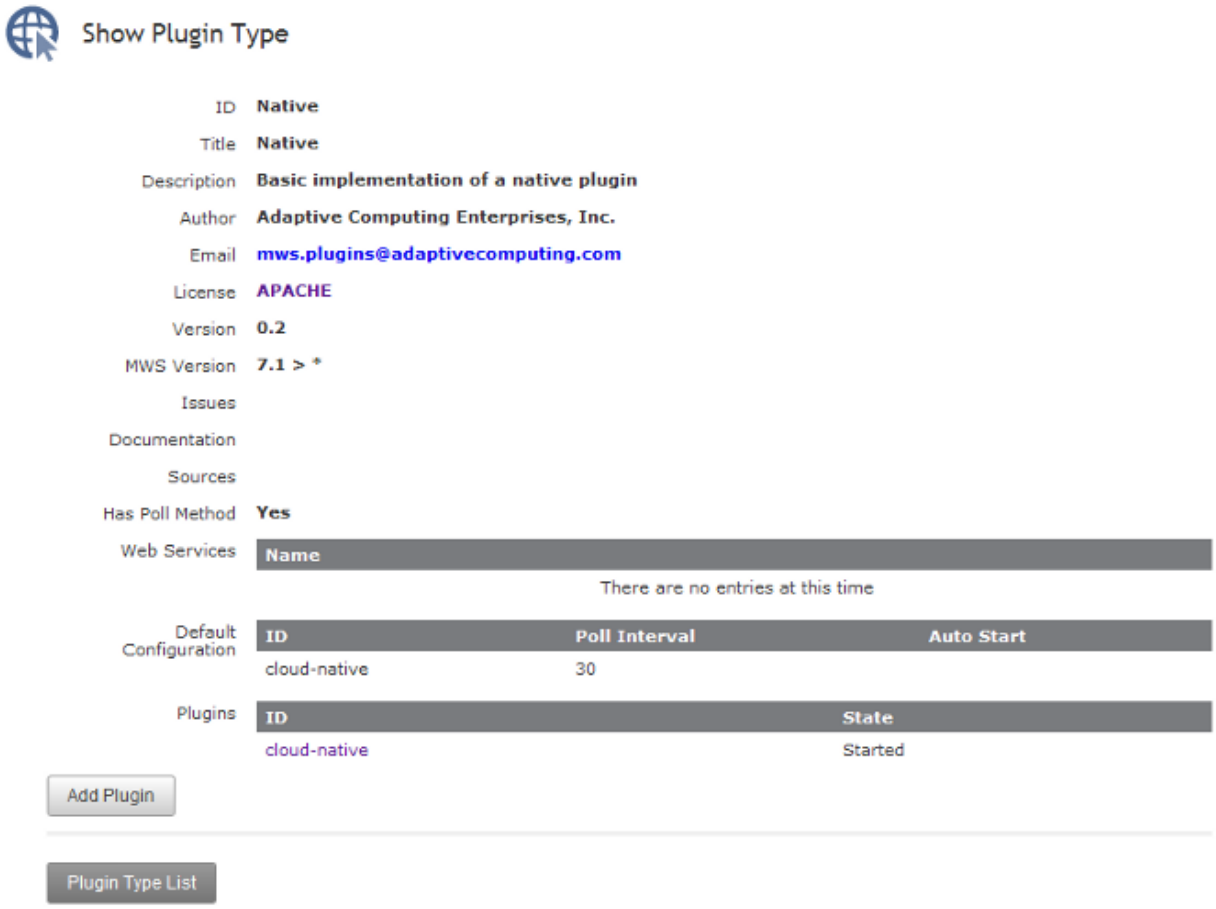


Related Topics

- Plugin Type Management

Displaying Plugin Types

To show information about a plugin type, go to the **Plugin Type List** page and click the desired plugin type.



Related Topics

- Plugin Type Management

Plugin Type Documentation

To show the documentation for a plugin type, go to the **Plugin Type List** page and click the desired plugin type. Then, click the **Open Documentation** button. This will display any documentation bundled with the plugin type.

Related Topics

- Plugin Type Management

Add or Update Plugin Types

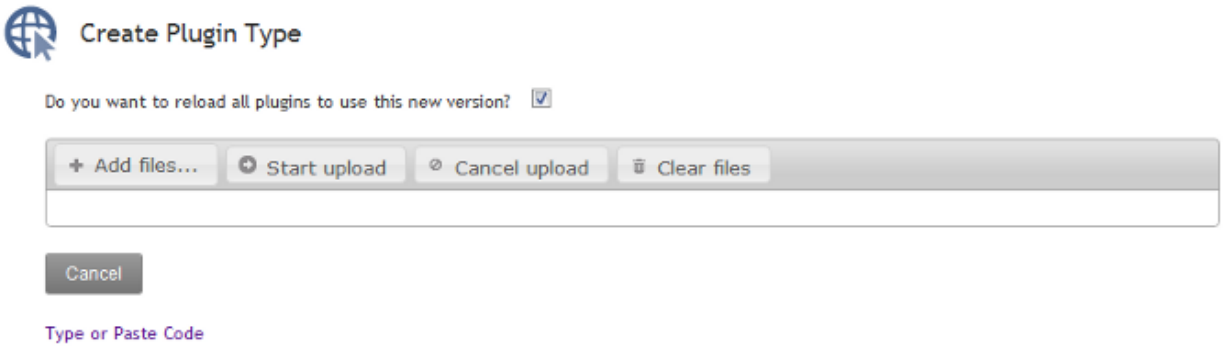
Plugin types can be uploaded into Moab Web Services using a Groovy file, a Java Archive (JAR) file, or pasted Groovy code. To access the plugin type

upload page, navigate to the **Plugin Type List** page and click **Add or Update Plugin Type**. The default interface of this page enables the uploading of a single Groovy class file or a JAR file.

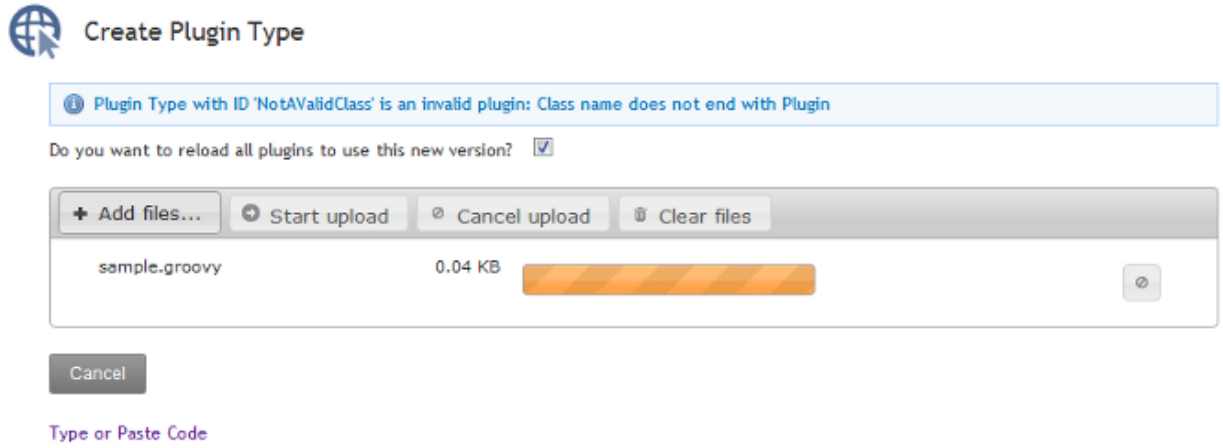
When a plugin type is updated, by default all corresponding plugins created from the plugin type will be recreated. If this behavior is not desired, clear the **Do you want to reload all plugins to use this new version?** checkbox before uploading the plugin type.

Single class file

Groovy files containing a single plugin type may be uploaded at the `/mws/admin/plugin-types/create` URL.



If the upload failed or an error occurred during initialization of the plugin, an error message will be displayed.

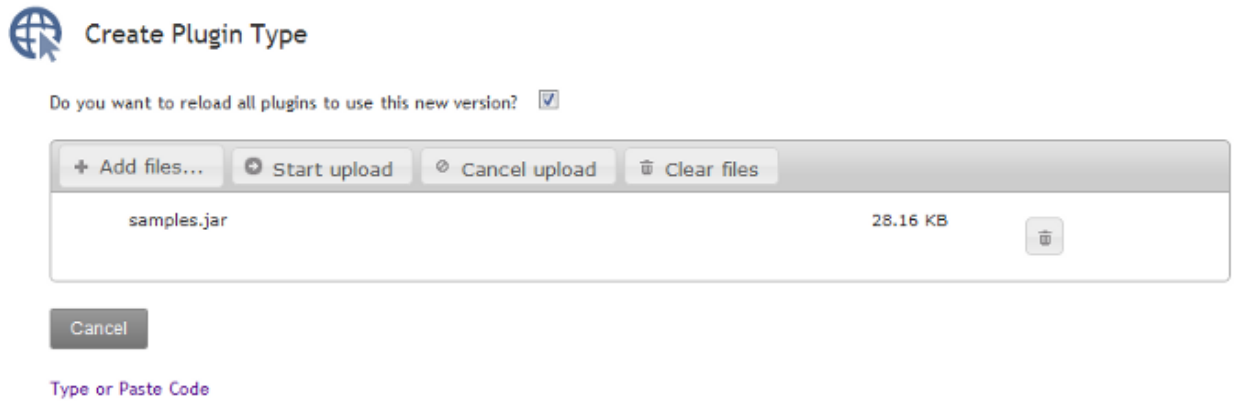


JAR file

A JAR file, as described in Packaging Plugins, containing one or more plugins may also be uploaded using the same process as the Groovy file.

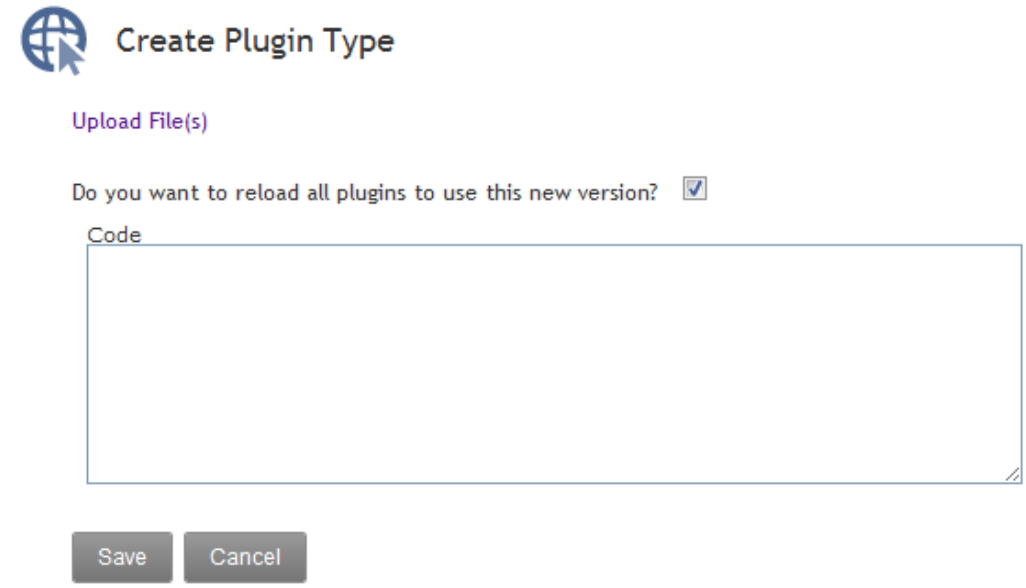
Click **Add files...**, select the `.jar` file, and click the **Start upload** button. If the upload failed or an error occurred during initialization of the plugin(s), an error message will be displayed.

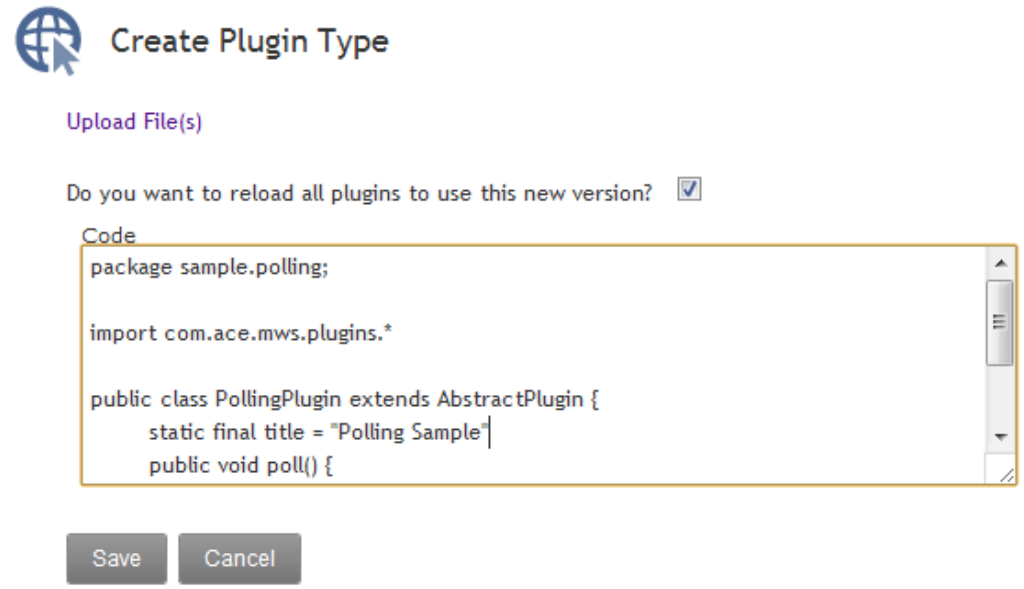
The JAR upload process differs from the single file in that if successful, the name of the JAR file itself is displayed instead of the plugin name(s).



Code

To paste or type code directly into MWS and have it be loaded as a single class file, click **Type or Paste Code**, and type or paste the code into the presented text box.





When the code is in the box, click **Create**. If the upload succeeded and the code was able to be compiled as Groovy, the browser will be redirected to the **Show Plugin Type** page. If the upload failed or an error occurred during compilation or initialization of the plugin, an error message will be displayed.

You may need to refer to the MWS log file for additional details and error messages in the case of a failure.

Related Topics

- Plugin Type Management

Plugin Management

Plugins may be managed and accessed with Moab Web Services dynamically, even while running. This includes plugin instance and lifecycle management. Additionally, default configuration values may be set for new plugins. In order to access custom web services, the REST API must be utilized as described in Accessing Plugin Web Services. The available fields that are displayed with plugins are given in the **Fields: Plugins** reference.

This section contains these topics:

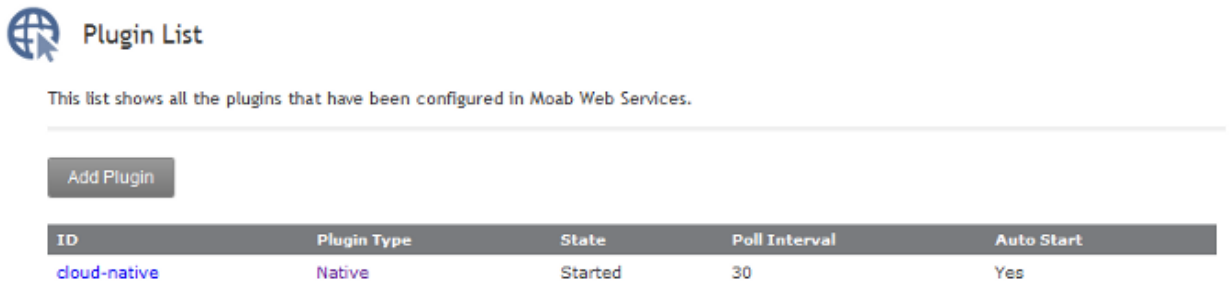
- Listing Plugins
- Creating a Plugin
- Displaying a Plugin
- Modifying a Plugin
- Deleting a Plugin
- Monitoring and Lifecycle Controls
- Setting Default Plugin Configuration

Related Topics

- About Moab Web Services Plugins

Listing Plugins

To list all plugins, browse to the MWS home page (for example, `https://servername/mws`). Log in as the admin user, then click **Plugins > Plugins**.

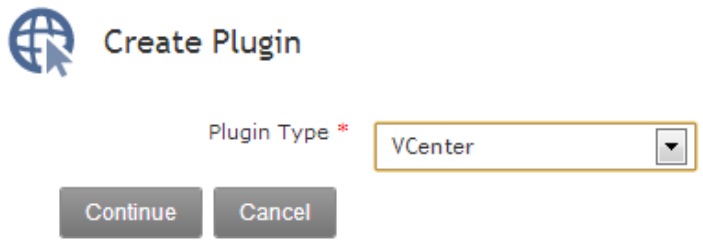


Related Topics

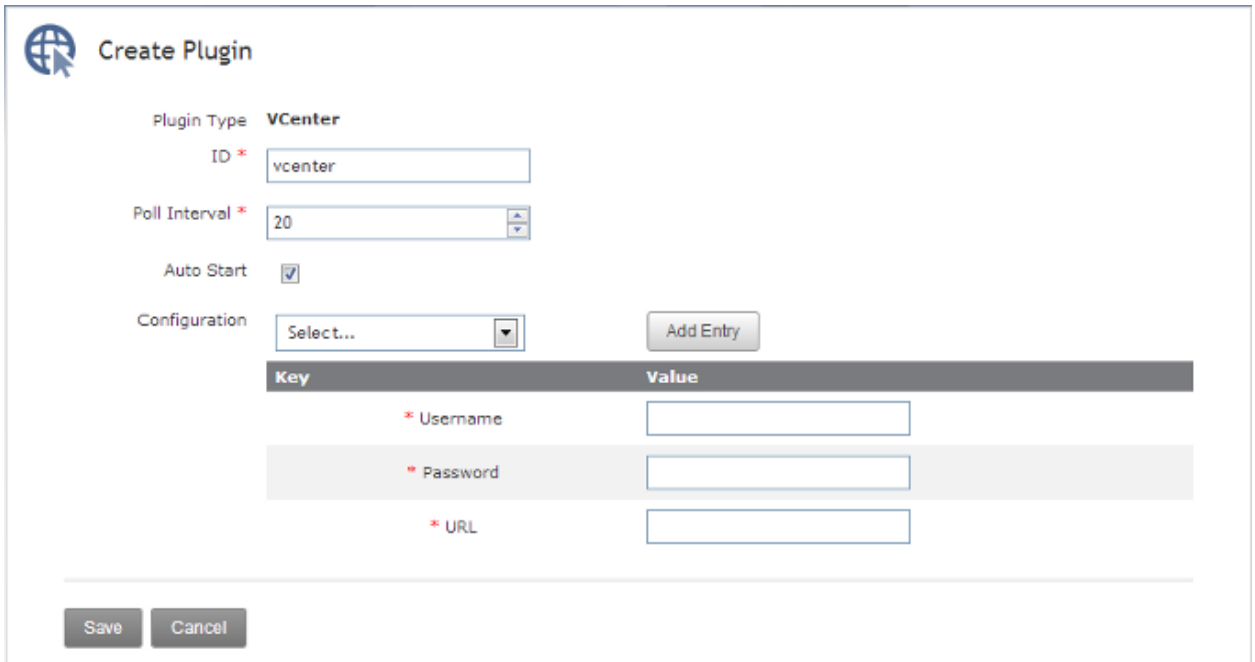
- Plugin Management

Creating a Plugin

To create a plugin, go to the **Plugin List** page and click **Add Plugin**. First, a **Plugin Type** must be selected to continue to actually create the plugin.



The page is automatically built to support the plugin type's constraints (see Configuration Constraints). The **ID** field will be automatically filled in with a suggested value, and the **Poll Interval** field will be displayed only if the plugin type has a `poll` method. The required configuration fields are displayed by default, and optional fields may be selected and added to the configuration from the drop down at the top of the configuration section. See the **Fields: Plugins** reference section for more information on the fields.

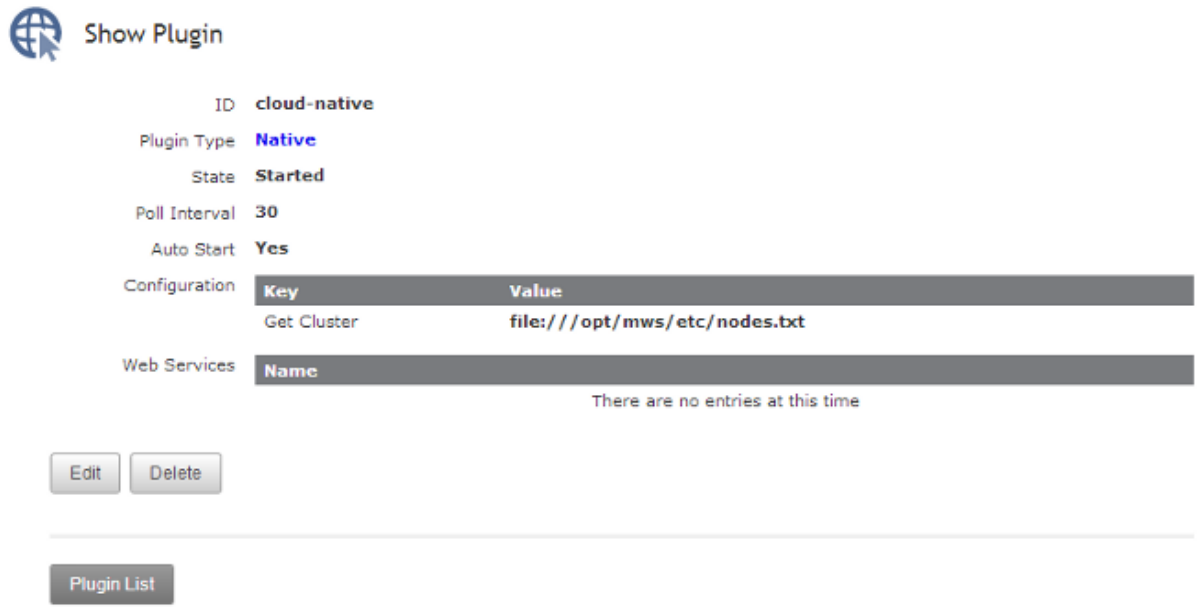


Related Topics

- Plugin Management

Displaying a Plugin

To show information about a plugin, go to the **Plugin List** page and click the desired plugin ID.

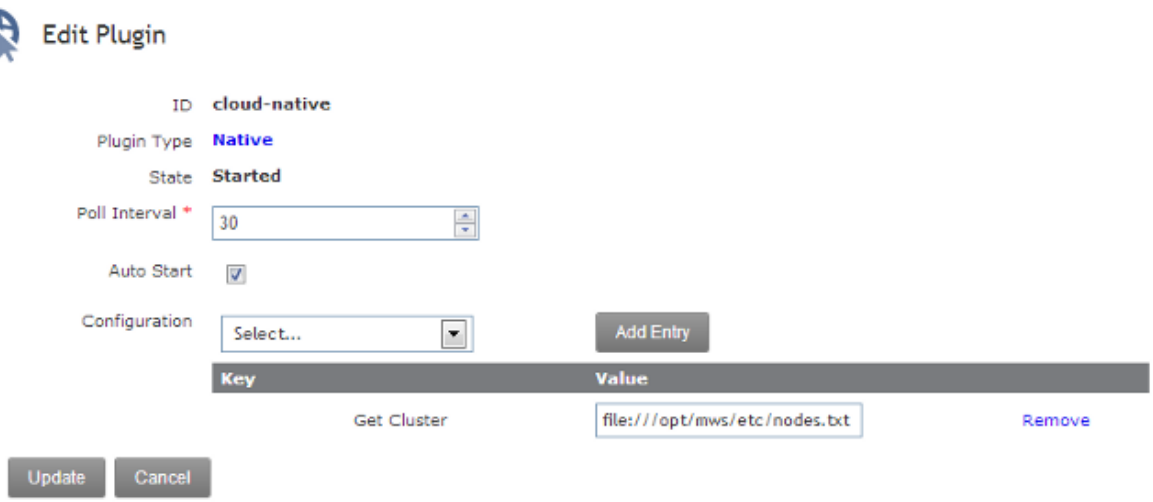


Related Topics

- Plugin Management

Modifying a Plugin

To modify a plugin, go to the **Plugin List** page, click the desired plugin ID, and then click **Edit**. See the **Fields: Plugins** reference section for more information on available fields.



Related Topics

- Plugin Management

Deleting a Plugin

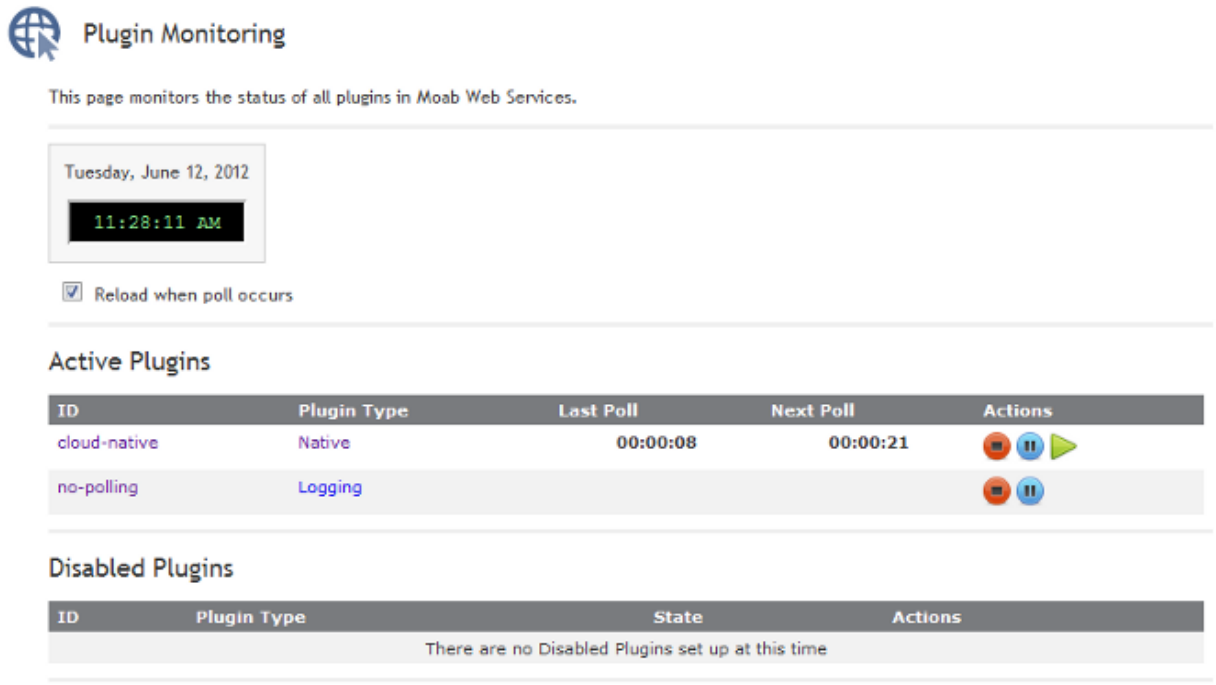
To delete a plugin, go to the **Plugin List** page, click the desired plugin ID, and then click **Delete**. A confirmation message is shown. If the **OK** button is clicked, the plugin is deleted from the system and cannot be recovered, including all configuration.

Related Topics

- Plugin Management

Monitoring and Lifecycle Controls

To monitor and control the lifecycle of plugins, browse to the MWS home page (for example, `https://servername/mws`). Log in as the admin user, then click **Plugins > Plugin Monitoring**. This page displays the current state of all plugins as well as their polling status.

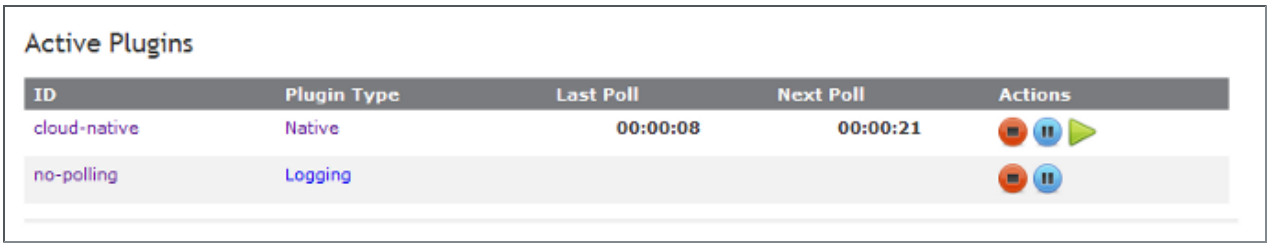


If plugins are created from plugin types which do not have a `poll` method, their lifecycle controls will be limited. Any information below which mentions polling does not apply to the 'no-polling' plugin shown in the screenshots.

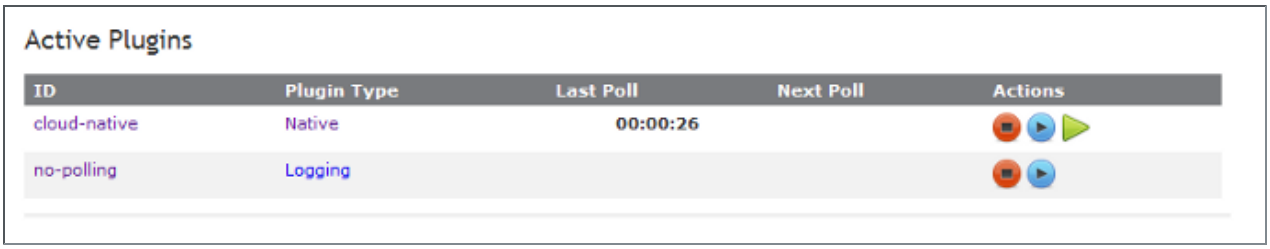
Active plugins

Active plugins are those which are in the Started or Paused states. These are available to receive events such as polling. If paused, a plugin will not receive events but is not actually stopped, therefore no stop events are triggered.

The following images demonstrate the status of plugins in the active states.



Started plugins which can include the relative time of the last poll as well as the time of the next poll in a countdown format. Action buttons are available to stop or pause the plugin as well as trigger an immediate poll event.

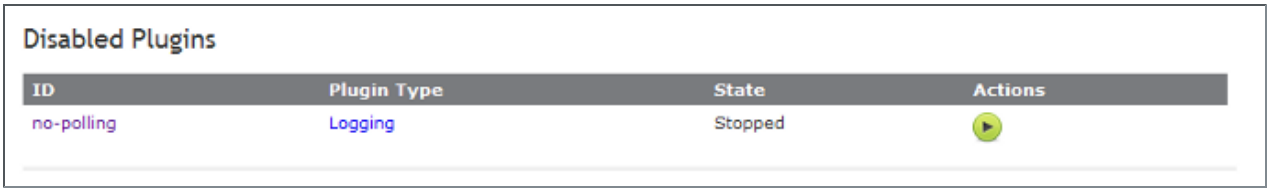


Paused plugins which can include only the last polling time. Action buttons are available to stop or resume the plugin, as well as trigger an immediate poll event.

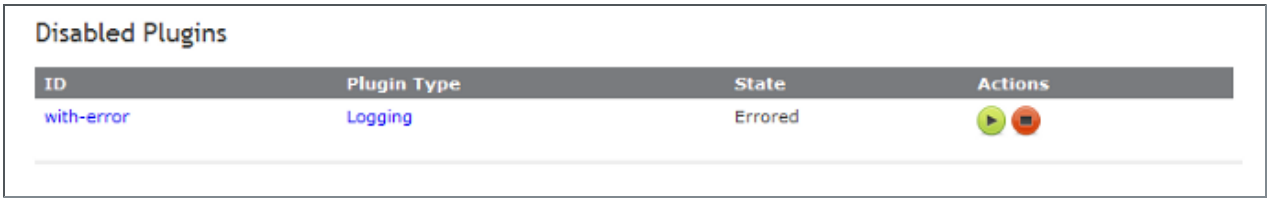
Disabled plugins

Disabled plugins are those which are in the Stopped or Errored states. These plugins do not receive events such as polling. If errored, a plugin may either be stopped, which represents a "clearing" of the error, or started normally. However, if no action is taken on an errored plugin, it likely will not start due to the fact that most plugins are put into the errored state during startup of the plugin.

The following images demonstrate the representation of plugins in the disabled states.



Stopped plugins. A single action button is available to attempt to start the plugin.



An errored plugin. As mentioned previously, action buttons are available to stop the plugin or clear the error as well as attempt to start the plugin. If the start fails, an error message will be displayed.

Related Topics

- Plugin Management

Setting Default Plugin Configuration

Configuration of default values for plugin configuration parameters involves setting fields in the MWS configuration file. These values are used if no values are provided when creating a new plugin. Additionally, the default values will be displayed to the user on the **Create Plugin** page.

The parameters to configure are documented on Configuration and comprise most values starting with plugins.

Related Topics

- Plugin Management

Plugin Services

To use the built-in services, declare a variable with the correct name as a property in the plugin class.

The convention for each service name is to remove the leading "I" and lower case the resulting first letter. For example, the property to use the IMoabRestService would be called moabRestService. The following is an example of using the IPluginControlService in this manner.

```
Using the IPluginControlService
-----------------------------------

package example;
import com.adaptc.mws.plugins.*;

class ExamplePlugin {
       IPluginControlService pluginControlService;

       public poll() {
              // Use service…
              pluginControlService.stop("pluginId");
       }
}
```

Use of the Groovy anonymous type "def" may also be used. For example, the service definition above would use `def pluginControlService` instead of `IPluginControlService pluginControlService`.

Do *not* attempt to create a new instance of the services before use, such as in a constructor. The services will be automatically injected before any methods are called on the plugin.

API documentation

The `com.adaptc.mws.plugins` package contains interfaces for all bundled services available to plugin types. These may be used as discussed above. All services begin with "I" and end with "Service", as in IMoabRestService (Moab REST Service).

Related Topics

- About Moab Web Services Plugins

Job RM Service

The job RM service may be used to report job state data to Moab Workload Manager through the RM interface. See Reporting State Data for more information. It may also be used to retrieve previous reports made by a plugin. Please note that due to data consolidation (see Data Consolidation), old job reports may no longer exist in the database by the time the query is done.

The `jobRMService` property will be injected with a class of type IJobRMService in all plugin types. Note that it is not available for injection in translators or custom components.

Related Topics

- Plugin Services

Moab REST Service

The Moab REST service may be used to access the MWS RESTful API (see Resources Introduction) in plugins. All "requests" made through this service are internal only and no data is actually transmitted over the network. See Accessing MWS REST Resources for more information.

The `moabRestService` property will be injected with a class of type IMoabRestService in all plugin types.

Accessing resources

In order to access a resource, a relative URL matching that in the documentation must be used along with a HTTP method, such as GET, POST, PUT, or DELETE. The method names on IMoabRestService match the HTTP methods directly. For example, to call a GET operation on `/rest/jobs`, use the following code:

```
moabRestService.get("/rest/jobs")
```

Using parameters correctly

Although the ordering of the parameters for each method on IMoabRestService may seem confusing at first glance, this is to allow for easy use with Groovy. Examples are given below for each combination of parameters:

```
String URL
-----------------------------------

moabRestService.get("/rest/jobs")
```

```
Map options, String URL
-----------------------------------

moabRestService.get("/rest/jobs", hooks:true, contentType:"application/json")
```

```
String URL, Closure data
-----------------------------------

moabRestService.get("/rest/jobs/job.1") {
       [flags:"RESTARTABLE"]
}
```

```
Map options, String URL, Closure data
-----------------------------------

moabRestService.get("/rest/jobs/job.1", hooks:true, contentType:"application/json") {
       [flags:"RESTARTABLE"]
}
```

Options

The following options are valid in each method call supporting the `options` parameter:

Name	Type	Default	Description
data	See Valid data types	--	Specifies the body of the "request." This can be overwritten by the `data` Closure parameter.
hooks	Boolean	false	Specifies whether or not hooks (see Pre- and Post-Processing Hooks) are run as part of the "request."
contentType	String	application/json	Indicates the content type used for the request.
params	Map	--	Indicates URL query parameters to use for the "request," such as `query`, `sort`, `proxy-user`, or others.

Valid data types

If the `data` Closure parameter is specified, it overwrites the data option. In each case, there are four valid types for the data option or return value of the data closure:

- A non-null JSON instance.
- A valid JSON string. This will be converted into a JSON instance.
- A valid Map instance. This will be converted into a JSONObject instance.
- A valid List instance. This will be converted into a JSONArray instance.

A JSONException may be thrown if the JSON string is invalid or the Map or List contains values that cannot be serialized to JSON.

Related Topics

- Plugin Services

Node RM Service

The node RM service may be used to report node state data to Moab Workload Manager through the RM interface. See Reporting State Data for more information. It may also be used to retrieve previous reports made by a plugin. Please note that due to data consolidation (see Data Consolidation), old node reports may no longer exist in the database by the time the query is done.

The `nodeRMService` property will be injected with a class of type INodeRMService in all plugin types. Note that it is not available for injection in translators or custom components.

Related Topics

- Plugin Services

Plugin Control Service

This service is currently in Beta. Interfaces may change significantly in future releases.

The control service allows lifecycle management operations to be performed on plugins. It also provides methods to create and retrieve plugins. Note that the plugin control service may be used by other plugins, allowing one plugin to dynamically create, retrieve, start, or stop plugins.

The `pluginControlService` property will be injected with a class of type IPluginControlService in all plugin types.

Examples

```
Create plugin with default configuration
-------------------------------------

try {
       if (pluginControlService.createPlugin("myPlugin", "Native"))
              log.info "myPlugin was created successfully!"
       else
              log.warn "There was an error creating myPlugin"
} catch(PluginStartException e) {
       log.warn "There was a problem starting the new plugin: ${e.message}"
} catch(InvalidPluginConfigurationException e) {
       log.warn "There were errors with the plugin's configuration: ${e.errors}"
}
```

```
Create plugin with custom configuration
-------------------------------------

if (pluginControlService.createPlugin("myPlugin", "Native", [autoStart:false,
pollInterval:600]))
       log.info "myPlugin was created successfully!"
else
       log.warn "There was an error creating myPlugin"
```

```
Start plugin
-------------------------------------

try {
       pluginControlService.start("myPlugin")
} catch(PluginStartException e) {
       log.warn "There was a problem starting the plugin: ${e.message}"
} catch(InvalidPluginException) {
       log.warn "The plugin 'myPlugin' is invalid"
} catch(InvalidPluginConfigurationException e) {
       log.warn "The plugin has an invalid configuration: ${e.errors}"
}
```

```
Stop plugin
-------------------------------------

try {
       pluginControlService.stop("myPlugin")
} catch(PluginStopException e) {
       log.warn "There was a problem stopping the plugin: ${e.message}"
} catch(InvalidPluginException) {
       log.warn "The plugin 'myPlugin' is invalid"
}
```

```
Configure plugin
-------------------------------------

try {
       pluginControlService.configure("myPlugin")
} catch(InvalidPluginException) {
       log.warn "The plugin 'myPlugin' is invalid"
} catch(InvalidPluginConfigurationException e) {
       log.warn "The plugin has an invalid configuration: ${e.errors}"
}
```

Related Topics

- Plugin Services

Plugin Datastore Service

The datastore service is provided to allow a plugin to persist data to the database that is isolated from all other persistent data. In other words, this service provides access to a plugin's individual datastore (see Individual Datastore).

The `pluginDatastoreService` property will be injected with a class of type IPluginDatastoreService in all plugin types. Note that it is not available for injection in translators or custom components.

Examples

```
Adding a single custom entry
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               def data = [:]
               … // Add data here to the Map
               if (pluginDatastoreService.addData(collectionName, data))
                       log.info("Data successfully added")
               else
                       log.warn("There was an error adding the data")
       }
}
```

```
Adding multiple entries
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               def dataList = []
               dataList.add( /* Custom Map of data here */)
               dataList << … // Custom Map of data here
               if (pluginDatastoreService.addData(collectionName, dataList))
                       log.info("Data entries successfully added")
               else
                       log.warn("There was an error adding the data entries")
       }
}
```

```
Updating a single entry
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               def data = [:]
               … // Add data here to the Map
               if (pluginDatastoreService.updateData(collectionName, "key", "value", data))
                       log.info("Data successfully updated")
               else
                       log.warn("There was an error updating the data")
       }
}
```

```
Querying if a collection exists
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               if (pluginDatastoreService.exists(collectionName))
                       log.info("Collection exists")
               else
                       log.warn("The collection does not exist")
       }
}
```

```
Querying contents of a collection
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               def dataList = pluginDatastoreService.getCollection(collectionName)
               if (dataList!=null)
                       log.info("Collection successfully queried")
               else
                       log.warn("The collection does not exist!")
       }
}
```

```
Retrieving a single entry
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               def data = pluginDatastoreService.getData(collectionName, "key", "value")
               if (data!=null)
                       log.info("Data successfully retrieved")
               else
                       log.warn("The entry with key==value does not exist")
       }
}
```

```
Removing a collection
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               def data = pluginDatastoreService.clearCollection(collectionName)
               // Data now contains the collection that was cleared
               if (data!=null)
                       log.info("Collection successfully cleared")
               else
                       log.warn("The collection does not exist!")
       }
}
```

```
Removing a single entry
-----------------------------------

package example

public class ExamplePlugin {
       def pluginDatastoreService

       public void poll() {
               def collectionName = "collection1"
               if (pluginDatastoreService.removeData(collectionName, "key", "value"))
                       log.info("Data entry successfully removed")
               else
                       log.warn("The entry where key==value does not exist!")
       }
}
```

Related Topics

- Plugin Services

Plugin Event Service

The event service is provided to ease the burden and reduce boilerplate code for creating new events and updating notification conditions. For more information on how to use this service, see Creating Events and Notifications.

The `pluginEventService` property will be injected with a class of type IPluginEventService in all plugin types. Note that it is not available for injection in translators or custom components.

Examples

```
Creating a custom event
-----------------------------------

package example

import com.adaptc.mws.plugins.IPluginEventService.Severity
import com.adaptc.mws.plugins.IPluginEventService.EscalationLevel
import com.adaptc.mws.plugins.IPluginEventService.AssociatedObject

public class ExamplePlugin {
       def pluginEventService

       public void poll() {
               // Create a completely custom event
               pluginEventService.createEvent(Severity.INFO, EscalationLevel.USER, 0x4F, "Custom
Type",
                                              "poll", "My event occurred", null, null)
       }
}
```

```
Creating a custom event with messages
-----------------------------------

package example

import com.adaptc.mws.plugins.IPluginEventService.Severity
import com.adaptc.mws.plugins.IPluginEventService.EscalationLevel
import com.adaptc.mws.plugins.IPluginEventService.AssociatedObject

public class ExamplePlugin {
       def pluginEventService

       public void poll() {
       // Use i18n messages for another event
       def args = ["arg1", "arg2"]
               pluginEventService.createEvent(Severity.WARN, EscalationLevel.POWER_USER, 0x5F,
"Custom Type",
                                              "poll", message
(code:"examplePlugin.customEvent.message", args:args), args,
                                              // AssociatedObjects or simple maps may be
used
                                              [new AssociatedObject(type:"type1", id:"id1"),
[type:"type2", id:"id2"])
       }
}
```

```
Creating an event from EventEnumeration
-----------------------------------

package example

import com.adaptc.mws.plugins.EventEnumeration
import com.adaptc.mws.plugins.IPluginEventService.Severity
import com.adaptc.mws.plugins.IPluginEventService.EscalationLevel
import com.adaptc.mws.plugins.IPluginEventService.AssociatedObject

public class ExamplePlugin {
       def pluginEventService

       public void poll() {
              // Messages are pulled for messages.properties file(s) and the arguments are used
       def args = ["arg1", "arg2"]
              pluginEventService.createEvent(MyEvents.EVENT_INFO, args, [[type:"type1", id:"id1"])
              pluginEventService.createEvent(MyEvents.EVENT_WARN, args, [[type:"type2", id:"id2"])
       }
}

@EventEnumeration
enum MyEvents {
       EVENT_INFO("Information", INFO, USER),
       EVENT_ERROR("Warning", WARN, USER)

       static final String EVENT_TYPE_PREFIX = "Example Plugin"
       static final String ORIGIN_SUFFIX = "poll"
}
```

```
Create or update a notification
-----------------------------------

package example

import com.adaptc.mws.plugins.IPluginEventService.EscalationLevel
import com.adaptc.mws.plugins.IPluginEventService.AssociatedObject

public class ExamplePlugin {
       def pluginEventService

       public void poll() {
              pluginEventService.updateNotification(EscalationLevel.POWER_USER, "There is an error
with node1",
                                          // If non-null, this must always be an associated object, never
                                          new AssociatedObject(id:"node1", type:"node"), null)
       }
}
```

Related Topics

- Events
- Notifications
- Notification Conditions
- Plugin Services
- Plugin Developer's Guide
- Fields: Events

- Resources Introduction
- Creating Events and Notifications

SSL Service

The SSL service may be used to manage and load certificates or keys from disk and create socket connections. See Managing SSL Connections for more information.

The `sslService` property will be injected with a class of type ISslService in all plugin types.

Related Topics

- Plugin Services

Storage RM Service

The storage RM service may be used to report storage state data to Moab Workload Manager through the RM interface. See Reporting State Data for more information. It may also be used to retrieve previous reports made by a plugin. Please note that due to data consolidation (see Data Consolidation), old storage reports may no longer exist in the database by the time the query is done.

The `storageRMService` property will be injected with a class of type IStorageRMService in all plugin types. Note that it is not available for injection in translators or custom components.

Related Topics

- Reporting State Data
- Plugin Services

Virtual Machine RM Service

The virtual machine RM service may be used to report virtual machine state data to Moab Workload Manager through the RM interface. See Reporting State Data for more information. It may also be used to retrieve previous reports made by a plugin. Please note that due to data consolidation Data Consolidation, old virtual machine reports may no longer exist in the database by the time the query is done.

The `virtualMachineRMService` property will be injected with a class of type IVirtualMachineRMService in all plugin types. Note that it is not available for injection in translators or custom components.

Related Topics

- Plugin Services

Chapter 7 Plugin Types

In this chapter:

- Power Management Plugin on page 405
- OpenStack Plugin on page 411
- ViewpointQueryHelper Plugin on page 415

Power Management Plugin

The Power Management plugin is used as a resource manager to Moab to report and manipulate the power state (On or Off) for each node. Moab considers nodes in the power state On or Off; however, through Torque and scripts, we are able to separate the Off state into those controlled through the operating system (Standby, Suspend, Hibernate, Shutdown) and those controlled through hardware (Off). This plugin provides an easy way to integrate with Moab to translate Moab's Off action into the desired Torque or script action for each node. A cluster will have multiple instances of this plugin when it has varied hardware integration and/or credentials.

Creating a Power Management Plugin

To create a Power Management plugin, see Creating a Plugin. During plugin creation, refer to the Configuration section.

Configuration

Configuration Parameters

Name	Key	Required	Type	Description
Node Configuration File	nodeConfigurationFile	Yes	String	File containing list of nodes that use the scripts and credentials in this plugin instance. This is also the file to configure a particular node's off state, or an off state that will override the default off state for this instance.
Username File	usernameFile	Yes	String	File containing username issued to the scripts with the -u option.

Name	Key	Required	Type	Description
Password File	passwordFile	Yes	String	File containing password issued to the scripts with the -p option.
Node Power Script	nodePowerScript	Yes	String	Script that powers on and off nodes and wakes them from a low power state.
Node Query Script	nodeQueryScript	Yes	String	Script that queries power state using an intelligent platform management interface.
Default Power Off State	defaultPowerOffState	Yes	String	Actual state (Standby, Suspend, Hibernate, Shutdown, or Off) nodes will go into when Moab powers them off.
Max Threads	maxThreads	Yes	Integer	Thread count issued to the scripts with the -t option (defaults to 4).

Plugin Management

For information on managing the IPMI plugin, including stopping it, starting it, and checking on its status, see the Plugin Management section of the MWS Guide.

Web Services

Node Power (Secured)

Resource URLs

Resource
`/rest/plugins/<pluginId>/services/nodePower`
`/rest/plugins/<pluginId>/services/node-power`

URL Parameters

Parameter	Name	Type	Description
nodes	Moab Nodes	String	A comma-delimited list of Moab node names. It is required.

Parameter	Name	Type	Description
power	The Power State	String	The power command Moab issues the node (On or Off).

Response Fields

Field	Name	Type	Description
success	Success Indicator	Boolean	True if the power script and/or pbsnodes was successful, otherwise false.
messages	Messages	List of Strings	Only present when the request was not successful or the node was not configured with the plugin instance. Contains error messages describing why the pbsnodes or the power script failed.

Additional Information

This web service was intended for Moab's use only and is exposed for debugging and testing your customized scripts.

Reload Node Configuration (Secured)

Resource URLs

Resource
`/rest/plugins/<pluginId>/services/reloadNodeConfiguration`
`/rest/plugins/<pluginId>/services/reload-node-configuration`

URL Parameters

Parameter	Name	Type	Description
No URL parameters required			

Response Fields

Field	Name	Type	Description
success	Success Indicator	Boolean	True if the reload succeeded, otherwise false.
messages	Messages	List of Strings	Only present when the request failed. Contains error messages describing why the reload failed.

Additional Information

The reloadNodeConfiguration web service must be run after any change to the node configuration file for it to take effect.

Node Configuration File

The node configuration file is used when the plugin is first instantiated or the `reloadNodeConfiguration` web service is called. The plugin expects a file that is readable by the tomcat user and has a Moab node name on each line. If the user would like to override the default power-off state of the node, then the node name is followed by a space and the state. For example, a node configuration file might look like this:

```
node01.ac
node02.ac
node03.ac Hibernate
node04.ac Suspend
```

The valid power-off states include Standby, Suspend, Hibernate, Shutdown, and Off. If no power-off state is provided for the node in the configuration file, then the default power-off state will be used.

The Node Power and Query Script

The plugin uses the power script to power on nodes from all power states and to power off nodes only into the Off power state. The plugin uses the power state of the node to decide whether to power on the node with `wake` or `on`. If the node is in Standby or Suspend, the plugin will call the script with the `wake` parameter. If the node is in Hibernate, Shutdown, or Off, the plugin will call the script with the `on` parameter. The plugin calls the power node script with the `off` parameter to put the node in the Off state (it uses Torque to put the node in the Standby, Suspend, Hibernate, and Shutdown state).

The plugin uses the query script to know if a node is in the Off power state. If the query script reports the node as Off, the plugin will report the node as Off to Moab. If the query script reports the node as On, the plugin will look to Torque to make sure the node is in a Running power state before it reports it as On.

The plugin passes the `usernameFile`, `passwordFile`, and `maxThreads` configuration parameters down to the scripts. The node power script is called with this syntax:

```
<nodePowerScript> -u <usernameFile> -p <passwordFile> -t <maxThreads> node01 node02
node03 ... <on|off|wake>
```

The node query script is called with this syntax:

```
<nodeQueryScript> -u <usernameFile> -p <passwordFile> -t <maxThreads> node01 node02
node03 ...
```

The plugin expects the scripts to print JSON to standard out. An example query script output would look like this:

```
[
    {
        "name": "node01.ac",
        "power": "ON",
        "Processor_2_Temp": 61,
        "Processor_1_Temp": 54
    },
    ...
]
```

Notice it is a list of nodes where each node has the required fields `name` and `power`. All the other key-value pairs will be reported to Moab as a generic resource as long as the value is a number.

The output for the node power plugin is not required; however, the output is read to give the user a detailed error message if needed. For both the node power and query scripts, if the field `error` exists, the plugin will log an error with all the strings in the list. An example error returned to the plugin would look like this:

```
[
    {
        "command": "ipmitool -I lan -H node01i -U admin -f /opt/moab/etc/power-
management/abc-plugin-password-file sdr type temperature",
        "name": "node01.ac",
        "error": [
            "big error"
        ]
    }
    ...
]
```

Troubleshooting

The Power Management plugin logs all errors and warnings to the MWS log file, which is `/opt/mws/log/mws.log` by default. The stacktrace.log file, in the same directory as `mws.log`, can also be helpful in diagnosing problems. If your MWS supports notifications, they are also helpful in diagnosing the error states the plugin is in, if any. Just check for notifications from the `PowerManagement` plugin type and the instance that you are interested in. When the issue has been resolved, you can dismiss the notification. For more information, see the Notification and Notification Condition Resource in the MWS documentation.

Set the appropriate MWS RM precedence

The Create/Edit Plugin pages give the option to set the precedence of the Moab RM plugin. The purpose of the Power Management Plugin is to report node power; however, if the precedence is too low another Moab RM plugin with a higher precedence and conflicting node might overwrite the node power. To check what MWS is reporting to Moab, go to the URL:

```
http://<MWS host>:8080/mws/rest/plugins/all/rm/cluster-query[?api-version=3]
```

To check what your plugin instance is reporting to Moab, use the URL:

```
http://<MWS host>:8080/mws/rest/plugins/<instance-name>/rm/cluster-query[?api-
version=3]
```

If the power is reported in your instance but not to Moab, please increase the precedence of the Moab RM plugin.

Configure the MWS RM in Moab

First, the following lines must be in the Moab Workload Manager configuration file or one of its included files:

```
RMCFG[mws]                          TYPE=MWS
RMCFG[mws]                          FLAGS=UserSpaceIsSeparate
RMCFG[mws]                          BASEURL=http://<mws host>:8080/mws
```

Next, edit the MWS credential information in the Moab private configuration file (`/opt/moab/etc/moab-private.cfg`, by default). Here are the default values:

```
CLIENTCFG[RM:mws] USERNAME=moab-admin PASSWORD=changeme!
```

For more information see the Resource manager queries section in the MWS documentation.

Configure Torque with tomcat administrator

The plugin assumes that Torque is installed on the same host as MWS and that tomcat is an administrator. This can be verified with qmgr. Run the command:

```
qmgr -c 's s managers += tomcat@<mws_host>'
```

For more information see the Specifying non-root administrators section of the Torque documentation.

Make sure the Node and Power scripts work first.

The default scripts are included in `/opt/moab/tools/mws/power_management` and have their own documentation with the `-h` option. They need to have a file that maps each node in the Moab cluster to the IPMI address that the script will need to call using ipmitool. It also needs a file that includes the IPMI password. After that is provided and ipmitool is installed and working, the scripts will successfully implement the interface needed for this plugin.

Related Topics

- pbsnodes -m in the *Torque Resource Manager 9.0.0 Administrator Guide*
- Green computing overview in the *Moab Workload Manager 9.0.0 Administrator Guide*

OpenStack Plugin

The OpenStack plugin allows Elastic Computing instances to be provisioned and de-provisioned using an OpenStack service provider. This enables Moab Workload Manager to "burst" to an OpenStack cloud in order to add or remove nodes dynamically based on policies and workload. See Elastic Computing Overview for more information.

To use the OpenStack plugin, Moab Workload Manager (MWM) must be properly configured. See Integration with a Private OpenStack Cloud.

Create an OpenStack Plugin

To create an OpenStack plugin, see Creating a Plugin. During plugin creation, refer to the Configuration section.

Configuration

Configuration Parameters

Name	Key	Required	Type	Description
OpenStack Endpoint	osEndpoint	Yes	String	Endpoint URL to connect to OpenStack.
OpenStack Username	osUsername	Yes	String	OpenStack username.
OpenStack Password	osPassword	Yes	String	OpenStack password.
OpenStack Tenant	osTenant	Yes	String	OpenStack tenant that will contain created VMs.
OpenStack Flavor Name	osFlavorName	Yes	String	OpenStack flavor name to use for new VMs.
OpenStack Image Name	osImageName	Yes	String	OpenStack image name to use for new VMs.
OpenStack Keypair Name	osKeyPairName	No	String	OpenStack keypair name to use for new VMs.
OpenStack Customization Script	osInitScript	No	String	OpenStack customization script (also called user data) to use for new VMs. This may be used to install and configure RM agents (such as the Torque pbs_mom) on the provisioned servers.
Match Image Prefix	matchImagePrefix	Yes	Boolean	If true, the first OpenStack image starting with the specified name will be used; if false, the full name will be matched.

Name	Key	Required	Type	Description
Use Bootable Image	useBootableImage	Yes	Boolean	If true, non-bootable images will be ignored when searching for a matching image; if false, no bootable checking will be done.
Use Snapshot	useSnapshot	Yes	Boolean	If true, the image used must be a snapshot; if false, the image must not be a snapshot.
OpenStack VLAN Name	osVlanName	No	String	VLAN name configured in OpenStack that contains the IP that Moab should use. If left empty, the first IP address for the first network will be used.
OpenStack Instance Name Pattern	osInstanceNamePattern	Yes	String	Pattern to use for new VM instance names. This must contain the server number token and at least one of the date or request ID tokens.
Active Timeout (s)	activeTimeoutSeconds	Yes	String	Number of seconds to wait for a VM to be active.
Delete Timeout (s)	deleteTimeoutSeconds	Yes	String	Number of seconds to wait for a VM to be deleted successfully.
Maximum Concurrent Requests Limit	maxRequestLimit	Yes	String	Maximum number of OpenStack requests to execute at one time.

Web Services

Elastic Compute Trigger (Secured)

Resource URLs

Resource
`/rest/plugins/<pluginId>/services/triggerElastic`
`/rest/plugins/<pluginId>/services/triggerElastic`

URL Parameters

Parameter	Name	Type	Description
requestId	Request ID	String	Request ID for VM naming and logging purposes.
serverCount	Server Count	Integer	Number of VMs to provision for this request.

Response Fields

Field	Name	Type	Description
id	ID	String	OpenStack ID of the provisioned server.
name	Name	String	OpenStack (unqualified) name of the provisioned server.
ipAddress	IP Address	String	IP address of the provisioned server.
powerState	Power State	String	Power state of the provisioned server (Active or Unknown).

Additional Information

This web service provisions OpenStack VMs based on request parameters. VMs are provisioned in parallel and information on the new servers is rendered as a JSON array in the output. If any step of the provisioning fails for any server, all servers are destroyed immediately before returning an error.

This should only be called as part of processing an elastic compute trigger.

Node End Trigger (Secured)

Resource URLs

Resource
`/rest/plugins/<pluginId>/services/triggerNodeEnd`
`/rest/plugins/<pluginId>/services/trigger-node-end`
`/rest/plugins/<pluginId>/services/triggerNodeEnd/<id>`
`/rest/plugins/<pluginId>/services/trigger-node-end/<id>`

URL Parameters

Parameter	Name	Type	Description
id	Server Name	String	Name of the VM to delete.

Response Fields

Field	Name	Type	Description
messages	Message(s) describing the result of the delete operation	List of Strings	

Additional Information

This web service deletes OpenStack VMs based on the request parameter.

This should only be called as part of processing a node end trigger.

Troubleshooting

The OpenStack plugin logs all errors and warnings to the MWS log file, which is /opt/mws/log/mws.log by default. The stacktrace.log file, which is located in the same directory as mws.log, may also be helpful in diagnosing problems.

ViewpointQueryHelper Plugin

This plugin is required if Viewpoint is part of your configuration.

This plugin provides a web service that allows Viewpoint to query the Moab Workload Manager Insight database.

Configuration Parameters

Name	Required	Type	Description
host	Yes	String	The hostname or IP address of the machine where MongoDB is running.
database	Yes	String	The name of the MongoDB database that Insight writes to.
port	Yes	Integer	The port the MongoDB database is running on.

Name	Required	Type	Description
user	Yes	String	The username with which to connect to MongoDB.
password	Yes	String	The password associated with the username parameter.

Permissions

Unless otherwise specified, all web services in the plugin must be called by an authenticated user who has the *read-insight-privileged* permission. This permission exists in MWS HPC environments and is granted by default to user that have the *HPCAdmin* role.

> The *read-insight-privileged* permission and the *HPCAdmin* role are created when MWS starts with the configuration option *mws.suite* set to "HPC".

Queries

Global URL Parameters

The following URL parameters are available on most web service calls. See the web service call documentation for specifics. With the exception of api-version all global URL parameters are optional.

Parameter	Type	Required	Comment	Example
api-version	JSON Object	Yes	Request a specific API version.	api-version=3
pretty	Boolean	No	Controls pretty printing of output.	pretty=true
query	JSON Object	No	Causes only results that meet filtering conditions to be returned.	query={"configured_processors":4}
sort	JSON Object	No	Determines how results will be sorted.	sort={"node_id":1}
max	Integer	No	The max number of results to return.	max=10

Parameter	Type	Required	Comment	Example
offset	Integer	No	How many results to skip (allows paging).	offset=30

Example 7-1: Example:

```
GET http://localhost:8080/mws/rest/plugins/viewpoint-query-
helper/services/queryNodeList?api-version=3&pretty=true&start_datetime=2014-11-
24%2000:00:00%20MST&end_datetime=2014-11-25%2000:00:00%20MST&query={state:%22BUSY%22}
&sort={node_name:1}&max=10&offset=20
```

Filtering

Additional filtering conditions are possible using MongoDB comparison and logical operators (http://docs.mongodb.org/manual/reference/operator/query/) on selected fields.

See the Queryable/Sortable columns in the Response Fields tables that follow to determine if a particular field is legal to be included in a query.

In the queryNodeList service you can show only nodes with an ID of 4, 5, or 6.

```
query={
    "node_id":{"$in":[4,5,6]},
    }
```

You can query on more than one condition and only nodes that meet all conditions will be shown. For example, to show only nodes in state RUNNING whose job_count is 3 with exactly 4 configured_processors:

```
query={
    "job_count":3,
    "state":"RUNNING",
    "configured_processors":4
    }
```

It is also possible to show only nodes with a field whose value is in a finite set of values using the $in operator. For example, to show only nodes whose state is either BUSY, IDLE, or DOWN:

```
query={
    "state":{"$in":["BUSY","IDLE","DOWN"]}
    }
```

It is also possible to query using greater than and less than comparisons for numeric fields.

Comparator	Encoded
>	$gt
>=	$gte
<	$lt
<=	$lte

To query for nodes whose job_count is greater than 2:

```
query={
    "job_count":{"$gt":2}
    }
```

To query for nodes whose job_count is less than or equal to 15:

```
query={
    "job_count":{"$lte":15}
    }
```

To query for nodes whose job_count is between 2 and 15 inclusive:

```
query={
    "$and":[{"job_count":{"$gte":2}},{"job_count":{"$lte":15}}]
}
```

Sorting

Each request may optionally specify how the results are to be sorted. Sorting must be done based on one of the columns in the results (e.g. node_id, node_name, state, etc.). Sorting can be done in ascending or descending order. To sort in ascending order use a 1.

To sort by node_id in ascending order:

```
sort={node_id:1}
```

To sort by node_id in descending order:

```
sort={node_id:-1}
```

See the Queryable/Sortable column in the Response Fields tables below to determine if a particular field is legal to be sorted by.

Max and Offset

The max and offset parameters limit the number of results returned and allow Viewpoint to page through the results.

To return only 10 results:

```
max=10
```

To skip the first 30 results and return results 31-40:

```
max=10&offset=30
```

Web Services

Query Average Procs Per Job (Secured)

Used by the Viewpoint Resource Job Timeline page to return the average number of procs used by jobs that ran at any point between the start and end time.

Resource URLs

Resource
/rest/plugins/<pluginId>/services/queryAverageProcsPerJob
/rest/plugins/<pluginId>/services/query-average-procs-per-job

URL Parameters

Parameter	Type	Required	Comment	Example
start_datetime	Date	Yes	The start of the time window displayed by the Resource Job Timeline page.	2014-11-21 15:00:00 UTC
end_datetime	Date	Yes	The end of the time window displayed by the Resource Job Timeline page.	2014-11-21 19:00:00 UTC

The query, sort, max, and offset Global URL Parameters are ignored by this web service. See Global URL Parameters on page 416 for additional URL parameters.

Response Fields

Field	Type	Queryable/Sortable	Description
avg	Float	No	The average number of processors used by jobs that ran in the time window

Query Jobs Per Node (Secured)

Used by the Viewpoint Resource Job Timeline page to return a list where each result represents the portion of a job that ran on a particular node.

Resource URLs

The result is an array of JSON object. Each object in the response has the following fields:

Resource
/rest/plugins/<pluginId>/services/queryJobsPerNode
/rest/plugins/<pluginId>/services/query-jobs-per-node

URL Parameters

Parameter	Type	Required	Comment	Example
start_datetime	Date	Yes	The start of the time window displayed by the Resource Job Timeline page.	2014-11-21 15:00:00 UTC
end_datetime	Date	Yes	The end of the time window displayed by the Resource Job Timeline page.	2014-11-21 19:00:00 UTC

See Global URL Parameters on page 416 for additional URL parameters.

Response Fields

The result is an array of JSON objects. Each object in the response has the following fields:

Field	Type	Queryable/Sortable	Description
job_id	Integer	Yes	The Insight database ID for this job.
job_name	String	Yes	The name of this job according to Moab Workload Manager.
node_id	Integer	Yes	The Insight database ID for one of the nodes this job was running on.
node_name	String	Yes	The name of the node according to Moab Workload Manager.
start_datetime	Date	Yes	The date/time the job started.
end_datetime	Date	Yes	The date/time the job ended.
processors_on_node	Integer	Yes	The number of processors this job was running on this particular node.
node_count	Integer	Yes	The number of nodes this job was running on.
wallclock_limit_seconds	Integer	Yes	The wallclock limit of this job in seconds.

Query Node List (Secured)

Used by the Viewpoint Resource Job Timeline page to return a list of nodes.

Resource URLs

Resource
/rest/plugins/<pluginId>/services/queryNodeList
/rest/plugins/<pluginId>/services/query-node-list

URL Parameters

Parameter	Type	Required	Comment	Example
start_datetime	Date	Yes	The start of the time window displayed by the Resource Job Timeline page.	2014-11-21 15:00:00 UTC
end_datetime	Date	Yes	The end of the time window displayed by the Resource Job Timeline page.	2014-11-21 19:00:00 UTC

See Global URL Parameters on page 416 for additional URL parameters.

Response Fields

The result is an array of JSON objects. Each object in the response has the following fields:

Field	Type	Queryable/Sortable	Description
node_id	Integer	Yes	The Insight database ID for the node.
node_name	String	Yes	The name of the node according to Moab Workload Manager.
state	String	Yes	The current state of the node.
configured_processors	Integer	Yes	The number of configured processors on the node.
job_count	Integer	Yes	The number of jobs that were running at any time in the time window.
initial_state	String	No	The state of the node at the start of the time window.
state_transitions	List of JSON Objects	No	A list of all node state transitions in the time window.

Filtering and sorting on either the initial_state or the state_transitions field is not possible.

Query Reservations Per Node (Secured)

Used by the Viewpoint Resource Job Timeline page to return a list where each result represents the portion of a reservation that reserved a particular node. Thus a reservation on three different nodes would be represented as three different results.

Resource URLs

Resource
/rest/plugins/<pluginId>/services/queryReservationsPerNode
/rest/plugins/<pluginId>/services/query-reservations-per-node

URL Parameters

Parameter	Type	Required	Comment	Example
start_datetime	Date	Yes	The start of the time window displayed by the Resource Job Timeline page.	2014-11-21 15:00:00 UTC
end_datetime	Date	Yes	The end of the time window displayed by the Resource Job Timeline page.	2014-11-21 19:00:00 UTC

See Global URL Parameters on page 416 for additional URL parameters.

Response Fields

The result is an array of JSON objects. Each object in the response has the following fields:

Field	Type	Queryable/Sortable	Description
reservation_id	Integer	Yes	The Insight database ID for this reservation.
reservation_name	String	Yes	The name of this reservation according to Moab Workload Manager.

Field	Type	Queryable/Sortable	Description
node_id	Integer	Yes	The Insight database ID for one of the nodes this reservation was on.
node_name	String	Yes	The name of the node according to Moab Workload Manager.
start_datetime	Date	Yes	The date/time the reservation started.
end_datetime	Date	Yes	The date/time the reservation ended.
processors_on_node	Integer	Yes	The number of processors this reservation reserved on this particular node.

Query Workload (Secured)

Used by the Viewpoint Home and Workload pages to return a list of jobs.

- If the user calling this web service is an administrator, all jobs will be returned that match the filter and paging criteria the user has specified. A user is considered to be an administrator if the user has the *read-insight-privileged* permission.
- If the user does not have the read-insight-privileged but *does* have the *read-insight-user*, the user will be allowed to see only jobs that he or she submitted. More specifically, jobs for which the user_name field does not match the username of the user calling the web service will be filtered out.
- Users that do not have either *read-insight-privileged* or *read-insight-user* will not be allowed to call this web service.

Resource URLs

Resource
/rest/plugins/<pluginId>/services/queryReservationsPerNode
/rest/plugins/<pluginId>/services/query-reservations-per-node

URL Parameters

See Global URL Parameters on page 416.

Response Fields

The result is an array of JSON objects. Each object in the response has the following fields:

Field	Type	Queryable/Sortable	Description
job_id	Integer	Yes	A unique identifier for the job.
job_name	String	Yes	The name of the job according to Moab Workload Manager.
job_custom_name	String	Yes	The user specified name for the job.
job_start_datetime	Date	Yes	The date/time the job started.
end_datetime	Date	Yes	The date/time the job ended.
processors_count	Integer	Yes	The number of processors required by the job.
node_count	Integer	Yes	The number of nodes used by the job.
user_name	String	Yes	The name of the user that submitted the job.
job_state	String	Yes	The state of the job.
holds	String	Yes	A comma separated list of the holds on the job.
wallclock_seconds	Integer	Yes	The number of wallclock seconds required by the job.

Field	Type	Queryable/Sortable	Description
job_sort	Integer	Yes	The default sort criteria for the jobs on the Viewpoint Home page.

Chapter 8 References

In this chapter:

- Client Code Samples
- Configuration
- Resources reference

Client Code Samples

The code samples contained in this section of the reference material are provided to help quick start integration with MWS. They are provided as a convenience and not as fully developed APIs.

All examples use the default configuration of MWS, including the default username and password, and assume that MWSis deployed at `http://localhost:8080/mws`.

This section contains these topics:

- Javascript Code Samples
- PHP Code Samples
- Perl Code Samples
- Python Code Examples
- curl Samples

Related Topics

- Configuration

Javascript Code Samples

When utilizing Javascript to interact with MWS, it is recommended to use libraries that provide a simple browser-independent syntax for performing REST calls. It must also be noted that it is not recommended to make calls to MWS directly from client- side Javascript, as this will contain the username and password for MWS and could potentially be retrieved and utilized by a malicious user.

Examples for specific libraries are given below.

jQuery

The jQuery `ajax` function makes it extremely simple to perform all REST calls. There are also some shorthand methods which utilize the `ajax` method in a simple manner—such as `getJSON` and `post`—but these do not have the ability to specify authentication parameters, which is why they are not used below.

> In all examples given, it is assumed that the `$` variable is mapped to `jQuery`, which is the default configuration of jQuery.

GET

```
$.ajax({
  url: "http://localhost:8080/mws/rest/jobs",
  dataType: 'json',
  username: 'admin',
  password: 'secret',
  success: new function(data) {
    $('.result').html(data);
    alert("GET was successful");
  )
});
```

POST

```
$.ajax({
  url: "http://localhost:8080/mws/rest/jobs",
  data: {
    "commandFile":"/tmp/test.sh",
    "initialWorkingDirectory":"/tmp",
    "user":"adaptive",
    "requirements":[{"requiredNodeCountMinimum":1}]
  }
  dataType: 'json',
  username: 'admin',
  password: 'secret',
  success: new function(data) {
    $('.result').html(data);
    alert("GET was successful");
  )
});
```

PUT

```
$.ajax({
  type: 'PUT',
  url: 'http://localhost:8080/mws/rest/jobs/Moab.1',
  data: {"holds":["user"]},
  dataType: "json",
  username: 'admin',
  password: 'secret',
  success: function(data) {
    $('.result').html(data);
    alert("PUT was successful");
  }
});
```

DELETE

```
$.ajax({
  type: 'DELETE',
  url: 'http://localhost:8080/mws/rest/jobs/Moab.1',
  username: 'admin',
  password: 'secret',
  success: function(data) {
    $('.result').html(data);
    alert("DELETE was successful");
  }
});
```

Related Topics

- Client code samples

PHP Code Samples

All of these examples make use of the cURL PHP Extension. While this library is easy to use for GET and POST requests, it is somewhat more difficult to use for PUT and DELETE requests. Each will be covered in this topic.

Notice the use of the JSON PHP Extension's `json_decode` and `json_encode` functions.

GET

```
<?php
$baseUrl = "http://localhost:8080/mws/rest";
$resource = "/jobs";
$username = "admin";
$password = "secret";
$ch = curl_init();
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");
curl_setopt($ch, CURLOPT_URL, "$baseUrl$resource");

$responseBody = curl_exec($ch);
$responseInfo = curl_getinfo($ch);
curl_close($ch);

if ($responseInfo["http_code"]!=200 && $responseInfo["http_code"]!=201) {
  print_r($responseInfo);
  echo $responseBody;
} else {
  print_r(json_decode($responseBody));
}
?>
```

POST

```
<?php
$baseUrl = "http://localhost:8080/mws/rest";
$resource = "/jobs";
$username = "admin";
$password = "secret";
$requestPayload = array(
  "commandFile"=>"/tmp/test.sh",
  "initialWorkingDirectory"=>"/tmp",
  "user"=>"adaptive",
  "requirements"=>array(
      array("requiredNodeCountMinimum"=>1)
  )
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Content-Type: application/json"));
curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");
curl_setopt($ch, CURLOPT_URL, "$baseUrl$resource");
// Setup POST request
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($requestPayload));
curl_setopt($ch, CURLOPT_POST, 1);

$responseBody = curl_exec($ch);
$responseInfo = curl_getinfo($ch);
curl_close($ch);

if ($responseInfo["http_code"]!=200 && $responseInfo["http_code"]!=201) {
  print_r($responseInfo);
  echo $responseBody;
} else {
  print_r(json_decode($responseBody));
}
?>
```

PUT

```
<?php
$baseUrl = "http://localhost:8080/mws/rest";
$resource = "/jobs/Moab.1";
$username = "admin";
$password = "secret";
$requestPayload = array(
  "holds"=>array("user")
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Content-Type: application/json"));
curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");
curl_setopt($ch, CURLOPT_URL, "$baseUrl$resource");
// Setup PUT request
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($requestPayload));
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "PUT");

$responseBody = curl_exec($ch);
$responseInfo = curl_getinfo($ch);
curl_close($ch);

if ($responseInfo["http_code"]!=200 && $responseInfo["http_code"]!=201) {
  print_r($responseInfo);
  echo $responseBody;
} else {
  print_r(json_decode($responseBody));
}
```

DELETE

```
<?php
$baseUrl = "http://localhost:8080/mws/rest";
$resource = "/jobs/Moab.1";
$username = "admin";
$password = "secret";
$requestPayload = array(
  "holds"=>array("user")
);
$ch = curl_init();
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_TIMEOUT, 10);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Content-Type: application/json"));
curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");
curl_setopt($ch, CURLOPT_URL, "$baseUrl$resource");
// Setup DELETE request
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "DELETE");

$responseBody = curl_exec($ch);
$responseInfo = curl_getinfo($ch);
curl_close($ch);

if ($responseInfo["http_code"]!=200 && $responseInfo["http_code"]!=201) {
  print_r($responseInfo);
  echo $responseBody;
} else {
  print_r(json_decode($responseBody));
}
```

Related Topics

- Client Code Samples

Perl Code Samples

These examples all utilize the `LWP::UserAgent` module, which must be installed before running them.

GET

```
#!/usr/bin/perl -w
use strict;
use warnings;
# Create a user agent object
use LWP::UserAgent;
my $ua = LWP::UserAgent->new;
$ua->agent("MyApp/0.1");

# Create a request
my $req = HTTP::Request->new(GET => 'http://localhost:8080/mws/rest/images');
$req->content_type('application/json');
$req->authorization_basic("admin", "secret");

# Pass request to the user agent and get a response back
my $res = $ua->request($req);

# Check the outcome of the response
if ($res->is_success) {
  print $res->content;
} else {
  print $res->status_line, "n";
}
```

POST

```
#!/usr/bin/perl -w
use strict;
use warnings;
# Create a user agent object
use LWP::UserAgent;
my $ua = LWP::UserAgent->new;
$ua->agent("MyApp/0.1");

# Create a request
my $req = HTTP::Request->new(POST => 'http://localhost:8080/mws/rest/images');
$req->content_type('application/json');
$req->authorization_basic("admin", "secret");
$req->content('
{"profile":"compute","osVersion":"6","name":"centos6stateless","hypervisor":0,"archite
cture":"x86_64","osName":"centos","osType":"linux","type":"stateless"}');

# Pass request to the user agent and get a response back
my $res = $ua->request($req);

# Check the outcome of the response
if ($res->is_success) {
  print $res->content;
} else {
  print $res->status_line, "n";
}
```

PUT

```
#!/usr/bin/perl -w
use strict;
use warnings;
# Create a user agent object
use LWP::UserAgent;
my $ua = LWP::UserAgent->new;
$ua->agent("MyApp/0.1");

# Create a request
my $req = HTTP::Request->new(PUT => 'http://localhost:8080/mws/rest/images/centos6-
stateless');
$req->content_type('application/json');
$req->authorization_basic("admin", "secret");
$req->content('{"osVersion":"6.6"}');

# Pass request to the user agent and get a response back
my $res = $ua->request($req);

# Check the outcome of the response
if ($res->is_success) {
  print $res->content;
} else {
  print $res->status_line, "n";
}
```

DELETE

```
#!/usr/bin/perl -w
use strict;
use warnings;
# Create a user agent object
use LWP::UserAgent;
my $ua = LWP::UserAgent->new;
$ua->agent("MyApp/0.1");

# Create a request
my $req = HTTP::Request->new(DELETE => 'http://localhost:8080/mws/rest/images/centos6-
stateless');
$req->content_type('application/json');
$req->authorization_basic("admin", "secret");

# Pass request to the user agent and get a response back
my $res = $ua->request($req);

# Check the outcome of the response
if ($res->is_success) {
  print $res->content;
} else {
  print $res->status_line, "n";
}
```

Related Topics

- Client Code Samples

Python Code Examples

Notice the use of the json module to build a Python object from the return JSON data. If you want, you an also use `json.dumps` to create a JSON string from a Python object.

Simple request (GET)

```
import httplib
import base64
import string
import json
def get(base, port, url):
       conn = httplib.HTTPConnection(base, port, timeout=60)
       conn.request('GET', url, None, { 'Authorization' : 'Basic '+string.strip
(base64.encodestring('admin:secret'))})
       return conn.getresponse().read()

data = get("localhost", 8080, "/mws/rest/jobs?format=json")
print json.loads(data)
```

Complex request (POST)

```
import httplib
import base64
import string
import json
def post(base, port, url, payload):
       conn = httplib.HTTPConnection(base, port, timeout=60)
       conn.request('POST', url, payload, { 'Authorization' : 'Basic '+string.strip
(base64.encodestring('admin:secret')), 'Content-Type' : 'application/json' })
       r = conn.getresponse()
       return r.read()

# Note that json.dumps may also be used to create the json string from a python object
data = post("localhost", 8080, "/mws/rest/jobs", '
{"commandFile":"/tmp/test.sh","initialWorkingDirectory":"/tmp","user":"adaptive","requ
irements":[{"requiredNodeCountMinimum":1}]}')
print json.loads(data)
```

Related Topics

- Client Code Samples

curl Samples

Unlike the other code samples given in this section, these samples are simple commands that can be run from any server command line with the `curl` program installed in order to communicate with MWS.

GET

```
curl -u admin:secret -X GET -H "Content-Type: application/json"
http://localhost:8080/mws/rest/jobs
```

POST

```
curl -u admin:secret -X POST -H "Content-Type: application/json"
http://localhost:8080/mws/rest/jobs
  -d '
{"commandFile":"/tmp/test.sh","initialWorkingDirectory":"/tmp","user":"adaptive","requ
irements":[{"requiredNodeCountMinimum":1}]}'
```

PUT

```
curl -u admin:secret -X PUT -H "Content-Type: application/json"
http://localhost:8080/mws/rest/jobs
  -d '{"holds":["user"]}'
```

DELETE

```
curl -u admin:secret -X DELETE -H "Content-Type: application/json"
http://localhost:8080/mws/rest/jobs
```

Related Topics

- Client Code Samples

Configuration

These properties can be modified by setting the appropriate values in the `mws-config.groovy` file. This file is located in `MWS_HOME/etc/` or `/opt/mws/etc/` by default as explained in Configuring Moab Web Services.

> The configuration file is read not only on startup, but also each time it is changed. Several properties, including those for Moab Workload Manager (`moab`), Moab Accounting Manager (`mam`), Mongo (`grails.mongo`), and authentication (`auth`) are processed after each change and can affect the runtime behavior of MWS.
>
> Configuration files can also be placed in the `MWS_HOME/etc/mws.d` directory. Any configuration files here get merged with `MWS_HOME/etc/mws-config.groovy`. In case of conflict, the configuration in `MWS_HOME/etc/mws.d` takes precedence.

Configuration reference

For all possible values that can be set, please see the Grails reference guide. For project specific settings (usually the only ones you'll need to change), you may set the following properties:

Property	Type	Default	Description
auth.defaultUser.password	String	changeme!	Unencoded password of the default admin user.
auth.defaultUser.username	String	moab-admin	Username of the default admin user (only created if no other users exist).
grails.mongo.host	String	127.0.0.1	The MongoDB host to use (Note that MongoDB runs on `127.0.0.1` and *not* `localhost` by default).

Property	Type	Default	Description
dataSource_insight.password	String	changeme!	The password for the username used to log in to the Insight database.
dataSource_insight.url	String	jdbc:postgresql://127.0.0.1:5432/insight	The JDBC URL for the Insight database. For more information, see Configuring Moab Web Services.
dataSource_insight.username	String	mws	The username used to log into the Insight database.
grails.mongo.port	Integer	27017	The MongoDB port to use.
grails.mongo.replicaSet	List of Strings	n/a	The MongoDB replica set servers to use (for example, `["moab1:27017","moab2:27017"]`); note that `grails.mongo.host` *must* be set to `null` to use this option.
grails.mongo.databaseName	String	mws	The MongoDB database name to use.
grails.mongo.username	String	-	(Optional) The username to use when connecting to MongoDB.
grails.mongo.password	String	-	(Optional) The password to use when connecting to MongoDB.
grails.mongo.options.connectionsPerHost	Integer	50	The number of connections allowed per host.

Property	Type	Default	Description
grails.-mongo.-options.threadsAllowedToBlockForConnecti	Integer	5	The number of threads per connection allowed to wait for an available connection.
grails.mongo.options.autoConnectRetry	Boolean	true	Controls whether the system retries automatically on connection errors.
grails.mime.use.accept.header	Boolean	false	When enabled, uses the HTTP Content-Accept header to determine the content type used for return data (JSON only for now).
grails.plugins.springsecurity.basic.realmName	String	Moab Web Services	The HTTP realm used when using basic auth.
grails.plugins.springsecurity.active	Boolean	true	Enables or disables security for MWS as a whole, including all providers.
grails.plugins.springsecurity.useBasicAuth	Boolean	true	Enables or disables basic auth with a simple username/password.
grails.plugins.springsecurity.oauthProvider.active	Boolean	true	Enables or disables the OAuth2 provider.
ldap.baseDNs	List of Strings	-	A list of distinguished names that are the root entries for LDAP searches.
ldap.bindUser	String	-	The distinguished name of the LDAP bind user.

Property	Type	Default	Description
ldap.directory.type	String	-	The type of LDAP directory (for example, "Microsoft Active Directory"). See Configuring Moab Web Services for valid values..
ldap.password	String	-	The password of the LDAP bind user
ldap.port	Integer	-	LDAP server's port
ldap.security.server.certificate	String	-	The filename of the LDAP server's PEM encoded X.509 certificate. See Setting up MWS Security for more information.
ldap.security.type	String	-	How the connection between MWS and LDAP is secured. See Setting up MWS Security for more information.
ldap.server	String	-	LDAP server hostname or IP address
mam.server	String	localhost	Moab Accounting Manager server hostname or IP address
mam.port	Integer	7112	Moab Accounting Manager server's port
mam.secretKey	String	mamsecret	Secret key used to communicate with Moab Accounting Manager

Property	Type	Default	Description
moab.databaseName	String	moab	The name of the MongoDB database to use to retrieve current Moab data; this should match the database setting in Moab.
moab.messageQueue.port	Integer	5570	The port on which Moab publishes ZeroMQ messages.
moab.messageQueue.secretKey	String	-	Used to encrypt and decrypt messages on the message queue using AES. Must be a Base64-enoded 128-bit (16-byte) key. For example: `"1r6RvfqJa6voezy5wAx0hw=="`
moab.port	Integer	42559	Moab server's port
moab.secretKey	String	moabsecret	Secret key used to communicate with Moab. See Moab Configuration.
moab.server	String	localhost	Moab server hostname or IP address
mws.cache.duration.default	Integer	60	The default number of seconds to use for caching objects from Moab. This is only supported in certain objects such as policies.

Property	Type	Default	Description
mws.cache.duration.policy	Integer	180	The number of seconds that the cache for policies is valid. If set to null, the default is used.
mws.certificates.location	String	etc/ssl.crt	The directory (relative or absolute) where plugin certificates are stored. See the Managing SSL Connections.
mws.events.expireAfterSeconds	Integer	2592000	Events older than this many seconds (30 days by default) will be deleted from the database. Effective only with MongoDB 2.2 or later.
mws.health.check.period	Integer	30	The number of seconds in between health checks. Used in creating notification conditions if problems exist in configuration or connections. For more information, see Notification Conditions.
mws.hooks.location	String	hooks	The directory (relative or absolute) where Hooks are stored. See Pre- and Post-Processing Hooks for more information.
mws.plugins.location	String	plugins	The directory (relative or absolute) where Plugins are stored. See About Moab Web Services Plugins for more information.

Property	Type	Default	Description
mws.messageQueue.port	Integer	5564	The port on which MWS publishes ZeroMQ messages.
mws.messageQueue.address	String	-	The IP address on which MWS publishes ZeroMQ messages.
mws.services.hooks.syncInterval	Integer	30	The number of seconds between each time MWS checks for service phase transition hooks that completed or timed out.
mws.services.phases.syncInterval	Integer	14400	The number of seconds between each time MWS checks with Moab Workload Manager to verify that the service phases are correctly synchronized.
mws.suite	String	CLOUD	The suite or context that MWS is running in (see Suite for valid values)
pam.configuration.service	String	-	The name of the PAM configuration file located in `/etc/pam.d`. This parameter and specification tells MWS which PAM configuration file you want to use. For more information, see PAM (Pluggable Authentication Module) Configuration Using mws-config.groovy.

Property	Type	Default	Description
plugins.pluginType	String	-	Default configuration value for the plugin `pluginType` field (see Setting Default Plugin Configuration).
plugins.autoStart	Boolean	true	Default configuration value for the plugin `autoStart` field (see Setting Default Plugin Configuration).
plugins.pollInterval	Integer	30	Default configuration value for the plugin `pollInterval` field (see Setting Default Plugin Configuration).
plugins.config	Map	-	Default configuration value for the plugin `config` field (see Setting Default Plugin Configuration).
plugins.loadInitialPlugins	Boolean	true	If true, loads the initial plugins defined for uploaded or built-in plugin types (see Plugin Projects and Metadata).
plugins.stateConsolidationPolicy	NodeStatePolicy	null	If "optimistic", treats state data optimistically. If "pessimistic", treats state state pessimistically. May be null. See Data Consolidation for more information.

Property	Type	Default	Description
plugins.defaultHypervisorType	String	ESX	This is reported to Moab when a node report references a hypervisor image that does not have the `hypervisorType` or `extensions.xcat.hvType` fields set. See Fields: Images.

Logging reference

The following loggers are available to use for debugging purposes:

Logger	Default	Description
grails.app	debug	Most classes in the main MWS application.
grails.app.bootstrap.BootStrap	debug	Handles startup and initialization of MWS.
com.ace.mws	debug	The base logger for MWS specific functionality not included in other loggers (this comprises very few classes).
grails.app.services.com.ace.mws.plugins.PluginUtilityService	debug	Class for initializing and helper methods of plugins.
com.ace.mws.hooks.HookUtils	debug	Helper class for loading hooks during startup process.
plugins	debug	All MWS plugins (see About Moab Web Services Plugins).
com.ace.mws.plugins	debug	MWS plugin helper class, used to create and initialize plugins.
com.ace.mws.gapi	warn	Base logger for all Moab connections, requests, and responses.

Logger	Default	Description
com.ace.mws.gapi.Connection	info	Logger which controls all requests and responses from Moab.
com.ace.mws.gapi.parsers	info	Loggers for parsers of Moab's data.
com.ace.mws.gapi.serializers	info	Loggers for all serialization from MWS to Moab Wire Protocol.
grails.app.service.grails.plugins.reloadconfig	info	Handles dynamic reloading of configuration files.
net.sf.json	error	JSON and XML processing library.
org.springframework.security	info	Authentication/authorization logger.
org.codehaus.groovy.grails.web.servlet	error	Loggers for request handlers.
org.codehaus.groovy.grails.web.mapping	error	URL mapping.
org.codehaus.groovy.grails.web.mapping.filter	error	URL mapping.
org.codehaus.groovy.grails.plugins	error	All grails plugins (MWS internal).
org.codehaus.groovy.grails.commons	error	Core application and class-loading.

Related Topics

- Configuring Moab Web Services

Resource Reference

Resources reference

This section contains the type and description of all possible fields in each MWS resource object. Because of significant changes in the API introduced between releases, MWS possesses a versioned API. Each resource contains drop-down sections for each API version.

This section contains these topics:

- Fields: Access Control Lists (ACLs)
- Fields: Accounts
- Fields: Allocations
- Fields: Charge Rates
- Fields: Credentials
- Fields: Events
- Fields: Fund Balances
- Fields: Fund Statements
- Fields: Fund Statement Summary
- Fields: Funds
- Fields: Images
- Fields: Job Arrays
- Fields: Job Templates
- Fields: Jobs
- Fields: Liens
- Fields: Metric Types
- Fields: Nodes
- Fields: Notification Conditions
- Fields: Notifications
- Fields: Organizations
- Fields: User's Permissions
- Fields: Plugins
- Fields: Plugin Types
- Fields: Policies

- Fields: Principals
- Fields: Quotes
- Fields: Report Datapoints
- Fields: Report Samples
- Fields: Reports
- Fields: Reservations
- Fields: Resource Types
- Fields: Roles
- Fields: Standing Reservations
- Fields: Transactions
- Fields: Usage Records
- Fields: Users

Related Topics

- Resources Introduction
- Global URL Parameters

Fields: Access Control Lists (ACLs)

See the associated Access Control Lists (ACLs) resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`acl-rules`	Permissions
Hooks filename	`acl-rules.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

AclRule

This class represents a rule that can be in Moab's access control list (ACL) mechanism.

The basic AclRule information is the object's name and type. The type directly maps to an AclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	PUT	Description
affinity	**AclAffinity**	Yes	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	**ComparisonOperator**	Yes	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
type	**AclType**	Yes	The type of the object that is being granted (or denied) access.
value	**String**	Yes	The name of the object that is being granted (or denied) access.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

AclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
PROC	Not supported
QTIME	Not supported

Value	Description
QUEUE	Not supported
RACK	Not supported
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

API version 2

AclRule

This class represents a rule that can be in Moab's access control list (ACL) mechanism.

The basic AclRule information is the object's name and type. The type directly maps to an AclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	PUT	Description
affinity	**AclAffinity**	Yes	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	**ComparisonOperator**	Yes	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
type	**AclType**	Yes	The type of the object that is being granted (or denied) access.
value	**String**	Yes	The name of the object that is being granted (or denied) access.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

AclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
PROC	Not supported
QTIME	Not supported

Value	Description
QUEUE	Not supported
RACK	Not supported
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

Related Topics

- Access Control Lists (ACLs)

Accounting

Fields: Accounts

See the associated Accounting Accounts resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/accounts`	Permissions
Hooks filename	`accounting.accounts.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Account

Users may be designated as members of an account and may be allowed to share its allocations. The user members may be designated as active or inactive, and as an account admin or not an account admin. Default account properties include the description, the organization it is part of, and whether or not it is active. An account's user membership can also be adjusted. By default, a standard user may only query accounts they belong to.

Field Name	Type	Description
id	**String**	The unique account identifier
active	**Boolean**	A boolean indicating whether this account is active or not
creationTime	**Date**	The time this account was created
deleted	**Boolean**	A boolean indicating whether this account is deleted or not
description	**String**	The account description
modificationTime	**Date**	The time this account was last modified
organization	**String**	The organization to which the account belongs
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction
users	**Set<AccountUser>**	The users associated with this account

AccountUser

An account user is a person authorized to use an account.

Field Name	Type	Description
id	**String**	The unique user identifier
active	**Boolean**	A boolean indicating whether this user is active or not

Field Name	Type	Description
admin	**Boolean**	A boolean indicating wheter this user is an admin or not

API version 2

Account

Users may be designated as members of an account and may be allowed to share its allocations. The user members may be designated as active or inactive, and as an account admin or not an account admin. Default account properties include the description, the organization it is part of, and whether or not it is active. An account's user membership can also be adjusted. By default, a standard user may only query accounts they belong to.

Field Name	Type	Description
id	**String**	The unique account identifier
active	**Boolean**	A boolean indicating whether this account is active or not
creationTime	**Date**	The time this account was created
deleted	**Boolean**	A boolean indicating whether this account is deleted or not
description	**String**	The account description
modificationTime	**Date**	The time this account was last modified
organization	**String**	The organization to which the account belongs
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction
users	**Set<AccountUser>**	The users associated with this account

AccountUser

An account user is a person authorized to use an account.

Field Name	Type	Description
id	**String**	The unique user identifier
active	**Boolean**	A boolean indicating whether this user is active or not

Field Name	Type	Description
admin	**Boolean**	A boolean indicating wheter this user is an admin or not

Related Topics

- Accounting Accounts

Fields: Allocations

See the associated Accounting Allocations resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/allocations`	Permissions
Hooks filename	`accounting.allocations.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive

Field Name	Type	Description
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

API version 2

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive

Field Name	Type	Description
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

Related Topics

- Accounting Allocations

Fields: Charge Rates

See the associated Accounting Charge rates resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/charge-rates`	Permissions
Hooks filename	`accounting.charge-rates.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

ChargeRate

Charge rates establish how much to charge for usage. A charge rate consists of its name, an optional value and the amount. Both name and value are primary keys and a charge rate is uniquely defined by both its name and its value. A charge rate value that is null designates the default charge rate.

Field Name	Type	Description
id	**Long**	
amount	**String**	The charge rate amount
creationTime	**Date**	The date this charge rate was created
deleted	**Boolean**	A boolean indicating whether this charge rate is deleted or not
description	**String**	The charge rate description
modificationTime	**Date**	The date this charge rate was last modified
name	**String**	The charge rate name
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction
value	**String**	The charge rate value. This will be null for default charge rates.

API version 2

ChargeRate

Charge rates establish how much to charge for usage. A charge rate consists of its name, an optional value and the amount. Both name and value are primary keys and a charge rate is uniquely defined by both its name and its value. A charge rate value that is null designates the default charge rate.

Field Name	Type	Description
id	**Long**	
amount	**String**	The charge rate amount
creationTime	**Date**	The date this charge rate was created
deleted	**Boolean**	A boolean indicating whether this charge rate is deleted or not
description	**String**	The charge rate description
modificationTime	**Date**	The date this charge rate was last modified
name	**String**	The charge rate name
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction
value	**String**	The charge rate value. This will be null for default charge rates.

Related Topics

- Accounting Charge rates

Fields: Fund Balances

See the associated Accounting Funds resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/funds/balances`	Permissions
Hooks filename	`accounting.funds.balances.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

FundBalance

Represents a report of fund balance.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	The total adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	Allocations associated with this fund
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
available	**BigDecimal**	The total amount available for charging. amount - reserved + creditLimit
balance	**BigDecimal**	The allocation total not blocked by liens. amount - reserved
capacity	**BigDecimal**	The total amount allocated via deposits and credit limits. allocated + creditLimit
creationTime	**Date**	Date this fund was created
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
percentRemaining	**Double**	The percentage of allocation remaining. amount * 100 / allocated

Field Name	Type	Description
percentUsed	**Double**	The percentage of allocated used. used * 100 / allocated
reserved	**BigDecimal**	The sum of active lien amounts against this fund
used	**BigDecimal**	The total amount used this allocation cycle. allocated - amount

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created

Field Name	Type	Description
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

API version 2

FundBalance

Represents a report of fund balance.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	The total adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	Allocations associated with this fund
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
available	**BigDecimal**	The total amount available for charging. amount - reserved + creditLimit
balance	**BigDecimal**	The allocation total not blocked by liens. amount - reserved
capacity	**BigDecimal**	The total amount allocated via deposits and credit limits. allocated + creditLimit
creationTime	**Date**	Date this fund was created
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
percentRemaining	**Double**	The percentage of allocation remaining. amount * 100 / allocated

Field Name	Type	Description
percentUsed	**Double**	The percentage of allocated used. used * 100 / allocated
reserved	**BigDecimal**	The sum of active lien amounts against this fund
used	**BigDecimal**	The total amount used this allocation cycle. allocated - amount

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created

Field Name	Type	Description
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

Related Topics

- Accounting Funds

Fields: Fund Statement Summary

See the associated Accounting Funds resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/funds/reports/statement`	Permissions
Hooks filename	`accounting.funds.reports.statement.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

FundStatementSummary

An fund statement summary is related to and quite similar to the FundStatement report, but differs in the transactions field by using the FundTransactionSummary.

Field Name	Type	Description
id	**Long**	
endBalance	**BigDecimal**	The balance of the funds at the endTime of the statement
endTime	**Date**	The ending time that the statement covers
funds	**Set<Fund>**	The funds that this statement covers. Only a sub-set of the full fund fields are available from this property. This includes id, name, priority, description, and creationTime.
generationTime	**Date**	The date that the statement report was generated
startBalance	**BigDecimal**	The balance of the funds at the startTime of the statement
startTime	**Date**	The starting time that the statement covers
totalCredits	**BigDecimal**	The total number of credits that occurred during the time period that the statement covers
totalDebits	**BigDecimal**	The total number of debits that occurred during the time period that the statement covers
transactions	**Set<FundTransactionSummary>**	Summaries of the specific transactions which occurred during the time period that this statement covers.

Fund

A fund is a container for a time-bounded reference currency called credits for which the usage is restricted by constraints that define how the credits must be used. Much like with a bank, an fund is a repository for these

resource or service credits which are added through deposits and debited through withdrawals and charges. Each fund has a set of constraints designating which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, Feature, etc.). Fund constraints may also be negated with an exclamation point leading the constraint value.

When credits are deposited into an fund, they are associated with a time period within which they are valid. These time-bounded pools of credits are known as allocations. (An allocation is a pool of billable units associated with an fund for use during a particular time period.) By using multiple allocations that expire in regular intervals it is possible to implement a use-it-or-lose-it policy and establish an allocation cycle.

Funds may be nested. Hierarchically nested funds may be useful for the delegation of management roles and responsibilities. Deposit shares may be established that assist to automate a trickle-down effect for funds deposited at higher level funds. Additionally, an optional overflow feature allows charges against lower level funds to trickle up the hierarchy.

Funds may have an arbitrary name which is not necessarily unique for the fund. Funds may also have a priority which will influence the order of fund selection when charging.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	Total Adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	The allocations associated with this fund.
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
creationTime	**Date**	Date this fund was created
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
defaultDeposit	**String**	The default deposit amount

Field Name	Type	Description
deleted	**Boolean**	A boolean indicating whether this fund is deleted or not
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
initialDeposit	**BigDecimal**	The initial deposit amount
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
priority	**Integer**	The fund priority
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

FundTransactionSummary

Represents a Moab Accounting Manager transaction summary, which is a consolidated view of multiple transactions. The transactions are grouped by object and action, and a total count is given for the summary.

Field Name	Type	Description
id	**Long**	
count	**Long**	The number of transactions in this grouping of object and action
action	**String**	Action name for the transaction
amount	**BigDecimal**	Amount of the transaction. A positive or amount signifies a credit, while a negative or zero amount signifies a debit.
object	**String**	Object's name associated with the transaction

API version 2

FundStatementSummary

An fund statement summary is related to and quite similar to the FundStatement report, but differs in the transactions field by using the FundTransactionSummary.

Field Name	Type	Description
id	**Long**	
endBalance	**BigDecimal**	The balance of the funds at the endTime of the statement
endTime	**Date**	The ending time that the statement covers
funds	**Set<Fund>**	The funds that this statement covers. Only a sub-set of the full fund fields are available from this property. This includes id, name, priority, description, and creationTime.
generationTime	**Date**	The date that the statement report was generated
startBalance	**BigDecimal**	The balance of the funds at the startTime of the statement
startTime	**Date**	The starting time that the statement covers
totalCredits	**BigDecimal**	The total number of credits that occurred during the time period that the statement covers
totalDebits	**BigDecimal**	The total number of debits that occurred during the time period that the statement covers
transactions	**Set<FundTransactionSummary>**	Summaries of the specific transactions which occurred during the time period that this statement covers.

Fund

A fund is a container for a time-bounded reference currency called credits for which the usage is restricted by constraints that define how the credits must be used. Much like with a bank, an fund is a repository for these

resource or service credits which are added through deposits and debited through withdrawals and charges. Each fund has a set of constraints designating which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, Feature, etc.). Fund constraints may also be negated with an exclamation point leading the constraint value.

When credits are deposited into an fund, they are associated with a time period within which they are valid. These time-bounded pools of credits are known as allocations. (An allocation is a pool of billable units associated with an fund for use during a particular time period.) By using multiple allocations that expire in regular intervals it is possible to implement a use-it-or-lose-it policy and establish an allocation cycle.

Funds may be nested. Hierarchically nested funds may be useful for the delegation of management roles and responsibilities. Deposit shares may be established that assist to automate a trickle-down effect for funds deposited at higher level funds. Additionally, an optional overflow feature allows charges against lower level funds to trickle up the hierarchy.

Funds may have an arbitrary name which is not necessarily unique for the fund. Funds may also have a priority which will influence the order of fund selection when charging.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	Total Adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	The allocations associated with this fund.
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
creationTime	**Date**	Date this fund was created
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
defaultDeposit	**String**	The default deposit amount

Field Name	Type	Description
deleted	**Boolean**	A boolean indicating whether this fund is deleted or not
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
initialDeposit	**BigDecimal**	The initial deposit amount
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
priority	**Integer**	The fund priority
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

FundTransactionSummary

Represents a Moab Accounting Manager transaction summary, which is a consolidated view of multiple transactions. The transactions are grouped by object and action, and a total count is given for the summary.

Field Name	Type	Description
id	**Long**	
count	**Long**	The number of transactions in this grouping of object and action
action	**String**	Action name for the transaction
amount	**BigDecimal**	Amount of the transaction. A positive or amount signifies a credit, while a negative or zero amount signifies a debit.
object	**String**	Object's name associated with the transaction

Related Topics

- Accounting Funds

Fields: Fund Statements

See the associated Accounting Funds resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/funds/reports/statement`	Permissions
Hooks filename	`accounting.funds.reports.statement.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

FundStatement

An fund statement is a report generated from Moab Accounting Manager fund, allocation, and transaction data. It contains fields detailing the specific time period covered, the starting and ending balances, the total of the transactions, and fund and transaction details.

Field Name	Type	Description
id	**Long**	
endBalance	**BigDecimal**	The balance of the funds at the endTime of the statement
endTime	**Date**	The ending time that the statement covers
funds	**Set<Fund>**	The funds that this statement covers. Only a sub-set of the full fund fields are available from this property. This includes id, name, priority, description, and creationTime.
generationTime	**Date**	The date that the statement report was generated
startBalance	**BigDecimal**	The balance of the funds at the startTime of the statement
startTime	**Date**	The starting time that the statement covers
totalCredits	**BigDecimal**	The total number of credits that occurred during the time period that the statement covers
totalDebits	**BigDecimal**	The total number of debits that occurred during the time period that the statement covers
transactions	**Set<FundTransaction>**	Details of each specific transaction which occurred during the time period that this statement covers.

Fund

A fund is a container for a time-bounded reference currency called credits for which the usage is restricted by constraints that define how the credits must be used. Much like with a bank, an fund is a repository for these resource or service credits which are added through deposits and debited through withdrawals and charges. Each fund has a set of constraints

designating which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, Feature, etc.). Fund constraints may also be negated with an exclamation point leading the constraint value.

When credits are deposited into an fund, they are associated with a time period within which they are valid. These time-bounded pools of credits are known as allocations. (An allocation is a pool of billable units associated with an fund for use during a particular time period.) By using multiple allocations that expire in regular intervals it is possible to implement a use-it-or-lose-it policy and establish an allocation cycle.

Funds may be nested. Hierarchically nested funds may be useful for the delegation of management roles and responsibilities. Deposit shares may be established that assist to automate a trickle-down effect for funds deposited at higher level funds. Additionally, an optional overflow feature allows charges against lower level funds to trickle up the hierarchy.

Funds may have an arbitrary name which is not necessarily unique for the fund. Funds may also have a priority which will influence the order of fund selection when charging.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	Total Adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	The allocations associated with this fund.
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
creationTime	**Date**	Date this fund was created
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
defaultDeposit	**String**	The default deposit amount
deleted	**Boolean**	A boolean indicating whether this fund is deleted or not

Field Name	Type	Description
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
initialDeposit	**BigDecimal**	The initial deposit amount
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
priority	**Integer**	The fund priority
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation

Field Name	Type	Description
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

FundTransaction

Represents a Moab Accounting Manager transaction.

Field Name	Type	Description
id	**Long**	
account	**String**	The account associated with the transaction. For a credit this will likely be zero
action	**String**	Action name for the transaction
amount	**BigDecimal**	Amount of the transaction. A positive or amount signifies a credit, while a negative or zero amount signifies a debit.
instance	**String**	Instance name
machine	**String**	The machine associated with the transaction. For a credit this will likely be zero. This field is not available in the Cloud context.
object	**String**	Object's name associated with the transaction
time	**Date**	The date at which the transaction occurred
user	**String**	The user associated with the transaction. For a credit this will likely be zero

API version 2

FundStatement

An fund statement is a report generated from Moab Accounting Manager fund, allocation, and transaction data. It contains fields detailing the specific time period covered, the starting and ending balances, the total of the transactions, and fund and transaction details.

Field Name	Type	Description
id	**Long**	
endBalance	**BigDecimal**	The balance of the funds at the endTime of the statement
endTime	**Date**	The ending time that the statement covers
funds	**Set<Fund>**	The funds that this statement covers. Only a sub-set of the full fund fields are available from this property. This includes id, name, priority, description, and creationTime.
generationTime	**Date**	The date that the statement report was generated
startBalance	**BigDecimal**	The balance of the funds at the startTime of the statement
startTime	**Date**	The starting time that the statement covers
totalCredits	**BigDecimal**	The total number of credits that occurred during the time period that the statement covers
totalDebits	**BigDecimal**	The total number of debits that occurred during the time period that the statement covers
transactions	**Set<FundTransaction>**	Details of each specific transaction which occurred during the time period that this statement covers.

Fund

A fund is a container for a time-bounded reference currency called credits for which the usage is restricted by constraints that define how the credits must be used. Much like with a bank, an fund is a repository for these resource or service credits which are added through deposits and debited through withdrawals and charges. Each fund has a set of constraints

designating which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, Feature, etc.). Fund constraints may also be negated with an exclamation point leading the constraint value.

When credits are deposited into an fund, they are associated with a time period within which they are valid. These time-bounded pools of credits are known as allocations. (An allocation is a pool of billable units associated with an fund for use during a particular time period.) By using multiple allocations that expire in regular intervals it is possible to implement a use-it-or-lose-it policy and establish an allocation cycle.

Funds may be nested. Hierarchically nested funds may be useful for the delegation of management roles and responsibilities. Deposit shares may be established that assist to automate a trickle-down effect for funds deposited at higher level funds. Additionally, an optional overflow feature allows charges against lower level funds to trickle up the hierarchy.

Funds may have an arbitrary name which is not necessarily unique for the fund. Funds may also have a priority which will influence the order of fund selection when charging.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	Total Adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	The allocations associated with this fund.
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
creationTime	**Date**	Date this fund was created
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
defaultDeposit	**String**	The default deposit amount
deleted	**Boolean**	A boolean indicating whether this fund is deleted or not

Field Name	Type	Description
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
initialDeposit	**BigDecimal**	The initial deposit amount
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
priority	**Integer**	The fund priority
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation

Field Name	Type	Description
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

FundTransaction

Represents a Moab Accounting Manager transaction.

Field Name	Type	Description
id	**Long**	
account	**String**	The account associated with the transaction. For a credit this will likely be zero
action	**String**	Action name for the transaction
amount	**BigDecimal**	Amount of the transaction. A positive or amount signifies a credit, while a negative or zero amount signifies a debit.
instance	**String**	Instance name
machine	**String**	The machine associated with the transaction. For a credit this will likely be zero. This field is not available in the Cloud context.
object	**String**	Object's name associated with the transaction
time	**Date**	The date at which the transaction occurred
user	**String**	The user associated with the transaction. For a credit this will likely be zero

Related Topics

- Accounting Funds

Fields: Funds

See the associated Accounting Funds resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/funds`	Permissions
Hooks filename	`accounting.funds.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Fund

A fund is a container for a time-bounded reference currency called credits for which the usage is restricted by constraints that define how the credits must be used. Much like with a bank, an fund is a repository for these resource or service credits which are added through deposits and debited through withdrawals and charges. Each fund has a set of constraints designating which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, Feature, etc.). Fund constraints may also be negated with an exclamation point leading the constraint value.

When credits are deposited into an fund, they are associated with a time period within which they are valid. These time-bounded pools of credits are known as allocations. (An allocation is a pool of billable units associated with an fund for use during a particular time period.) By using multiple allocations that expire in regular intervals it is possible to implement a use-it-or-lose-it policy and establish an allocation cycle.

Funds may be nested. Hierarchically nested funds may be useful for the delegation of management roles and responsibilities. Deposit shares may be established that assist to automate a trickle-down effect for funds deposited at higher level funds. Additionally, an optional overflow feature allows charges against lower level funds to trickle up the hierarchy.

Funds may have an arbitrary name which is not necessarily unique for the fund. Funds may also have a priority which will influence the order of fund selection when charging.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	Total Adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	The allocations associated with this fund.
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
creationTime	**Date**	Date this fund was created

Field Name	Type	Description
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
defaultDeposit	**String**	The default deposit amount
deleted	**Boolean**	A boolean indicating whether this fund is deleted or not
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
initialDeposit	**BigDecimal**	The initial deposit amount
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
priority	**Integer**	The fund priority
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft

protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

API version 2

Fund

A fund is a container for a time-bounded reference currency called credits for which the usage is restricted by constraints that define how the credits must be used. Much like with a bank, an fund is a repository for these resource or service credits which are added through deposits and debited through withdrawals and charges. Each fund has a set of constraints designating which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, Feature, etc.). Fund constraints may also be negated with an exclamation point leading the constraint value.

When credits are deposited into an fund, they are associated with a time period within which they are valid. These time-bounded pools of credits are known as allocations. (An allocation is a pool of billable units associated with an fund for use during a particular time period.) By using multiple allocations that expire in regular intervals it is possible to implement a use-it-or-lose-it policy and establish an allocation cycle.

Funds may be nested. Hierarchically nested funds may be useful for the delegation of management roles and responsibilities. Deposit shares may be established that assist to automate a trickle-down effect for funds deposited at higher level funds. Additionally, an optional overflow feature allows charges against lower level funds to trickle up the hierarchy.

Funds may have an arbitrary name which is not necessarily unique for the fund. Funds may also have a priority which will influence the order of fund selection when charging.

Field Name	Type	Description
id	**Long**	The unique fund identifier
allocated	**BigDecimal**	Total Adjusted allocations. This value is affected positively by deposits, activations and destination transfers and affected negatively by withdrawals, deactivations and source transfers that have occurred since the last reset.
allocations	**Set<Allocation>**	The allocations associated with this fund.
amount	**BigDecimal**	The sum of active allocation amounts within this fund. It does not take into fund current liens.
creationTime	**Date**	Date this fund was created

Field Name	Type	Description
creditLimit	**BigDecimal**	The sum of active credit limits within this fund
defaultDeposit	**String**	The default deposit amount
deleted	**Boolean**	A boolean indicating whether this fund is deleted or not
description	**String**	The fund description
fundConstraints	**Set<FundConstraint>**	Constraints on fund usage.
initialDeposit	**BigDecimal**	The initial deposit amount
modificationTime	**Date**	The date this fund was last modified
name	**String**	The name of this fund
priority	**Integer**	The fund priority
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Allocation

An allocation is a time-bounded pool of resource or service credits associated with an fund. An fund may have multiple allocations, each for use during a different time period.

An allocation has a start time and an end time that defines the time period during which the allocation may be used. By default an allocation is created with an unbounded time period (-infinity to infinity). An active flag is automatically updated to true if the fund is within its valid timeframe or false if it is not. An allocation may also have a credit limit representing the amount by which it can go negative. Thus, by having a positive balance in the Amount field, the fund is like a debit fund, implementing a pay-first use-later model. By establishing a credit limit instead of depositing an initial balance, the fund will be like a credit fund, implementing a use-first pay-later model. These strategies can be combined by depositing some amount of funds coupled with a credit limit, implementing a form of overdraft

protection where the funds will be used down to the negative of the credit limit.

Field Name	Type	Description
id	**String**	The unique identifier for this allocation
active	**Boolean**	Indicates whether this allocation is active or not
allocated	**BigDecimal**	Adjusted allocation. This value stores the effective allocated amount based on the initial deposit and subsequent allocation adjustments via deposits, withdrawals or transfers.
amount	**BigDecimal**	The amount of this allocation
creationTime	**Date**	The date this allocation was created
creditLimit	**BigDecimal**	Determines how far in the negative this allocation is permitted to be used (enforced in quotes and liens)
deleted	**Boolean**	A boolean indicating whether this allocation is deleted or not
description	**String**	The description of this allocation
endTime	**Date**	The date this allocation becomes inactive
fund	**String**	The fund Id associated with this allocation
modificationTime	**Date**	The date this allocation was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The date this allocation becomes active
transactionId	**Long**	The id of the last modifying transaction

FundConstraint

Constraints designate which entities (such as Users, Accounts, Machines, Classes, Organizations, etc.) may access the encapsulated credits in a fund or for which aspects of usage the funds are intended (QualityOfService, GeographicalArea, etc.).

Field Name	Type	Description
id	**String**	The unique identifier of this constraint.
fund	**String**	The fund ID that this constraint is associated with.
name	**String**	The name of the constraint.
value	**String**	The value of the constraint. The constraint may be negated by the used of an exclamation point leading the value.

Related Topics

- Accounting Funds

Fields: Liens

> See the associated Accounting Liens resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/liens`	Permissions
Hooks filename	`accounting.liens.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Lien

A lien is a reservation or hold placed against an allocation. Before usage of a resource or service begins, a lien is placed against one or more allocations within the requesting user's applicable funds. Subsequent usage requests will also post liens while the available balance (active allocations minus liens) allows. When the usage ends, the lien is removed and the actual charge is made to the allocation(s). This procedure ensures that usage will only be permitted so long as the requestors have sufficient funds.

Field Name	Type	Description
id	**Long**	The unique lien identifier
allocations	**Set<LienAllocation>**	The allocation amounts reserved with this lien.
creationTime	**Date**	The date this lien was created
deleted	**Boolean**	A boolean indicating whether this lien is deleted or not
description	**String**	The lien description
duration	**Long**	The expected duration of the reserved usage in seconds
endTime	**Date**	The time the lien becomes inactive
instance	**String**	The lien is against the specified instance (i.e. job id)
modificationTime	**Date**	The date this lien was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The time the lien becomes active
transactionId	**Long**	The id of the last modifying transaction
usageRecord	**Long**	The id of the usage record associated with the lien and containing the usage properties

LienAllocation

Amounts of the allocations that the lien has holds against

Field Name	Type	Description
id	**String**	The child allocation id
amount	**Long**	The amount reserved against the allocation by this lien
fund	**Long**	The fund that the allocation is in
lien	**String**	The parent lien id

API version 2

Lien

A lien is a reservation or hold placed against an allocation. Before usage of a resource or service begins, a lien is placed against one or more allocations within the requesting user's applicable funds. Subsequent usage requests will also post liens while the available balance (active allocations minus liens) allows. When the usage ends, the lien is removed and the actual charge is made to the allocation(s). This procedure ensures that usage will only be permitted so long as the requestors have sufficient funds.

Field Name	Type	Description
id	**Long**	The unique lien identifier
allocations	**Set<LienAllocation>**	The allocation amounts reserved with this lien.
creationTime	**Date**	The date this lien was created
deleted	**Boolean**	A boolean indicating whether this lien is deleted or not
description	**String**	The lien description
duration	**Long**	The expected duration of the reserved usage in seconds
endTime	**Date**	The time the lien becomes inactive
instance	**String**	The lien is against the specified instance (i.e. job id)
modificationTime	**Date**	The date this lien was last modified
requestId	**Long**	The id of the last modifying request
startTime	**Date**	The time the lien becomes active
transactionId	**Long**	The id of the last modifying transaction
usageRecord	**Long**	The id of the usage record associated with the lien and containing the usage properties

LienAllocation

Amounts of the allocations that the lien has holds against

Field Name	Type	Description
id	**String**	The child allocation id
amount	**Long**	The amount reserved against the allocation by this lien
fund	**Long**	The fund that the allocation is in
lien	**String**	The parent lien id

Related Topics

- Accounting Liens

Fields: Organizations

See the associated Accounting Organizations resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/organizations`	Permissions
Hooks filename	`accounting.organizations.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Organization

An organization is a virtual organization in which accounts are grouped. An account may only belong to a single organization while an organization may have multiple accounts. For example, an account may represent a project or cost-center while an organization may represent an institutional department or business division. The purpose of defining organizations is to support the ability to produce reporting for higher-order organizational entities beyond the individual account. Default organization properties include an id (name in MAM) and a description. An organization can be created, queried, modified, and deleted.

Field Name	Type	Description
id	**String**	The unique organization identifier
creationTime	**Date**	The date this organization was created
deleted	**Boolean**	A boolean indicating whether this organization is deleted or not
description	**String**	The organization description
modificationTime	**Date**	The date this organization was last modified
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

API version 2

Organization

An organization is a virtual organization in which accounts are grouped. An account may only belong to a single organization while an organization may have multiple accounts. For example, an account may represent a project or cost-center while an organization may represent an institutional department or business division. The purpose of defining organizations is to support the ability to produce reporting for higher-order organizational entities beyond the individual account. Default organization properties include an id (name in MAM) and a description. An organization can be created, queried, modified, and deleted.

Field Name	Type	Description
id	**String**	The unique organization identifier
creationTime	**Date**	The date this organization was created
deleted	**Boolean**	A boolean indicating whether this organization is deleted or not
description	**String**	The organization description
modificationTime	**Date**	The date this organization was last modified
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Related Topics

- Accounting Organizations

Fields: Quotes

See the associated Accounting Quotes resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/quotes`	Permissions
Hooks filename	`accounting.quotes.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Quote

Quotes can be used to determine how much it will cost to use a resource or service. Provided the cost-only option is not specified, this step will additionally verify that the submitter has sufficient funds and meets all the allocation policy requirements for the usage, and can be used at the submission of the usage request as an early filter to prevent the usage from getting blocked when it tries to obtain a lien to start later. If a guaranteed quote is requested, a quote id is returned and can be used in the subsequent charge to guarantee the rates that were used to form the original quote. A guaranteed quote has the side effect of creating a quote record and a permanent usage record. A quote id will be returned which can be used with the lien and charge to claim the quoted charge rates. A cost-only quote can be used to determine how much would be charged for usage without verifying sufficient funds or checking to see if the charge could succeed.

Field Name	Type	Description
id	**Long**	The unique quote identifier
amount	**BigDecimal**	The total amount of the quote
chargeRates	**Set<QuoteChargeRate>**	The applied charges that make up this quote.
creationTime	**Date**	The date this quote was created
deleted	**Boolean**	A boolean indicating whether this quote is deleted or not
description	**String**	The quote description
duration	**Long**	The expected duration of the quoted usage in seconds
endTime	**Date**	The time the quote becomes inactive
instance	**String**	The quote instance name. (i.e. job id)
modificationTime	**Date**	The date this quote was last modified
pinned	**Boolean**	Boolean indicating whether the quote is pinned or not
requestId	**Long**	The id of the last modifying request

Field Name	Type	Description
startTime	**Date**	The time the quote becomes active
transactionId	**Long**	The id of the last modifying transaction
usageRecord	**Long**	The usage record id associated with this quote

QuoteChargeRate

Saved charge rates to be used when the quote is referenced

Field Name	Type	Description
id	**Long**	
amount	**String**	The charge rate amount
name	**String**	The child charge rate name
quote	**String**	The parent quote id
value	**String**	The child charge rate value

API version 2

Quote

Quotes can be used to determine how much it will cost to use a resource or service. Provided the cost-only option is not specified, this step will additionally verify that the submitter has sufficient funds and meets all the allocation policy requirements for the usage, and can be used at the submission of the usage request as an early filter to prevent the usage from getting blocked when it tries to obtain a lien to start later. If a guaranteed quote is requested, a quote id is returned and can be used in the subsequent charge to guarantee the rates that were used to form the original quote. A guaranteed quote has the side effect of creating a quote record and a permanent usage record. A quote id will be returned which can be used with the lien and charge to claim the quoted charge rates. A cost-only quote can be used to determine how much would be charged for usage without verifying sufficient funds or checking to see if the charge could succeed.

Field Name	Type	Description
id	**Long**	The unique quote identifier
amount	**BigDecimal**	The total amount of the quote
chargeRates	**Set<QuoteChargeRate>**	The applied charges that make up this quote.
creationTime	**Date**	The date this quote was created
deleted	**Boolean**	A boolean indicating whether this quote is deleted or not
description	**String**	The quote description
duration	**Long**	The expected duration of the quoted usage in seconds
endTime	**Date**	The time the quote becomes inactive
instance	**String**	The quote instance name. (i.e. job id)
modificationTime	**Date**	The date this quote was last modified
pinned	**Boolean**	Boolean indicating whether the quote is pinned or not
requestId	**Long**	The id of the last modifying request

Field Name	Type	Description
startTime	**Date**	The time the quote becomes active
transactionId	**Long**	The id of the last modifying transaction
usageRecord	**Long**	The usage record id associated with this quote

QuoteChargeRate

Saved charge rates to be used when the quote is referenced

Field Name	Type	Description
id	**Long**	
amount	**String**	The charge rate amount
name	**String**	The child charge rate name
quote	**String**	The parent quote id
value	**String**	The child charge rate value

Related Topics

- Accounting Quotes

Fields: Transactions

See the associated Accounting Transactions resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/transactions`	Permissions
Hooks filename	`accounting.transactions.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Transaction

Moab Accounting Manager logs all modifying transactions in a detailed transaction journal (queries are not recorded). Previous transactions can be queried but not modified or deleted. By default, a standard user may only query transactions performed by them.

Field Name	Type	Description
id	**Long**	The unique transaction identifier
account	**String**	The account name associated with the transaction
action	**String**	The transaction action name
actor	**String**	The authenticated user that performed the action
allocation	**Long**	The allocation id associated with the transaction
amount	**BigDecimal**	The amount
child	**String**	If the transaction object is an association, this is the value of the child
creationTime	**Date**	The date this transaction was created
deleted	**Boolean**	A boolean indicating whether this transaction is deleted or not
delta	**BigDecimal**	The effective change (positive or negative) to the balance of an allocation
description	**String**	The description for the transaction
duration	**Long**	The duration associated with the transaction in seconds
fund	**Long**	The fund id associated with the transaction
instance	**String**	The instance name (e.g. the job id)
key	**String**	The object primary key value

Field Name	Type	Description
machine	**String**	The machine name associated with the transaction (e.g. the cluster name)
modificationTime	**Date**	The date this transaction was last modified
object	**String**	The transaction object name
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction
usageRecord	**Long**	The usage record id associated with the transaction
user	**String**	The user name associated with the transaction

API version 2

Transaction

Moab Accounting Manager logs all modifying transactions in a detailed transaction journal (queries are not recorded). Previous transactions can be queried but not modified or deleted. By default, a standard user may only query transactions performed by them.

Field Name	Type	Description
id	**Long**	The unique transaction identifier
account	**String**	The account name associated with the transaction
action	**String**	The transaction action name
actor	**String**	The authenticated user that performed the action
allocation	**Long**	The allocation id associated with the transaction
amount	**BigDecimal**	The amount
child	**String**	If the transaction object is an association, this is the value of the child
creationTime	**Date**	The date this transaction was created
deleted	**Boolean**	A boolean indicating whether this transaction is deleted or not
delta	**BigDecimal**	The effective change (positive or negative) to the balance of an allocation
description	**String**	The description for the transaction
duration	**Long**	The duration associated with the transaction in seconds
fund	**Long**	The fund id associated with the transaction
instance	**String**	The instance name (e.g. the job id)
key	**String**	The object primary key value

Field Name	Type	Description
machine	**String**	The machine name associated with the transaction (e.g. the cluster name)
modificationTime	**Date**	The date this transaction was last modified
object	**String**	The transaction object name
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction
usageRecord	**Long**	The usage record id associated with the transaction
user	**String**	The user name associated with the transaction

Related Topics

- Accounting Transactions

Fields: Usage Records

See the associated Accounting Usage Records resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/usage-records`	Permissions
Hooks filename	`accounting.usage-records.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

UsageRecord

A usage record tracks the usage of resources and services on your system, recording the charge and the details of the usage in a usage record.

Usage Record quotes can be used to determine how much it will cost to use a resource or service. Provided the cost-only option is not specified, this step will additionally verify that the submitter has sufficient funds and meets all the allocation policy requirements for the usage, and can be used at the submission of the usage request as an early filter to prevent the usage from getting blocked when it tries to obtain a lien to start later. If a guaranteed quote is requested, a quote id is returned and can be used in the subsequent charge to guarantee the rates that were used to form the original quote. A guaranteed quote has the side effect of creating a quote record and a permanent usage record. A quote id will be returned which can be used with the lien and charge to claim the quoted charge rates. A cost-only quote can be used to determine how much would be charged for usage without verifying sufficient funds or checking to see if the charge could succeed.

A usage lien can be used to place a hold on the user's fund before usage starts to ensure that the credits will be there when it completes. The replace option may be specified if you want the new lien to replace existing liens of the same instance name (associated with the same usage record). The modify option may be specified to dynamically extend any existing lien with the same instance name with the specified characteristics instead of creating a new one.

A usage charge debits the appropriate allocations based on the attributes of the usage. The charge is calculated based on factors including the resources and services used, the usage time, and other quality-based factors. By default, any liens associated with the charge will be removed. The incremental option may be specified if you want associated liens to be reduced instead of removed. If a usage record already exists for the instance being charged it will be updated with the data properties passed in with the charge request, otherwise a new usage record will be created.

Field Name	Type	POST	Description
id	**Long**	No	The unique usage record identifier
charge	**String**	No	The cumulative amount charged
creationTime	**Date**	No	The date this usage record was created

Field Name	Type	POST	Description
deleted	**Boolean**	No	A boolean indicating whether this usage record is deleted or not
instance	**String**	No	The usage record instance name (i.e. job id)
modificationTime	**Date**	No	The date this usage record was last modified
qualityOfService	**String**	No	The quality of service associated with the usage
quote	**Long**	No	The associated quote id
requestId	**Long**	No	The id of the last modifying request
stage	**String**	No	The last affecting action (i.e. Create, Quote, Reserve, Query)
transactionId	**Long**	No	The id of the last modifying transaction
type	**String**	No	The usage record type
user	**String**	No	The user name associated with the usage

API version 2

UsageRecord

A usage record tracks the usage of resources and services on your system, recording the charge and the details of the usage in a usage record.

Usage Record quotes can be used to determine how much it will cost to use a resource or service. Provided the cost-only option is not specified, this step will additionally verify that the submitter has sufficient funds and meets all the allocation policy requirements for the usage, and can be used at the submission of the usage request as an early filter to prevent the usage from getting blocked when it tries to obtain a lien to start later. If a guaranteed quote is requested, a quote id is returned and can be used in the subsequent charge to guarantee the rates that were used to form the original quote. A guaranteed quote has the side effect of creating a quote record and a permanent usage record. A quote id will be returned which can be used with the lien and charge to claim the quoted charge rates. A cost-only quote can be used to determine how much would be charged for usage without verifying sufficient funds or checking to see if the charge could succeed.

A usage lien can be used to place a hold on the user's fund before usage starts to ensure that the credits will be there when it completes. The replace option may be specified if you want the new lien to replace existing liens of the same instance name (associated with the same usage record). The modify option may be specified to dynamically extend any existing lien with the same instance name with the specified characteristics instead of creating a new one.

A usage charge debits the appropriate allocations based on the attributes of the usage. The charge is calculated based on factors including the resources and services used, the usage time, and other quality-based factors. By default, any liens associated with the charge will be removed. The incremental option may be specified if you want associated liens to be reduced instead of removed. If a usage record already exists for the instance being charged it will be updated with the data properties passed in with the charge request, otherwise a new usage record will be created.

Field Name	Type	POST	Description
id	**Long**	No	The unique usage record identifier
charge	**String**	No	The cumulative amount charged
creationTime	**Date**	No	The date this usage record was created

Field Name	Type	POST	Description
deleted	**Boolean**	No	A boolean indicating whether this usage record is deleted or not
instance	**String**	No	The usage record instance name (i.e. job id)
modificationTime	**Date**	No	The date this usage record was last modified
qualityOfService	**String**	No	The quality of service associated with the usage
quote	**Long**	No	The associated quote id
requestId	**Long**	No	The id of the last modifying request
stage	**String**	No	The last affecting action (i.e. Create, Quote, Reserve, Query)
transactionId	**Long**	No	The id of the last modifying transaction
type	**String**	No	The usage record type
user	**String**	No	The user name associated with the usage

Related Topics

- Accounting Usage Records

Fields: Users

See the associated Accounting Users resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`accounting/users`	Permissions
Hooks filename	`accounting.users.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

User

A user is a person authorized to use a resource or service. Default user properties include the common name, phone number, email address, default account, and description for that person.

Field Name	Type	Description
id	**String**	The unique user identifier
active	**Boolean**	A boolean indicating whether this user is active or not
creationTime	**Date**	The date this user was created
defaultAccount	**String**	The default account for this user
deleted	**Boolean**	A boolean indicating whether this user is deleted or not
description	**String**	The user description
emailAddress	**String**	The user's email address
modificationTime	**Date**	The date this user was last modified
phoneNumber	**String**	The user's phone number
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

API version 2

User

A user is a person authorized to use a resource or service. Default user properties include the common name, phone number, email address, default account, and description for that person.

Field Name	Type	Description
id	**String**	The unique user identifier
active	**Boolean**	A boolean indicating whether this user is active or not
creationTime	**Date**	The date this user was created
defaultAccount	**String**	The default account for this user
deleted	**Boolean**	A boolean indicating whether this user is deleted or not
description	**String**	The user description
emailAddress	**String**	The user's email address
modificationTime	**Date**	The date this user was last modified
phoneNumber	**String**	The user's phone number
requestId	**Long**	The id of the last modifying request
transactionId	**Long**	The id of the last modifying transaction

Related Topics

- Accounting Users

Fields: Credentials

See the associated Credentials resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`credentials`	Permissions
Hooks filename	`credentials.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Credential

A credential is an entity, such as a user or a group, that has access to resources. Credentials allow specification of job ownership, tracking of resource usage, enforcement of policies, and many other features.

Field Name	Type	PUT	Description
name	**String**	No	The name of the credential.

API version 2

Credential

A credential is an entity, such as a user or a group, that has access to resources. Credentials allow specification of job ownership, tracking of resource usage, enforcement of policies, and many other features.

Field Name	Type	PUT	Description
name	**String**	No	The name of the credential.

Related Topics

- Credentials

Fields: Events

See the associated Events resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`events`	Permissions
Hooks filename	`events.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Event

Represents an event originating from any component in the system (MWM, MWS, MAM, etc). Events are related to, but not the same as, Notifications. See NotificationCondition for an explanation of when to use an event vs a notification.

Field Name	Type	POST	Description
id	**String**	No	The unique ID for this event
arguments	**List<String>**	Yes	The event's arguments
associatedObjects	**Set<AssociatedObject>**	Yes	Objects relating to the event
code	**int**	Yes	This is a positive, 32-bit numeric value. Source code that needs to take action on events based on which event (error) occurred can switch based on this value. The top 16 bits are determined by the severity of the event and the component that emits it. The bottom 16 bits are assigned by any arbitrary mechanism convenient to a component. Each component thus has 64k unique event codes that it can assign. Once assigned, event codes are immutable; it can never be the case that error 12345 means one thing in release A, and a different thing in release B.
eventDate	**Date**	Yes	The date and time the event occurred, not the date and time MWS received the event. It is up to the reporting component to report this time accurately. Required during POST.
eventType	**String**	Yes	Signifies what type of event. Cannot contain single quotes(') or double quotes(").
message	**String**	Yes	A summary of what happened that caused this event
origin	**String**	Yes	The origin of this event. Cannot contain single quotes(') or double quotes(").

Field Name	Type	POST	Description
severity	**EventSeverity**	Yes	Signifies the severity of an event.
tenant	**Map<String, String>**	No	The event's tenant (contains tenant id and name)

AssociatedObject

Represents and uniquely identifies an object associated with an event. (e.g node, job, reservation, trigger)

Field Name	Type	POST	Description
id	**String**	Yes	The object id (e.g. reservation.1, job.21, vm3). Cannot contain single quotes(') or double quotes(").
type	**String**	Yes	The type of object (e.g. node, job, reservation). Cannot contain single quotes(') or double quotes(").

EventSeverity

Value	Description
INFO	
WARN	
ERROR	
FATAL	

API version 2

EventVersion2

Field Name	Type	POST	Description
id	**String**	No	The unique ID for this event
details	**Map<String, Map>**	Yes	A map where detail name maps to detail value. (e.g. "sourceHypervisor" => "blade256", "destinationHypervisor" => "blade257", "os" => "centos-6.5-stateless)
errorMessage	**ErrorMessageVersion2**	Yes	Details about any errors associated with the event. If this event was not associated with any errors this field will be null
eventCategory	**String**	Yes	Signifies what category of event.
eventTime	**Date**	Yes	The time the event occurred, not the time MWS received the event. It is up to the reporting component to report this time accurately. Corresponds to eventDate in API Version 3. Required during POST.
eventType	**String**	Yes	Signifies what type of event.
facility	**String**	Yes	A categorization of how this event fits in with other events.
initiatedBy	**UserDetailsVersion2**	Yes	Details about the user that initiated this event
primaryObject	**MoabObjectVersion2**	Yes	Most events will have a "primary object" associated with it. An event can have at most ONE primary object. For example, a JobStart event will have a primary job object, so the type would be "job" and the object ID would be the ID of the job. Primary objects are, however, optional, depending on the type of event. For example, a "SchedulerCommand" event does not have a primary object.

Field Name	Type	POST	Description
relatedObjects	Set<MoabObjectVersion2>	Yes	Objects relating to the event that are not the primary object. Corresponds to associatedObjects in API Version 3.
severity	**String**	Yes	Signifies the severity of an event. Severity can be "FATAL", "ERROR", "WARN", "INFO"
sourceComponent	**String**	Yes	What Adaptive Computing component reported this event. Examples: "MWM", "MWS", "MAM", etc. Corresponds to origin in API Version 3.
status	**String**	Yes	The status of the reported event.

ErrorMessageVersion2

Field Name	Type	POST	Description
errorCode	**String**	Yes	The original error code generated or detected by the originator.
message	**String**	Yes	If an event has a status of "failure" or other non-successful operation, this field should provide a human-friendly error message Corresponds to Event.message in API Version 3 and above.
originator	**String**	Yes	The software component or entity that generated or detected the error (e.g. Moab, Torque, MWS, Viewpoint, RM, Database, etc).

UserDetailsVersion2

Field Name	Type	POST	Description
proxyUser	**String**	Yes	The proxy user that initiated the event.
user	**String**	Yes	The user that initiated the event.

MoabObjectVersion2

Field Name	Type	POST	Description
id	**String**	Yes	The moab object id (e.g. reservation.1, job.21, vm3)
serialization	**String**	Yes	A serialized representation of the object
type	**String**	Yes	The moab object type (e.g. node, job, reservation)

Related Topics

- Events

Fields: Images

See the associated Images resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`images`	Permissions
Hooks filename	`images.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Image

An image is used to track the different types of operating systems and hypervisors available in a data center. If the image is a hypervisor, it can contain other images which are the available virtual machines of the hypervisor.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this image.
active	**Boolean**	Yes	Yes	If false, the image is flagged as inactive and should not be used. Defaults to true.

Field Name	Type	POST	PUT	Description
extensions	**Map<String, Map>**	Yes	Yes	A map containing maps which represent settings for provisioning managers. Only one extension may be present on an image at a time currently. Valid default provisioning manager specific extensions include 'xcat'. Required properties for 'xcat' when hypervisor is false: • os - The name of the operating system according to xCAT • architecture - The architecture, such as x86_64 • profile - The xCAT profile to use for the image Required properties for 'xcat' when hypervisor is true: • os - The name of the operating system according to xCAT • architecture - The architecture, such as x86_64 • profile - The xCAT profile to use for the image • hvGroupName - The name of the xCAT hypervisor group • vmGroupName - The name of the xCAT VM group
features	**Set<String>**	Yes	Yes	The set of features used by the provisioning manager.

Field Name	Type	POST	PUT	Description
hypervisor	**Boolean**	Yes	Yes	Whether or not the image is a hypervisor. Required during POST. Note that this is related to, but not the same as, supportsPhysicalMachine. Also, when this is false, no virtualizedImages may be specified for an image.
hypervisorType	**String**	Yes	Yes	The type of the hypervisor, which is indicative of the hypervisor technology used in this image. Required if this image is a hypervisor image.
name	**String**	Yes	Yes	The unique human-readable name of this image. Required during POST.
osType	**String**	Yes	Yes	The type of the operating system such as 'Linux' or 'Windows'. Required during POST.
supportsPhysicalMachine	**Boolean**	Yes	Yes	Specifies whether the image can be used to provision a physical machine, defaults to false. Either this or supportsVirtualMachine must be set to true. Note that this is related to, but not the same as, hypervisor. Some images may not be hypervisors but can be provisioned on a physical machine.
supportsVirtualMachine	**Boolean**	Yes	Yes	Specifies whether the image can be used to provision a virtual machine, defaults to false. Either this or supportsPhysicalMachine must be set to true.
templateName	**String**	Yes	Yes	The VM template to use for this image. Only valid if the type is set to a valid template type such as 'ImageType.LINKED_CLONE'.

Field Name	Type	POST	PUT	Description
type	**ImageType**	Yes	Yes	The type of the image. This property may affect the valid values to use for other fields. See ImageType for more information. (See also: templateName.)
virtualizedImages	**Set<Image>**	Yes	Yes	The set of images available on this hypervisor.

ImageType

Represents an image type, such as stateful or stateless. This is used by provisioning managers and applications to correctly provision and represent the image.

Certain types are only valid for images configured as templates using the Image.templateName field. This currently includes ImageType.LINKED_CLONE and ImageType.FULL_CLONE.

Value	Description
STATEFUL	
STATELESS	
STATELITE	
LINKED_CLONE	Template type. When this image type is used, the Image.hypervisor field must be set to false, Image.supportsVirtualMachine must be true, and Image.supportsPhysicalMachine must be false.
FULL_CLONE	Template type. When this image type is used, the Image.hypervisor field must be set to false, Image.supportsVirtualMachine must be true, and Image.supportsPhysicalMachine must be false.

API version 2

Image

An image is used to track the different types of operating systems and hypervisors available in a data center. If the image is a hypervisor, it can contain other images which are the available virtual machines of the hypervisor.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this image.
active	**Boolean**	Yes	Yes	If false, the image is flagged as inactive and should not be used. Defaults to true.

Field Name	Type	POST	PUT	Description
extensions	**Map<String, Map>**	Yes	Yes	A map containing maps which represent settings for provisioning managers. Only one extension may be present on an image at a time currently. Valid default provisioning manager specific extensions include 'xcat'. Required properties for 'xcat' when hypervisor is false: • os - The name of the operating system according to xCAT • architecture - The architecture, such as x86_64 • profile - The xCAT profile to use for the image Required properties for 'xcat' when hypervisor is true: • os - The name of the operating system according to xCAT • architecture - The architecture, such as x86_64 • profile - The xCAT profile to use for the image • hvGroupName - The name of the xCAT hypervisor group • vmGroupName - The name of the xCAT VM group
features	**Set<String>**	Yes	Yes	The set of features used by the provisioning manager.

Field Name	Type	POST	PUT	Description
hypervisor	**Boolean**	Yes	Yes	Whether or not the image is a hypervisor. Required during POST. Note that this is related to, but not the same as, supportsPhysicalMachine. Also, when this is false, no virtualizedImages may be specified for an image.
hypervisorType	**String**	Yes	Yes	The type of the hypervisor, which is indicative of the hypervisor technology used in this image. Required if this image is a hypervisor image.
name	**String**	Yes	Yes	The unique human-readable name of this image. Required during POST.
osType	**String**	Yes	Yes	The type of the operating system such as 'Linux' or 'Windows'. Required during POST.
supportsPhysicalMachine	**Boolean**	Yes	Yes	Specifies whether the image can be used to provision a physical machine, defaults to false. Either this or supportsVirtualMachine must be set to true. Note that this is related to, but not the same as, hypervisor. Some images may not be hypervisors but can be provisioned on a physical machine.
supportsVirtualMachine	**Boolean**	Yes	Yes	Specifies whether the image can be used to provision a virtual machine, defaults to false. Either this or supportsPhysicalMachine must be set to true.
templateName	**String**	Yes	Yes	The VM template to use for this image. Only valid if the type is set to a valid template type such as 'ImageType.LINKED_CLONE'.

Field Name	Type	POST	PUT	Description
type	**ImageType**	Yes	Yes	The type of the image. This property may affect the valid values to use for other fields. See ImageType for more information. (See also: templateName.)
virtualizedImages	**Set<Image>**	Yes	Yes	The set of images available on this hypervisor.

ImageType

Represents an image type, such as stateful or stateless. This is used by provisioning managers and applications to correctly provision and represent the image.

Certain types are only valid for images configured as templates using the Image.templateName field. This currently includes ImageType.LINKED_CLONE and ImageType.FULL_CLONE.

Value	Description
STATEFUL	
STATELESS	
STATELITE	
LINKED_CLONE	Template type. When this image type is used, the Image.hypervisor field must be set to false, Image.supportsVirtualMachine must be true, and Image.supportsPhysicalMachine must be false.
FULL_CLONE	Template type. When this image type is used, the Image.hypervisor field must be set to false, Image.supportsVirtualMachine must be true, and Image.supportsPhysicalMachine must be false.

Related Topics

- Images

Fields: Job Arrays

See the associated Job Arrays resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`job-arrays`	Permissions
Hooks filename	`job-arrays.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

JobArray

Job arrays are an easy way to submit many sub-jobs that perform the same work using the same script, but operate on different sets of data. Sub-jobs are the jobs created by an array job and are identified by the array job ID and an index; for example, if 235[1] is an identifier, the number 235 is a job array ID, and 1 is the sub-job.

Field Name	Type	POST	Description
cancellationPolicy	**CancellationPolicyInformation**	Yes	Represents the cancellation policy to use for the job array.
indexRanges	**List<JobArrayIndexRange>**	Yes	The index ranges used to generate the sub-job indices. To use hard-coded values, see indexValues.
indexValues	**List<Long>**	Yes	The index values to use for the sub-jobs. To use ranges, see indexRanges.
jobPrototype	**Job**	Yes	The definition of the job to use for each sub-job.
name	**String**	Yes	The name of the job array. In MWS API version 1, this is stored in the `name` field of the created jobs. In MWS API version 2, this is stored in the `customName` field of the created jobs.
slotLimit	**Long**	Yes	(Optional) The number of sub-jobs in the array that can run at a time.

CancellationPolicyInformation

Job arrays can be canceled based on the success or failure of the first or any sub-job. This class represents the failure policies.

Field Name	Type	POST	Description
anyJob	**CancellationPolicy**	Yes	The cancellation policy based on the result of any sub-job. May be used in combination with firstJob.

Field Name	Type	POST	Description
firstJob	CancellationPolicy	Yes	The cancellation policy based on the result of the first sub-job (array index 1). May be used in combination with anyJob.

CancellationPolicy

This enumeration represents job array cancellation policies, and is to be used in combination with CancellationPolicyInformation.

Value	Description
SUCCESS	Cancels the job array if the specified sub-job succeeds.
FAILURE	Cancels the job array if the specified sub-job fails.

JobArrayIndexRange

Represents information about a job index expression. This is used when creating job arrays only.

Field Name	Type	POST	Description
endIndex	**Long**	Yes	The end of the index range. i.e. `10` for `1-10`.
increment	**Long**	Yes	The increment of the index range, defaults to 1 and must be greater than 0. For a range of 1-10 with an increment of 2, the list of indices will be [1, 3, 5, 7, 9].
startIndex	**Long**	Yes	The start of the index range. i.e. `1` for `1-10`.

Job

This class represents a job in the Moab Workload Manager. A job is a request for compute resources (CPUs, memory, storage) with which the requester can do work for a given amount of time. In an HPC environment, this might be a batch script to perform a Monte Carlo simulation. In a cloud environment, this would be a virtual machine and its associated storage. Moab will evaluate the request and assign the requested resources to the requester based on policies, current demand, and other factors in the data center. A job will also usually have some process that Moab starts automatically at the assigned start time. In an HPC environment, this can be

starting a batch script on the assigned nodes. In a cloud environment, this can be starting provisioning processes to create the virtual machine and storage and install software on it.

Field Name	Type	POST	Description
id	**String**	No	The unique identifier of this job. Note: this field is not user-assigned and is generated by the database.
arrayIndex	**Long**	No	If this job is a sub-job of a JobArray, this field contains the index of this job in the array. For example, if this job is `Moab.1[2]`, the array index would be `2`.
arrayMasterName	**String**	No	If this job is a sub-job of a JobArray, this field contains the name of the job array master. For example, if this job is `Moab.1[2]`, the array master name would be `Moab.1`.
attributes	**Set<String>**	Yes	The list of generic attributes associated with this job.
blocks	**Set<JobBlock>**	No	Reasons the job is blocked from running.
bypassCount	**Integer**	No	The number of times the job has been backfilled.
cancelCount	**Integer**	No	The number of times a job has received a cancel request.

Field Name	Type	POST	Description
commandFile	**String**	Yes	The name of the job script file (absolute path). If commandFile is set and commandScript is not set, then MWS must have read access to the file. If commandFile and commandScript are both set, then MWS does not read the contents of the file, but it does provide the name of the file to Moab.
commandLineArguments	**String**	Yes	The command line arguments passed to the job script specified by commandFile or commandScript.
commandScript	**String**	Yes	The contents of the job script. This field must be Base64-encoded.
completionCode	**Integer**	No	The exit code from this job.
cpuTime	**Long**	No	CPU usage time in seconds as reported by the resource manager.
credentials	JobCredentials	Yes	The credentials (user and group, for example) associated with this job.
customName	**String**	Yes	The user-specified name of this job.
dates	JobDates	Yes	Various dates associated with this job.
deferCount	**Integer**	No	The number of times a job has been deferred.

Field Name	Type	POST	Description
dependencies	Set<JobDependency>	Yes	Dependencies that must be fulfilled before the job can start.
description	**String**	No	The description of the job. Can be set only in a job template.
duration	**Long**	Yes	The length of time in seconds requested for the job. Note that it is possible to set duration to "INFINITY" if the AllowInfiniteJobs flag is set on the scheduler in the moab.cfg.
durationActive	**Long**	No	The length of time in seconds the job has been active or running.
durationQueued	**Long**	No	The length of time in seconds the job has been eligible to run in the queue.
durationRemaining	**Long**	No	An estimate of the time remaining, in seconds, before the job will complete.
durationSuspended	**Long**	No	The length of time in seconds the job has been suspended.
emailNotifyAddresses	**Set<String>**	Yes	The list of addresses to whom email is sent by the execution server.
emailNotifyTypes	Set<JobEmailNotifyType>	Yes	The list of email notify types attached to the job.
environmentRequested	**Boolean**	Yes	Setting this field to true tells the Moab Workload Manager to set various variables, if populated, in the job's environment.

Field Name	Type	POST	Description
environmentVariables	**Map<String, Map>**	Yes	The list of environment variables for this job.
epilogScript	**String**	Yes	The path to the TORQUE epilog script.
flags	**Set<JobFlag>**	Yes	The flags that are set on this job.
holdDate	**Date**	No	The date the most recent hold was placed on the job.
holdReason	**JobHoldReason**	No	The reason the job is on hold.
holds	**Set<JobHoldType>**	Yes	The holds that are set on the job. The "User" hold type is valid during POST.
initialWorkingDirectory	**String**	Yes	The path to the directory in which the job will be started.
isActive	**Boolean**	No	True if the job is active, false if the job is complete.
jobGroup	**String**	Yes	The job group to which this job belongs (different from credentials.group).
masterNode	**DomainProxy**	No	The first node in the list of allocated nodes for this job. For TORQUE jobs, this represents the "mother superior."
memorySecondsDedicated	**Double**	No	The memory seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.

Field Name	Type	POST	Description
memorySecondsUtilized	**Double**	No	The memory seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
messages	**Set<Message>**	No	The list of messages associated with the job. The "message" field is valid during PUT.
migrateCount	**Integer**	No	The number of times the job has been migrated.
minimumPreemptTime	**Long**	No	The minimum length of time, in seconds, an active job must be running before it is eligible for preemption.
mwmName	**String**	No	The name of the Moab Workload Manager instance that owns this job.
name	**String**	No	The name of this job. This name is unique *per instance* of Moab Workload Manager (i.e. not globally).
nodesExcluded	**Set<DomainProxy>**	Yes	The list of nodes that should not be considered for this job.
nodesRequested	**Set<DomainProxy>**	Yes	The exact set, superset, or subset of nodes on which this job must run. (See also: nodesRequestedPolicy.)
nodesRequestedPolicy	**JobHostListMode**	Yes	Indicates an exact set, superset, or subset of nodes on which the job must run. Only relevant if nodesRequested is provided. (See also: nodesRequested.)

Field Name	Type	POST	Description
partitionAccessList	**Set<String>**	No	The list of partitions that this job can access.
partitionAccessListRequested	**Set<String>**	Yes	The list of partitions that this job has requested.
preemptCount	**Integer**	No	The number of times the job has been preempted.
priorities	**JobPriority**	Yes	The list of priorities for the job.
processorSecondsDedicated	**Double**	No	The processor seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.
processorSecondsLimit	**Double**	No	The limit for processorSecondsUtilized.
processorSecondsUtilized	**Double**	No	The processor seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
prologScript	**String**	Yes	The path to the TORQUE prolog script.
queueStatus	**JobQueueStatus**	No	The status of the job in its queue.
rejectPolicies	**Set<JobRejectPolicy>**	No	The list of policies enabled when a job is rejected.
requirements	**Set<JobRequirement>**	Yes	The list of items required for this job to run. Only JobRequirement.features is valid during PUT.

Field Name	Type	POST	Description
reservationRequested	DomainProxy	Yes	The reservation that the job requested.
resourceFailPolicy	JobResourceFailPolicyType	Yes	The policy that dictates what should happen to the job if it is running and at least one of the resources it is using fails.
resourceManagerExtension	**String**	Yes	If provided during POST, this string will be added to the resource manager extension section of the job submission. Example: "bandwidth=120;queuejob=false" Note that the delimiter between resourceManagerExtension elements is the semicolon.
resourceManagers	Set<ResourceManager>	No	The list of resource managers associated with this job.
rmStandardErrorFilePath	**String**	No	The path to the remote file containing the standard error of the job.
rmStandardOutputFilePath	**String**	No	The path to the remote file containing the standard output of the job.
shellName	**String**	Yes	Declares the shell that interprets the job script.
standardErrorFilePath	**String**	Yes	The path to the file containing the standard error of the job.
standardOutputFilePath	**String**	Yes	The path to the file containing the standard output of the job.

Field Name	Type	POST	Description
startCount	**Integer**	No	The number of times the job has been started.
states	JobStateInformation	No	Information about the state of the job.
submitHost	**String**	Yes	The host from which the job was submitted.
systemJobAction	**String**	No	The action the system job will take.
systemJobType	JobSystemJobType	No	The type of system job. In the Moab Cloud Suite, this will usually be "vmtracking" or "generic."
targetedJobAction	JobActionType	No	The action that this job is performing on another job.
targetedJobName	**String**	No	The name of the job on which this job is performing the targetedJobAction.
templates	Set<DomainProxy>	Yes	The list of all job templates to be set on this job.
triggers	**Set<String>**	No	The list of triggers associated with this job.
variables	**Map<String, Map>**	Yes	The list of variables that this job owns or sets on completion.
virtualContainers	Set<DomainProxy>	Yes	When submitting this job, add it to the specified existing virtual container. Valid during POST, but only one virtual container can be specified.

Field Name	Type	POST	Description
virtualMachines	Set<DomainProxy>	No	The list of virtual machines that are allocated to this job.
vmUsagePolicy	VMUsagePolicy	Yes	The requested Virtual Machine Usage Policy for this job.

JobBlock

Field Name	Type	POST	Description
category	JobBlockCategory	No	
message	**String**	No	
type	JobBlockType	No	

JobBlockCategory

Value	Description
depend	
jobBlock	
migrate	

JobBlockType

Value	Description
ActivePolicy	
BadUser	
Dependency	

Value	Description
EState	
FairShare	
Hold	
IdlePolicy	
LocalPolicy	
NoClass	
NoData	
NoResource	
NoTime	
PartitionAccess	
Priority	
RMSubmissionFailure	
StartDate	
State	
SysLimits	

JobCredentials

Moab Workload Manager supports the concept of credentials, which provide a means of attributing policy and resource access to entities such as users and groups. These credentials allow specification of job ownership, tracking of resource usage, enforcement of policies, and many other features.

Field Name	Type	POST	Description
account	**String**	Yes	The account credential is also referred to as the project. This credential is generally associated with a group of users along the lines of a particular project for accounting and billing purposes.
group	**String**	Yes	The group credential represents an aggregation of users. User-to-group mappings are often specified by the operating system or resource manager and typically map to a user's UNIX group ID. However, user-to-group mappings may also be provided by a security and identity management service, or you can specify such directly within Moab.
jobClass	**String**	Yes	The concept of the class credential is derived from the resource manager class or queue object. Classes differ from other credentials in that they more directly impact job attributes. In standard HPC usage, a user submits a job to a class and this class imposes a number of factors on the job. The attributes of a class may be specified within the resource manager or directly within Moab.
qos	**String**	No	The quality of service assigned to this job. The concept of a quality of service (QoS) credential is unique to Moab and is not derived from any underlying concept or peer service. In most cases, the QoS credential is used to allow a site to set up a selection of service levels for end-users to choose from on a long-term or job-by-job basis. QoS's differ from other credentials in that they are centered around special access where this access may allow use of additional services, additional resources, or improved responsiveness. Unique to this credential, organizations may also choose to apply different charge rates to the varying levels of service available within each QoS. As QoS is an internal credential, all QoS configuration occurs within Moab.
qosRequested	**String**	Yes	The quality of service requested for this job.
user	**String**	Yes	The user credential is the fundamental credential within a workload manager; each job requires an association with exactly one user. In fact, the user credential is the only required credential in Moab; all others are optional. In most cases, the job's user credential is configured within or managed by the operating system itself, although Moab may be configured to obtain this information from an independent security and identity management service.

JobDates

Field Name	Type	POST	Description
completedDate	**Date**	No	
createdDate	**Date**	No	
deadlineDate	**Date**	Yes	The deadline for completion of the job.
dispatchedDate	**Date**	No	
earliestRequestedStartDate	**Date**	Yes	The job will start no sooner than this date.
earliestStartDate	**Date**	No	
eligibleDate	**Date**	No	
lastCanceledDate	**Date**	No	
lastChargedDate	**Date**	No	
lastPreemptedDate	**Date**	No	
lastUpdatedDate	**Date**	No	
startDate	**Date**	No	
submitDate	**Date**	No	
terminationDate	**Date**	No	

JobDependency

Field Name	Type	POST	Description
name	**String**	Yes	The name of the object on on which the job is dependent.
type	**JobDependencyType**	Yes	The type of job dependency. Only set is valid for POST.

Field Name	Type	POST	Description
value	**String**	No	

JobDependencyType

Represents the type of a job dependency. For now, only the "set" type is supported.

Value	Description
set	

JobEmailNotifyType

Value	Description
JobStart	
JobEnd	
JobFail	
All	

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.
ADVRES	The job requires the use of a reservation.

Value	Description
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.
INTERACTIVE	The job needs to interactive input from the user to run.

Value	Description
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.

Value	Description
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

JobHoldReason

Value	Description
Admin	
NoResources	
SystemLimitsExceeded	

Value	Description
BankFailure	
CannotDebitAccount	
InvalidAccount	
RMFailure	
RMReject	
PolicyViolation	
CredAccess	
CredHold	
PreReq	
Data	
Security	
MissingDependency	

JobHoldType

Value	Description
User	
System	
Batch	
Defer	
All	

DomainProxy

A reference to an object contained within an object. For example, a Virtual Machine object contains a reference to the Node on which it is running. That reference is represented by this class.

Field Name	Type	POST	Description
name	**String**	Yes	The name of the object.

Message

Field Name	Type	POST	Description
count	**Integer**	No	The number of times this message has occurred.
createdDate	**Date**	No	The date this message was created.
expireDate	**Date**	No	The date this message expires.
message	**String**	No	The message itself.

JobHostListMode

Value	Description
superset	
subset	
exactset	

JobPriority

Field Name	Type	POST	Description
run	**Long**	No	

Field Name	Type	POST	Description
start	**Long**	No	
system	**Long**	No	
user	**Long**	Yes	The user-requested priority for the job. By default, the range is between -1024 and 0. To enable priority range from -1024 to +1023, set `ENABLEPOSUSERPRIORITY` in the `moab.cfg` file.

JobQueueStatus

Value	Description
active	
blocked	
completed	
eligible	

JobRejectPolicy

Value	Description
CANCEL	
HOLD	
IGNORE	
MAIL	
RETRY	

JobRequirement

Field Name	Type	POST	Description
architecture	**String**	Yes	The architecture required by the job.
attributes	**Map<String, JobRequirementAttribute>**	Yes	Required node attributes with version number support.
features	**Set<String>**	No	The list of node features the job is scheduled against.
featuresExcluded	**Set<String>**	Yes	Excluded node features. That is, do not select nodes with these features. (See also: featuresExcludedMode.)
featuresExcludedMode	**JobRequirementFeaturesMode**	Yes	Indicates whether excluded features should be AND'ed or OR'd. The default is AND. Only relevant if featuresExcluded is provided. (See also: featuresExcluded.)
featuresRequested	**Set<String>**	Yes	Requested node features. (See also: featuresRequestedMode.)
featuresRequestedMode	**JobRequirementFeaturesMode**	Yes	Indicates whether requested features should be AND'ed or OR'd. The default is AND. Only relevant if featuresRequested is provided. (See also: featuresRequested.)
image	**String**	Yes	The image required by the job.
index	**Integer**	No	The index of the requirement, starting with 0.
metrics	**Map<String, Double>**	No	Generic metrics associated with the job as reported by the resource manager.
nodeAccessPolicy	**NodeAccessPolicy**	Yes	Specifies how node resources should be accessed. Note: If the job requirements array has more than one element that contains nodeAccessPolicy, only the first occurrence will be used.

Field Name	Type	POST	Description
nodeAllocationPolicy	NodeAllocationPolicy	Yes	Specifies how node resources should be selected and allocated to the job. Note: If the job requirements array has more than one element that contains nodeAllocationPolicy, only the first occurrence will be used.
nodeCount	**Integer**	Yes	The number of nodes required by the job.
nodeSet	**String**	Yes	The requested node set of the job. This must follow the format `SETSELECTION:SETTYPE[:SETLIST]` • SETSELECTION - `ANYOF`, `ONEOF`, or `FIRSTOF` • SETTYPE - `FEATURE` or `VARATTR` • SETLIST - For `FEATURE`, a comma-separated list of features. For `VARATTR`, a key=value pair. Examples: • `ONEOF:FEATURE:fastos,hiprio,bigmem` • `FIRSTOF:VARATTR:datacenter=Provo:datacenter=SaltLake`
nodes	Set<AllocatedNode>	No	Nodes that have been allocated to meet this requirement.
reservation	DomainProxy	No	The allocated reservation (assigned after the job has a reservation).
resourcesPerTask	Map<String, JobResource>	Yes	Contains requirements for disk, memory, processors, swap, and generic resources. For disk, memory, and swap, the unit is MB. For each resource, the "dedicated" field can be set during POST.
taskCount	**Integer**	Yes	The number of tasks (processors) required by this job.
tasksPerNode	**Integer**	Yes	The number of tasks to map to each node.

JobRequirementAttribute

Field Name	Type	POST	Description
comparator	**String**	Yes	The comparison operator. Valid values: • >= - Greater than or equal to • > - Greater than • <= - Less than • < - Less than • %= - Equals • %! - Not equals • Null - Defaults to %= • = - (Deprecated) Equivalent to %=
displayValue	**String**	Yes	The display value for the required attribute.
restriction	**JobRequirementAttributeRestriction**	Yes	The restriction of this attribute. May be null, but defaults to JobRequirementAttributeRestriction.must.
value	**String**	Yes	The value of the required attribute. During POST, if value is missing, blank, or null, do not provide a comparator.

JobRequirementAttributeRestriction

Represents a restriction for a job requirement attribute.

Value	Description
must	

JobRequirementFeaturesMode

Value	Description
OR	
AND	

NodeAccessPolicy

This enumeration describes how node resources will be shared by various tasks.

Value	Description
NONE	
SHARED	Tasks from any combination of jobs may utilize available resources.
SHAREDONLY	Only jobs requesting shared node access may utilize available resources.
SINGLEJOB	Tasks from a single job may utilize available resourceses.
SINGLETASK	A single task from a single job may run on the node.
SINGLEUSER	Tasks from any jobs owned by the same user may utilize available resources.
UNIQUEUSER	Any number of tasks from a single job may allocate resources from a node but only if the user has no other jobs running on that node.
SINGLEGROUP	Any number of tasks from the same group may utilize node.
SINGLEACCOUNT	Any number of tasks from the same account may utilize node.
SINGLECLASS	Any number of tasks from the same class may utilize node.
SINGLEQOS	Any number of tasks from the same QOS (quality of service) may utilize node.

NodeAllocationPolicy

Node Allocation enumeration.

Value	Description
FIRSTSET	
MINGLOBAL	
MINLOCAL	
PLUGIN	
NONE	No node allocation policy is specified. Moab defaults to MINRESOURCE when this is the case.
FIRSTAVAILABLE	Simple first come, first served algorithm where nodes are allocated in the order they are presented by the resource manager. This is a very simple, very fast algorithm.
LASTAVAILABLE	This algorithm selects resources so as to minimize the amount of time after the job and before the trailing reservation. This algorithm is a best fit in time algorithm which minimizes the impact of reservation based node-time fragmentation. It is useful in systems where a large number of reservations (job, standing, or administrative) are in place.
MINRESOURCE	This algorithm prioritizes nodes according to the configured resources on each node. Those nodes with the fewest configured resources which still meet the job's resource constraints are selected.
CPULOAD	Nodes are selected which have the maximum amount of available, unused cpu power, i.e. [# of CPU's] - [CPU load]. Good algorithm for timesharing node systems. This algorithm is only applied to jobs starting immediately. For the purpose of future reservations, the **MINRESOURCE** algorithm is used.
LOCAL	This will call the locally created **contrib** node allocation algorithm.
CONTIGUOUS	This algorithm will allocate nodes in contiguous (linear) blocks as required by the Compaq RMS system.

Value	Description
MAXBALANCE	This algorithm will attempt to allocate the most 'balanced' set of nodes possible to a job. In most cases, but not all, the metric for balance of the nodes is node speed. Thus, if possible, nodes with identical speeds will be allocated to the job. If identical speed nodes cannot be found, the algorithm will allocate the set of nodes with the minimum node speed 'span' or range.
PRIORITY	This algorithm allows a site to specify the priority of various static and dynamic aspects of compute nodes and allocate them with preference for higher priority nodes. It is highly flexible allowing node attribute and usage information to be combined with reservation affinity.
FASTEST	This algorithm will select nodes in 'fastest node first' order. Nodes will be selected by node speed if specified. If node speed is not specified, nodes will be selected by processor speed. If neither is specified, nodes will be selected in a random order.
PROCESSORLOAD	Alias for CPULOAD.
NODESPEED	Alias for FASTEST.
INREPORTEDORDER	Alias for FIRSTAVAILABLE.
INREVERSEREPORTEDORDER	Alias for LASTAVAILABLE.
CUSTOMPRIORITY	Alias for PRIORITY.
PROCESSORSPEEDBALANCE	Alias for MAXBALANCE.
MINIMUMCONFIGUREDRESOURCES	Alias for MINRESOURCE.
CRAY3DTORUS	Enable topology awareness scheduling algorithm.

AllocatedNode

Field Name	Type	POST	Description
name	**String**	No	

Field Name	Type	POST	Description
taskCount	**Integer**	No	

JobResource

Represents counts of dedicated and utilized resources.

Field Name	Type	POST	Description
dedicated	**Integer**	No	The amount of this resource that has been allocated for running workload.
utilized	**Integer**	No	The amount of this resource that is currently reported as utilized by resource managers.

JobResourceFailPolicyType

Value	Description
CANCEL	
FAIL	
HOLD	
IGNORE	
NOTIFY	
REQUEUE	

ResourceManager

Field Name	Type	POST	Description
isDestination	**Boolean**	No	

Field Name	Type	POST	Description
isSource	**Boolean**	No	
jobName	**String**	No	
name	**String**	No	

JobStateInformation

Field Name	Type	POST	Description
state	**JobState**	No	
stateExpected	**JobState**	No	
stateLastUpdatedDate	**Date**	No	
subState	**JobSubState**	No	

JobState

Value	Description
Idle	
Starting	
Running	
Removed	
Completed	
Hold	
Deferred	

Value	Description
Vacated	
NotQueued	
Unknown	
Staging	
Suspended	
Blocked	

JobSubState

Value	Description
Epilogue	
Migrated	
Preempted	
Prologue	

JobSystemJobType

Value	Description
generic	
osprovision	
osprovision2	
poweroff	

Value	Description
poweron	
reset	
storage	
vmmap	
vmmigrate	
vmtracking	

JobActionType

Value	Description
DESTROY	
MIGRATE	
MODIFY	

VMUsagePolicy

This enumeration describes the virtual machine requirements of a job

Value	Description
REQUIREPM	Requires a physical machine.
PREFPM	Prefers a physical machine.
CREATEVM	Creates a virtual machine.
CREATEPERSISTENTVM	Creates a virtual machine that doesn't go away after the job is done.
REQUIREVM	Requires a virtual machine.

Value	Description
PREFVM	Prefers a virtual machine.

API version 2

JobArray

Job arrays are an easy way to submit many sub-jobs that perform the same work using the same script, but operate on different sets of data. Sub-jobs are the jobs created by an array job and are identified by the array job ID and an index; for example, if 235[1] is an identifier, the number 235 is a job array ID, and 1 is the sub-job.

Field Name	Type	POST	Description
cancellationPolicy	**CancellationPolicyInformation**	Yes	Represents the cancellation policy to use for the job array.
indexRanges	**List<JobArrayIndexRange>**	Yes	The index ranges used to generate the sub-job indices. To use hard-coded values, see indexValues.
indexValues	**List<Long>**	Yes	The index values to use for the sub-jobs. To use ranges, see indexRanges.
jobPrototype	**Job**	Yes	The definition of the job to use for each sub-job.
name	**String**	Yes	The name of the job array. In MWS API version 1, this is stored in the `name` field of the created jobs. In MWS API version 2, this is stored in the `customName` field of the created jobs.
slotLimit	**Long**	Yes	(Optional) The number of sub-jobs in the array that can run at a time.

CancellationPolicyInformation

Job arrays can be canceled based on the success or failure of the first or any sub-job. This class represents the failure policies.

Field Name	Type	POST	Description
anyJob	**CancellationPolicy**	Yes	The cancellation policy based on the result of any sub-job. May be used in combination with firstJob.

Field Name	Type	POST	Description
firstJob	CancellationPolicy	Yes	The cancellation policy based on the result of the first sub-job (array index 1). May be used in combination with anyJob.

CancellationPolicy

This enumeration represents job array cancellation policies, and is to be used in combination with CancellationPolicyInformation.

Value	Description
SUCCESS	Cancels the job array if the specified sub-job succeeds.
FAILURE	Cancels the job array if the specified sub-job fails.

JobArrayIndexRange

Represents information about a job index expression. This is used when creating job arrays only.

Field Name	Type	POST	Description
endIndex	**Long**	Yes	The end of the index range. i.e. `10` for `1-10`.
increment	**Long**	Yes	The increment of the index range, defaults to 1 and must be greater than 0. For a range of 1-10 with an increment of 2, the list of indices will be [1, 3, 5, 7, 9].
startIndex	**Long**	Yes	The start of the index range. i.e. `1` for `1-10`.

Job

This class represents a job in the Moab Workload Manager. A job is a request for compute resources (CPUs, memory, storage) with which the requester can do work for a given amount of time. In an HPC environment, this might be a batch script to perform a Monte Carlo simulation. In a cloud environment, this would be a virtual machine and its associated storage. Moab will evaluate the request and assign the requested resources to the requester based on policies, current demand, and other factors in the data center. A job will also usually have some process that Moab starts automatically at the assigned start time. In an HPC environment, this can be

starting a batch script on the assigned nodes. In a cloud environment, this can be starting provisioning processes to create the virtual machine and storage and install software on it.

Field Name	Type	POST	Description
id	**String**	No	The unique identifier of this job. Note: this field is not user-assigned and is generated by the database.
arrayIndex	**Long**	No	If this job is a sub-job of a JobArray, this field contains the index of this job in the array. For example, if this job is `Moab.1[2]`, the array index would be `2`.
arrayMasterName	**String**	No	If this job is a sub-job of a JobArray, this field contains the name of the job array master. For example, if this job is `Moab.1[2]`, the array master name would be `Moab.1`.
attributes	**Set<String>**	Yes	The list of generic attributes associated with this job.
blocks	**Set<JobBlock>**	No	Reasons the job is blocked from running.
bypassCount	**Integer**	No	The number of times the job has been backfilled.
cancelCount	**Integer**	No	The number of times a job has received a cancel request.

Field Name	Type	POST	Description
commandFile	**String**	Yes	The name of the job script file (absolute path). If commandFile is set and commandScript is not set, then MWS must have read access to the file. If commandFile and commandScript are both set, then MWS does not read the contents of the file, but it does provide the name of the file to Moab.
commandLineArguments	**String**	Yes	The command line arguments passed to the job script specified by commandFile or commandScript.
commandScript	**String**	Yes	The contents of the job script. This field must be Base64-encoded.
completionCode	**Integer**	No	The exit code from this job.
cpuTime	**Long**	No	CPU usage time in seconds as reported by the resource manager.
credentials	JobCredentials	Yes	The credentials (user and group, for example) associated with this job.
customName	**String**	Yes	The user-specified name of this job.
dates	JobDates	Yes	Various dates associated with this job.
deferCount	**Integer**	No	The number of times a job has been deferred.

Field Name	Type	POST	Description
dependencies	Set<JobDependency>	Yes	Dependencies that must be fulfilled before the job can start.
description	**String**	No	The description of the job. Can be set only in a job template.
duration	**Long**	Yes	The length of time in seconds requested for the job. Note that it is possible to set duration to "INFINITY" if the AllowInfiniteJobs flag is set on the scheduler in the moab.cfg.
durationActive	**Long**	No	The length of time in seconds the job has been active or running.
durationQueued	**Long**	No	The length of time in seconds the job has been eligible to run in the queue.
durationRemaining	**Long**	No	An estimate of the time remaining, in seconds, before the job will complete.
durationSuspended	**Long**	No	The length of time in seconds the job has been suspended.
emailNotifyAddresses	**Set<String>**	Yes	The list of addresses to whom email is sent by the execution server.
emailNotifyTypes	Set<JobEmailNotifyType>	Yes	The list of email notify types attached to the job.
environmentRequested	**Boolean**	Yes	Setting this field to true tells the Moab Workload Manager to set various variables, if populated, in the job's environment.

Field Name	Type	POST	Description
environmentVariables	**Map<String, Map>**	Yes	The list of environment variables for this job.
epilogScript	**String**	Yes	The path to the TORQUE epilog script.
flags	**Set<JobFlag>**	Yes	The flags that are set on this job.
holdDate	**Date**	No	The date the most recent hold was placed on the job.
holdReason	**JobHoldReason**	No	The reason the job is on hold.
holds	**Set<JobHoldType>**	Yes	The holds that are set on the job. The "User" hold type is valid during POST.
initialWorkingDirectory	**String**	Yes	The path to the directory in which the job will be started.
isActive	**Boolean**	No	True if the job is active, false if the job is complete.
jobGroup	**String**	Yes	The job group to which this job belongs (different from credentials.group).
masterNode	**DomainProxy**	No	The first node in the list of allocated nodes for this job. For TORQUE jobs, this represents the "mother superior."
memorySecondsDedicated	**Double**	No	The memory seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.

Field Name	Type	POST	Description
memorySecondsUtilized	**Double**	No	The memory seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
messages	**Set<Message>**	No	The list of messages associated with the job. The "message" field is valid during PUT.
migrateCount	**Integer**	No	The number of times the job has been migrated.
minimumPreemptTime	**Long**	No	The minimum length of time, in seconds, an active job must be running before it is eligible for preemption.
mwmName	**String**	No	The name of the Moab Workload Manager instance that owns this job.
name	**String**	No	The name of this job. This name is unique *per instance* of Moab Workload Manager (i.e. not globally).
nodesExcluded	**Set<DomainProxy>**	Yes	The list of nodes that should not be considered for this job.
nodesRequested	**Set<DomainProxy>**	Yes	The exact set, superset, or subset of nodes on which this job must run. (See also: nodesRequestedPolicy.)
nodesRequestedPolicy	**JobHostListMode**	Yes	Indicates an exact set, superset, or subset of nodes on which the job must run. Only relevant if nodesRequested is provided. (See also: nodesRequested.)

Field Name	Type	POST	Description
partitionAccessList	**Set<String>**	No	The list of partitions that this job can access.
partitionAccessListRequested	**Set<String>**	Yes	The list of partitions that this job has requested.
preemptCount	**Integer**	No	The number of times the job has been preempted.
priorities	JobPriority	Yes	The list of priorities for the job.
processorSecondsDedicated	**Double**	No	The processor seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.
processorSecondsLimit	**Double**	No	The limit for processorSecondsUtilized.
processorSecondsUtilized	**Double**	No	The processor seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
prologScript	**String**	Yes	The path to the TORQUE prolog script.
queueStatus	JobQueueStatus	No	The status of the job in its queue.
rejectPolicies	Set<JobRejectPolicy>	No	The list of policies enabled when a job is rejected.
requirements	Set<JobRequirement>	Yes	The list of items required for this job to run. Only JobRequirement.features is valid during PUT.

Field Name	Type	POST	Description
reservationRequested	DomainProxy	Yes	The reservation that the job requested.
resourceFailPolicy	JobResourceFailPolicyType	Yes	The policy that dictates what should happen to the job if it is running and at least one of the resources it is using fails.
resourceManagerExtension	**String**	Yes	If provided during POST, this string will be added to the resource manager extension section of the job submission. Example: "bandwidth=120;queuejob=false" Note that the delimiter between resourceManagerExtension elements is the semicolon.
resourceManagers	Set<ResourceManager>	No	The list of resource managers associated with this job.
rmStandardErrorFilePath	**String**	No	The path to the remote file containing the standard error of the job.
rmStandardOutputFilePath	**String**	No	The path to the remote file containing the standard output of the job.
shellName	**String**	Yes	Declares the shell that interprets the job script.
standardErrorFilePath	**String**	Yes	The path to the file containing the standard error of the job.
standardOutputFilePath	**String**	Yes	The path to the file containing the standard output of the job.

Field Name	Type	POST	Description
startCount	**Integer**	No	The number of times the job has been started.
states	JobStateInformation	No	Information about the state of the job.
submitHost	**String**	Yes	The host from which the job was submitted.
systemJobAction	**String**	No	The action the system job will take.
systemJobType	JobSystemJobType	No	The type of system job. In the Moab Cloud Suite, this will usually be "vmtracking" or "generic."
targetedJobAction	JobActionType	No	The action that this job is performing on another job.
targetedJobName	**String**	No	The name of the job on which this job is performing the targetedJobAction.
templates	Set<DomainProxy>	Yes	The list of all job templates to be set on this job.
triggers	**Set<String>**	No	The list of triggers associated with this job.
variables	**Map<String, Map>**	Yes	The list of variables that this job owns or sets on completion.
virtualContainers	Set<DomainProxy>	Yes	When submitting this job, add it to the specified existing virtual container. Valid during POST, but only one virtual container can be specified.

Field Name	Type	POST	Description
virtualMachines	Set<DomainProxy>	No	The list of virtual machines that are allocated to this job.
vmUsagePolicy	VMUsagePolicy	Yes	The requested Virtual Machine Usage Policy for this job.

JobBlock

Field Name	Type	POST	Description
category	JobBlockCategory	No	
message	**String**	No	
type	JobBlockType	No	

JobBlockCategory

Value	Description
depend	
jobBlock	
migrate	

JobBlockType

Value	Description
ActivePolicy	
BadUser	
Dependency	

Value	Description
EState	
FairShare	
Hold	
IdlePolicy	
LocalPolicy	
NoClass	
NoData	
NoResource	
NoTime	
PartitionAccess	
Priority	
RMSubmissionFailure	
StartDate	
State	
SysLimits	

JobCredentials

Moab Workload Manager supports the concept of credentials, which provide a means of attributing policy and resource access to entities such as users and groups. These credentials allow specification of job ownership, tracking of resource usage, enforcement of policies, and many other features.

Field Name	Type	POST	Description
account	**String**	Yes	The account credential is also referred to as the project. This credential is generally associated with a group of users along the lines of a particular project for accounting and billing purposes.
group	**String**	Yes	The group credential represents an aggregation of users. User-to-group mappings are often specified by the operating system or resource manager and typically map to a user's UNIX group ID. However, user-to-group mappings may also be provided by a security and identity management service, or you can specify such directly within Moab.
jobClass	**String**	Yes	The concept of the class credential is derived from the resource manager class or queue object. Classes differ from other credentials in that they more directly impact job attributes. In standard HPC usage, a user submits a job to a class and this class imposes a number of factors on the job. The attributes of a class may be specified within the resource manager or directly within Moab.
qos	**String**	No	The quality of service assigned to this job. The concept of a quality of service (QoS) credential is unique to Moab and is not derived from any underlying concept or peer service. In most cases, the QoS credential is used to allow a site to set up a selection of service levels for end-users to choose from on a long-term or job-by-job basis. QoS's differ from other credentials in that they are centered around special access where this access may allow use of additional services, additional resources, or improved responsiveness. Unique to this credential, organizations may also choose to apply different charge rates to the varying levels of service available within each QoS. As QoS is an internal credential, all QoS configuration occurs within Moab.
qosRequested	**String**	Yes	The quality of service requested for this job.
user	**String**	Yes	The user credential is the fundamental credential within a workload manager; each job requires an association with exactly one user. In fact, the user credential is the only required credential in Moab; all others are optional. In most cases, the job's user credential is configured within or managed by the operating system itself, although Moab may be configured to obtain this information from an independent security and identity management service.

JobDates

Field Name	Type	POST	Description
completedDate	**Date**	No	
createdDate	**Date**	No	
deadlineDate	**Date**	Yes	The deadline for completion of the job.
dispatchedDate	**Date**	No	
earliestRequestedStartDate	**Date**	Yes	The job will start no sooner than this date.
earliestStartDate	**Date**	No	
eligibleDate	**Date**	No	
lastCanceledDate	**Date**	No	
lastChargedDate	**Date**	No	
lastPreemptedDate	**Date**	No	
lastUpdatedDate	**Date**	No	
startDate	**Date**	No	
submitDate	**Date**	No	
terminationDate	**Date**	No	

JobDependency

Field Name	Type	POST	Description
name	**String**	Yes	The name of the object on on which the job is dependent.
type	**JobDependencyType**	Yes	The type of job dependency. Only set is valid for POST.

Field Name	Type	POST	Description
value	**String**	No	

JobDependencyType

Represents the type of a job dependency. For now, only the "set" type is supported.

Value	Description
set	

JobEmailNotifyType

Value	Description
JobStart	
JobEnd	
JobFail	
All	

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.
ADVRES	The job requires the use of a reservation.

Value	Description
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.
INTERACTIVE	The job needs to interactive input from the user to run.

Value	Description
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.

Value	Description
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

JobHoldReason

Value	Description
Admin	
NoResources	
SystemLimitsExceeded	

Value	Description
BankFailure	
CannotDebitAccount	
InvalidAccount	
RMFailure	
RMReject	
PolicyViolation	
CredAccess	
CredHold	
PreReq	
Data	
Security	
MissingDependency	

JobHoldType

Value	Description
User	
System	
Batch	
Defer	
All	

DomainProxy

A reference to an object contained within an object. For example, a Virtual Machine object contains a reference to the Node on which it is running. That reference is represented by this class.

Field Name	Type	POST	Description
name	**String**	Yes	The name of the object.

Message

Field Name	Type	POST	Description
count	**Integer**	No	The number of times this message has occurred.
createdDate	**Date**	No	The date this message was created.
expireDate	**Date**	No	The date this message expires.
message	**String**	No	The message itself.

JobHostListMode

Value	Description
superset	
subset	
exactset	

JobPriority

Field Name	Type	POST	Description
run	**Long**	No	

Field Name	Type	POST	Description
start	**Long**	No	
system	**Long**	No	
user	**Long**	Yes	The user-requested priority for the job. By default, the range is between -1024 and 0. To enable priority range from -1024 to +1023, set `ENABLEPOSUSERPRIORITY` in the `moab.cfg` file.

JobQueueStatus

Value	Description
active	
blocked	
completed	
eligible	

JobRejectPolicy

Value	Description
CANCEL	
HOLD	
IGNORE	
MAIL	
RETRY	

JobRequirement

Field Name	Type	POST	Description
architecture	**String**	Yes	The architecture required by the job.
attributes	**Map<String, JobRequirementAttribute>**	Yes	Required node attributes with version number support.
features	**Set<String>**	No	The list of node features the job is scheduled against.
featuresExcluded	**Set<String>**	Yes	Excluded node features. That is, do not select nodes with these features. (See also: featuresExcludedMode.)
featuresExcludedMode	**JobRequirementFeaturesMode**	Yes	Indicates whether excluded features should be AND'ed or OR'd. The default is AND. Only relevant if featuresExcluded is provided. (See also: featuresExcluded.)
featuresRequested	**Set<String>**	Yes	Requested node features. (See also: featuresRequestedMode.)
featuresRequestedMode	**JobRequirementFeaturesMode**	Yes	Indicates whether requested features should be AND'ed or OR'd. The default is AND. Only relevant if featuresRequested is provided. (See also: featuresRequested.)
image	**String**	Yes	The image required by the job.
index	**Integer**	No	The index of the requirement, starting with 0.
metrics	**Map<String, Double>**	No	Generic metrics associated with the job as reported by the resource manager.
nodeAccessPolicy	**NodeAccessPolicy**	Yes	Specifies how node resources should be accessed. Note: If the job requirements array has more than one element that contains nodeAccessPolicy, only the first occurrence will be used.

Field Name	Type	POST	Description
nodeAllocationPolicy	NodeAllocationPolicy	Yes	Specifies how node resources should be selected and allocated to the job. Note: If the job requirements array has more than one element that contains nodeAllocationPolicy, only the first occurrence will be used.
nodeCount	**Integer**	Yes	The number of nodes required by the job.
nodeSet	**String**	Yes	The requested node set of the job. This must follow the format `SETSELECTION:SETTYPE[:SETLIST]` • SETSELECTION - `ANYOF`, `ONEOF`, or `FIRSTOF` • SETTYPE - `FEATURE` or `VARATTR` • SETLIST - For `FEATURE`, a comma-separated list of features. For `VARATTR`, a key=value pair. Examples: • `ONEOF:FEATURE:fastos,hiprio,bigmem` • `FIRSTOF:VARATTR:datacenter=Provo:datacenter=SaltLake`
nodes	Set<AllocatedNode>	No	Nodes that have been allocated to meet this requirement.
reservation	DomainProxy	No	The allocated reservation (assigned after the job has a reservation).
resourcesPerTask	Map<String, JobResource>	Yes	Contains requirements for disk, memory, processors, swap, and generic resources. For disk, memory, and swap, the unit is MB. For each resource, the "dedicated" field can be set during POST.
taskCount	**Integer**	Yes	The number of tasks (processors) required by this job.
tasksPerNode	**Integer**	Yes	The number of tasks to map to each node.

JobRequirementAttribute

Field Name	Type	POST	Description
comparator	**String**	Yes	The comparison operator. Valid values: • >= - Greater than or equal to • > - Greater than • <= - Less than • < - Less than • %= - Equals • %! - Not equals • Null - Defaults to %= • = - (Deprecated) Equivalent to %=
displayValue	**String**	Yes	The display value for the required attribute.
restriction	**JobRequirementAttributeRestriction**	Yes	The restriction of this attribute. May be null, but defaults to JobRequirementAttributeRestriction.must.
value	**String**	Yes	The value of the required attribute. During POST, if value is missing, blank, or null, do not provide a comparator.

JobRequirementAttributeRestriction

Represents a restriction for a job requirement attribute.

Value	Description
must	

JobRequirementFeaturesMode

Value	Description
OR	
AND	

NodeAccessPolicy

This enumeration describes how node resources will be shared by various tasks.

Value	Description
NONE	
SHARED	Tasks from any combination of jobs may utilize available resources.
SHAREDONLY	Only jobs requesting shared node access may utilize available resources.
SINGLEJOB	Tasks from a single job may utilize available resourceses.
SINGLETASK	A single task from a single job may run on the node.
SINGLEUSER	Tasks from any jobs owned by the same user may utilize available resources.
UNIQUEUSER	Any number of tasks from a single job may allocate resources from a node but only if the user has no other jobs running on that node.
SINGLEGROUP	Any number of tasks from the same group may utilize node.
SINGLEACCOUNT	Any number of tasks from the same account may utilize node.
SINGLECLASS	Any number of tasks from the same class may utilize node.
SINGLEQOS	Any number of tasks from the same QOS (quality of service) may utilize node.

NodeAllocationPolicy

Node Allocation enumeration.

Value	Description
FIRSTSET	
MINGLOBAL	
MINLOCAL	
PLUGIN	
NONE	No node allocation policy is specified. Moab defaults to MINRESOURCE when this is the case.
FIRSTAVAILABLE	Simple first come, first served algorithm where nodes are allocated in the order they are presented by the resource manager. This is a very simple, very fast algorithm.
LASTAVAILABLE	This algorithm selects resources so as to minimize the amount of time after the job and before the trailing reservation. This algorithm is a best fit in time algorithm which minimizes the impact of reservation based node-time fragmentation. It is useful in systems where a large number of reservations (job, standing, or administrative) are in place.
MINRESOURCE	This algorithm prioritizes nodes according to the configured resources on each node. Those nodes with the fewest configured resources which still meet the job's resource constraints are selected.
CPULOAD	Nodes are selected which have the maximum amount of available, unused cpu power, i.e. [# of CPU's] - [CPU load]. Good algorithm for timesharing node systems. This algorithm is only applied to jobs starting immediately. For the purpose of future reservations, the **MINRESOURCE** algorithm is used.
LOCAL	This will call the locally created **contrib** node allocation algorithm.
CONTIGUOUS	This algorithm will allocate nodes in contiguous (linear) blocks as required by the Compaq RMS system.

Value	Description
MAXBALANCE	This algorithm will attempt to allocate the most 'balanced' set of nodes possible to a job. In most cases, but not all, the metric for balance of the nodes is node speed. Thus, if possible, nodes with identical speeds will be allocated to the job. If identical speed nodes cannot be found, the algorithm will allocate the set of nodes with the minimum node speed 'span' or range.
PRIORITY	This algorithm allows a site to specify the priority of various static and dynamic aspects of compute nodes and allocate them with preference for higher priority nodes. It is highly flexible allowing node attribute and usage information to be combined with reservation affinity.
FASTEST	This algorithm will select nodes in 'fastest node first' order. Nodes will be selected by node speed if specified. If node speed is not specified, nodes will be selected by processor speed. If neither is specified, nodes will be selected in a random order.
PROCESSORLOAD	Alias for CPULOAD.
NODESPEED	Alias for FASTEST.
INREPORTEDORDER	Alias for FIRSTAVAILABLE.
INREVERSEREPORTEDORDER	Alias for LASTAVAILABLE.
CUSTOMPRIORITY	Alias for PRIORITY.
PROCESSORSPEEDBALANCE	Alias for MAXBALANCE.
MINIMUMCONFIGUREDRESOURCES	Alias for MINRESOURCE.
CRAY3DTORUS	Enable topology awareness scheduling algorithm.

AllocatedNode

Field Name	Type	POST	Description
name	**String**	No	

Field Name	Type	POST	Description
taskCount	**Integer**	No	

JobResource

Represents counts of dedicated and utilized resources.

Field Name	Type	POST	Description
dedicated	**Integer**	No	The amount of this resource that has been allocated for running workload.
utilized	**Integer**	No	The amount of this resource that is currently reported as utilized by resource managers.

JobResourceFailPolicyType

Value	Description
CANCEL	
FAIL	
HOLD	
IGNORE	
NOTIFY	
REQUEUE	

ResourceManager

Field Name	Type	POST	Description
isDestination	**Boolean**	No	

Field Name	Type	POST	Description
isSource	**Boolean**	No	
jobName	**String**	No	
name	**String**	No	

JobStateInformation

Field Name	Type	POST	Description
state	**JobState**	No	
stateExpected	**JobState**	No	
stateLastUpdatedDate	**Date**	No	
subState	**JobSubState**	No	

JobState

Value	Description
Idle	
Starting	
Running	
Removed	
Completed	
Hold	
Deferred	

Value	Description
Vacated	
NotQueued	
Unknown	
Staging	
Suspended	
Blocked	

JobSubState

Value	Description
Epilogue	
Migrated	
Preempted	
Prologue	

JobSystemJobType

Value	Description
generic	
osprovision	
osprovision2	
poweroff	

Value	Description
poweron	
reset	
storage	
vmmap	
vmmigrate	
vmtracking	

JobActionType

Value	Description
DESTROY	
MIGRATE	
MODIFY	

VMUsagePolicy

This enumeration describes the virtual machine requirements of a job

Value	Description
REQUIREPM	Requires a physical machine.
PREFPM	Prefers a physical machine.
CREATEVM	Creates a virtual machine.
CREATEPERSISTENTVM	Creates a virtual machine that doesn't go away after the job is done.
REQUIREVM	Requires a virtual machine.

Value	Description
PREFVM	Prefers a virtual machine.

Related Topics

- Job Arrays

Fields: Jobs

See the associated Jobs resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`jobs`	Permissions
Hooks filename	`jobs.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Job

This class represents a job in the Moab Workload Manager. A job is a request for compute resources (CPUs, memory, storage) with which the requester can do work for a given amount of time. In an HPC environment, this might be a batch script to perform a Monte Carlo simulation. In a cloud environment, this would be a virtual machine and its associated storage. Moab will evaluate the request and assign the requested resources to the requester based on policies, current demand, and other factors in the data center. A job will also usually have some process that Moab starts automatically at the assigned start time. In an HPC environment, this can be starting a batch script on the assigned nodes. In a cloud environment, this can be starting provisioning processes to create the virtual machine and storage and install software on it.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique identifier of this job. Note: this field is not user-assigned and is generated by the database.
arrayIndex	**Long**	No	No	If this job is a sub-job of a JobArray, this field contains the index of this job in the array. For example, if this job is `Moab.1[2]`, the array index would be `2`.
arrayMasterName	**String**	No	No	If this job is a sub-job of a JobArray, this field contains the name of the job array master. For example, if this job is `Moab.1[2]`, the array master name would be `Moab.1`.
attributes	**Set<String>**	Yes	No	The list of generic attributes associated with this job.
blocks	**Set<JobBlock>**	No	No	Reasons the job is blocked from running.

Field Name	Type	POST	PUT	Description
bypassCount	**Integer**	No	No	The number of times the job has been backfilled.
cancelCount	**Integer**	No	No	The number of times a job has received a cancel request.
commandFile	**String**	Yes	No	The name of the job script file (absolute path). If commandFile is set and commandScript is not set, then MWS must have read access to the file. If commandFile and commandScript are both set, then MWS does not read the contents of the file, but it does provide the name of the file to Moab.
commandLineArguments	**String**	Yes	No	The command line arguments passed to the job script specified by commandFile or commandScript.
commandScript	**String**	Yes	No	The contents of the job script. This field must be Base64-encoded.
completionCode	**Integer**	No	No	The exit code from this job.
cpuTime	**Long**	No	No	CPU usage time in seconds as reported by the resource manager.
credentials	JobCredentials	Yes	Yes	The credentials (user and group, for example) associated with this job.
customName	**String**	Yes	Yes	The user-specified name of this job.

Field Name	Type	POST	PUT	Description
dates	JobDates	Yes	Yes	Various dates associated with this job.
deferCount	**Integer**	No	No	The number of times a job has been deferred.
dependencies	Set<JobDependency>	Yes	No	Dependencies that must be fulfilled before the job can start.
description	**String**	No	No	The description of the job. Can be set only in a job template.
duration	**Long**	Yes	Yes	The length of time in seconds requested for the job. Note that it is possible to set duration to "INFINITY" if the AllowInfiniteJobs flag is set on the scheduler in the moab.cfg.
durationActive	**Long**	No	No	The length of time in seconds the job has been active or running.
durationQueued	**Long**	No	No	The length of time in seconds the job has been eligible to run in the queue.
durationRemaining	**Long**	No	No	An estimate of the time remaining, in seconds, before the job will complete.
durationSuspended	**Long**	No	No	The length of time in seconds the job has been suspended.
emailNotifyAddresses	**Set<String>**	Yes	No	The list of addresses to whom email is sent by the execution server.

Field Name	Type	POST	PUT	Description
emailNotifyTypes	Set<JobEmailNotifyType>	Yes	No	The list of email notify types attached to the job.
environmentRequested	**Boolean**	Yes	No	Setting this field to true tells the Moab Workload Manager to set various variables, if populated, in the job's environment.
environmentVariables	**Map<String, Map>**	Yes	No	The list of environment variables for this job.
epilogScript	**String**	Yes	No	The path to the TORQUE epilog script.
flags	Set<JobFlag>	Yes	Yes	The flags that are set on this job.
holdDate	**Date**	No	No	The date the most recent hold was placed on the job.
holdReason	JobHoldReason	No	No	The reason the job is on hold.
holds	Set<JobHoldType>	Yes	Yes	The holds that are set on the job. The "User" hold type is valid during POST.
initialWorkingDirectory	**String**	Yes	No	The path to the directory in which the job will be started.
isActive	**Boolean**	No	No	True if the job is active, false if the job is complete.
jobGroup	**String**	Yes	No	The job group to which this job belongs (different from credentials.group).

Field Name	Type	POST	PUT	Description
masterNode	DomainProxy	No	No	The first node in the list of allocated nodes for this job. For TORQUE jobs, this represents the "mother superior."
memorySecondsDedicated	**Double**	No	No	The memory seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.
memorySecondsUtilized	**Double**	No	No	The memory seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
messages	Set<Message>	No	Yes	The list of messages associated with the job. The "message" field is valid during PUT.
migrateCount	**Integer**	No	No	The number of times the job has been migrated.
minimumPreemptTime	**Long**	No	No	The minimum length of time, in seconds, an active job must be running before it is eligible for preemption.
mwmName	**String**	No	No	The name of the Moab Workload Manager instance that owns this job.
name	**String**	No	No	The name of this job. This name is unique *per instance* of Moab Workload Manager (i.e. not globally).

Field Name	Type	POST	PUT	Description
nodesExcluded	**Set<DomainProxy>**	Yes	No	The list of nodes that should not be considered for this job.
nodesRequested	**Set<DomainProxy>**	Yes	No	The exact set, superset, or subset of nodes on which this job must run. (See also: nodesRequestedPolicy.)
nodesRequestedPolicy	**JobHostListMode**	Yes	No	Indicates an exact set, superset, or subset of nodes on which the job must run. Only relevant if nodesRequested is provided. (See also: nodesRequested.)
partitionAccessList	**Set<String>**	No	No	The list of partitions that this job can access.
partitionAccessListRequested	**Set<String>**	Yes	Yes	The list of partitions that this job has requested.
preemptCount	**Integer**	No	No	The number of times the job has been preempted.
priorities	**JobPriority**	Yes	Yes	The list of priorities for the job.
processorSecondsDedicated	**Double**	No	No	The processor seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.
processorSecondsLimit	**Double**	No	No	The limit for processorSecondsUtilized.

Field Name	Type	POST	PUT	Description
processorSecondsUtilized	**Double**	No	No	The processor seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
prologScript	**String**	Yes	No	The path to the TORQUE prolog script.
queueStatus	**JobQueueStatus**	No	No	The status of the job in its queue.
rejectPolicies	**Set<JobRejectPolicy>**	No	No	The list of policies enabled when a job is rejected.
requirements	**Set<JobRequirement>**	Yes	Yes	The list of items required for this job to run. Only JobRequirement.features is valid during PUT.
reservationRequested	**DomainProxy**	Yes	Yes	The reservation that the job requested.
resourceFailPolicy	**JobResourceFailPolicyType**	Yes	No	The policy that dictates what should happen to the job if it is running and at least one of the resources it is using fails.
resourceManagerExtension	**String**	Yes	No	If provided during POST, this string will be added to the resource manager extension section of the job submission. Example: "bandwidth=120;queuejob=false" Note that the delimiter between resourceManagerExtension elements is the semicolon.

Field Name	Type	POST	PUT	Description
resourceManagers	Set<ResourceManager>	No	No	The list of resource managers associated with this job.
rmStandardErrorFilePath	**String**	No	No	The path to the remote file containing the standard error of the job.
rmStandardOutputFilePath	**String**	No	No	The path to the remote file containing the standard output of the job.
shellName	**String**	Yes	No	Declares the shell that interprets the job script.
standardErrorFilePath	**String**	Yes	No	The path to the file containing the standard error of the job.
standardOutputFilePath	**String**	Yes	No	The path to the file containing the standard output of the job.
startCount	**Integer**	No	No	The number of times the job has been started.
states	JobStateInformation	No	No	Information about the state of the job.
submitHost	**String**	Yes	No	The host from which the job was submitted.
systemJobAction	**String**	No	No	The action the system job will take.
systemJobType	JobSystemJobType	No	No	The type of system job. In the Moab Cloud Suite, this will usually be "vmtracking" or "generic."

Field Name	Type	POST	PUT	Description
targetedJobAction	JobActionType	No	No	The action that this job is performing on another job.
targetedJobName	**String**	No	No	The name of the job on which this job is performing the targetedJobAction.
templates	Set<DomainProxy>	Yes	No	The list of all job templates to be set on this job.
triggers	**Set<String>**	No	No	The list of triggers associated with this job.
variables	**Map<String, Map>**	Yes	Yes	The list of variables that this job owns or sets on completion.
virtualContainers	Set<DomainProxy>	Yes	No	When submitting this job, add it to the specified existing virtual container. Valid during POST, but only one virtual container can be specified.
virtualMachines	Set<DomainProxy>	No	No	The list of virtual machines that are allocated to this job.
vmUsagePolicy	VMUsagePolicy	Yes	No	The requested Virtual Machine Usage Policy for this job.

JobBlock

Field Name	Type	POST	PUT	Description
category	JobBlockCategory	No	No	
message	**String**	No	No	
type	JobBlockType	No	No	

JobBlockCategory

Value	Description
depend	
jobBlock	
migrate	

JobBlockType

Value	Description
ActivePolicy	
BadUser	
Dependency	
EState	
FairShare	
Hold	
IdlePolicy	
LocalPolicy	
NoClass	
NoData	
NoResource	
NoTime	
PartitionAccess	

Value	Description
Priority	
RMSubmissionFailure	
StartDate	
State	
SysLimits	

JobCredentials

Moab Workload Manager supports the concept of credentials, which provide a means of attributing policy and resource access to entities such as users and groups. These credentials allow specification of job ownership, tracking of resource usage, enforcement of policies, and many other features.

Field Name	Type	POST	PUT	Description
account	**String**	Yes	Yes	The account credential is also referred to as the project. This credential is generally associated with a group of users along the lines of a particular project for accounting and billing purposes.
group	**String**	Yes	No	The group credential represents an aggregation of users. User-to-group mappings are often specified by the operating system or resource manager and typically map to a user's UNIX group ID. However, user-to-group mappings may also be provided by a security and identity management service, or you can specify such directly within Moab.
jobClass	**String**	Yes	Yes	The concept of the class credential is derived from the resource manager class or queue object. Classes differ from other credentials in that they more directly impact job attributes. In standard HPC usage, a user submits a job to a class and this class imposes a number of factors on the job. The attributes of a class may be specified within the resource manager or directly within Moab.

Field Name	Type	POST	PUT	Description
qos	**String**	No	No	The quality of service assigned to this job. The concept of a quality of service (QoS) credential is unique to Moab and is not derived from any underlying concept or peer service. In most cases, the QoS credential is used to allow a site to set up a selection of service levels for end-users to choose from on a long-term or job-by-job basis. QoS's differ from other credentials in that they are centered around special access where this access may allow use of additional services, additional resources, or improved responsiveness. Unique to this credential, organizations may also choose to apply different charge rates to the varying levels of service available within each QoS. As QoS is an internal credential, all QoS configuration occurs within Moab.
qosRequested	**String**	Yes	Yes	The quality of service requested for this job.
user	**String**	Yes	No	The user credential is the fundamental credential within a workload manager; each job requires an association with exactly one user. In fact, the user credential is the only required credential in Moab; all others are optional. In most cases, the job's user credential is configured within or managed by the operating system itself, although Moab may be configured to obtain this information from an independent security and identity management service.

JobDates

Field Name	Type	POST	PUT	Description
completedDate	**Date**	No	No	
createdDate	**Date**	No	No	
deadlineDate	**Date**	Yes	No	The deadline for completion of the job.
dispatchedDate	**Date**	No	No	
earliestRequestedStartDate	**Date**	Yes	Yes	The job will start no sooner than this date.

Field Name	Type	POST	PUT	Description
earliestStartDate	**Date**	No	No	
eligibleDate	**Date**	No	No	
lastCanceledDate	**Date**	No	No	
lastChargedDate	**Date**	No	No	
lastPreemptedDate	**Date**	No	No	
lastUpdatedDate	**Date**	No	No	
startDate	**Date**	No	No	
submitDate	**Date**	No	No	
terminationDate	**Date**	No	No	

JobDependency

Field Name	Type	POST	PUT	Description
name	**String**	Yes	No	The name of the object on on which the job is dependent.
type	**JobDependencyType**	Yes	No	The type of job dependency. Only set is valid for POST.
value	**String**	No	No	

JobDependencyType

Represents the type of a job dependency. For now, only the "set" type is supported.

Value	Description
set	

JobEmailNotifyType

Value	Description
JobStart	
JobEnd	
JobFail	
All	

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.
ADVRES	The job requires the use of a reservation.
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.

Value	Description
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.
INTERACTIVE	The job needs to interactive input from the user to run.
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.

Value	Description
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.

Value	Description
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

JobHoldReason

Value	Description
Admin	
NoResources	
SystemLimitsExceeded	
BankFailure	
CannotDebitAccount	
InvalidAccount	
RMFailure	
RMReject	
PolicyViolation	
CredAccess	

Value	Description
CredHold	
PreReq	
Data	
Security	
MissingDependency	

JobHoldType

Value	Description
User	
System	
Batch	
Defer	
All	

DomainProxy

A reference to an object contained within an object. For example, a Virtual Machine object contains a reference to the Node on which it is running. That reference is represented by this class.

Field Name	Type	POST	PUT	Description
name	**String**	Yes	No	The name of the object.

Message

Field Name	Type	POST	PUT	Description
count	**Integer**	No	No	The number of times this message has occurred.
createdDate	**Date**	No	No	The date this message was created.
expireDate	**Date**	No	No	The date this message expires.
message	**String**	No	Yes	The message itself.

JobHostListMode

Value	Description
superset	
subset	
exactset	

JobPriority

Field Name	Type	POST	PUT	Description
run	**Long**	No	No	
start	**Long**	No	No	
system	**Long**	No	No	
user	**Long**	Yes	Yes	The user-requested priority for the job. By default, the range is between -1024 and 0. To enable priority range from -1024 to +1023, set `ENABLEPOSUSERPRIORITY` in the `moab.cfg` file.

JobQueueStatus

Value	Description
active	
blocked	
completed	
eligible	

JobRejectPolicy

Value	Description
CANCEL	
HOLD	
IGNORE	
MAIL	
RETRY	

JobRequirement

Field Name	Type	POST	PUT	Description
architecture	**String**	Yes	No	The architecture required by the job.
attributes	**Map<String, JobRequirementAttribute>**	Yes	No	Required node attributes with version number support.

Field Name	Type	POST	PUT	Description
features	**Set<String>**	No	Yes	The list of node features the job is scheduled against.
featuresExcluded	**Set<String>**	Yes	No	Excluded node features. That is, do not select nodes with these features. (See also: featuresExcludedMode.)
featuresExcludedMode	**JobRequirementFeaturesMode**	Yes	No	Indicates whether excluded features should be AND'ed or OR'd. The default is AND. Only relevant if featuresExcluded is provided. (See also: featuresExcluded.)
featuresRequested	**Set<String>**	Yes	No	Requested node features. (See also: featuresRequestedMode.)
featuresRequestedMode	**JobRequirementFeaturesMode**	Yes	No	Indicates whether requested features should be AND'ed or OR'd. The default is AND. Only relevant if featuresRequested is provided. (See also: featuresRequested.)
image	**String**	Yes	No	The image required by the job.
index	**Integer**	No	No	The index of the requirement, starting with 0.
metrics	**Map<String, Double>**	No	No	Generic metrics associated with the job as reported by the resource manager.
nodeAccessPolicy	**NodeAccessPolicy**	Yes	No	Specifies how node resources should be accessed. Note: If the job requirements array has more than one element that contains nodeAccessPolicy, only the first occurrence will be used.

Field Name	Type	POST	PUT	Description
nodeAllocationPolicy	NodeAllocationPolicy	Yes	No	Specifies how node resources should be selected and allocated to the job. Note: If the job requirements array has more than one element that contains nodeAllocationPolicy, only the first occurrence will be used.
nodeCount	**Integer**	Yes	No	The number of nodes required by the job.
nodeSet	**String**	Yes	No	The requested node set of the job. This must follow the format `SETSELECTION:SETTYPE[:SETLIST]` • SETSELECTION - `ANYOF`, `ONEOF`, or `FIRSTOF` • SETTYPE - `FEATURE` or `VARATTR` • SETLIST - For `FEATURE`, a comma-separated list of features. For `VARATTR`, a key=value pair. Examples: • `ONEOF:FEATURE:fastos,hiprio,bigmem` • `FIRSTOF:VARATTR:datacenter=Provo:datacenter=SaltLake`
nodes	Set<AllocatedNode>	No	No	Nodes that have been allocated to meet this requirement.
reservation	DomainProxy	No	No	The allocated reservation (assigned after the job has a reservation).

Field Name	Type	POST	PUT	Description
resourcesPerTask	Map<String, JobResource>	Yes	No	Contains requirements for disk, memory, processors, swap, and generic resources. For disk, memory, and swap, the unit is MB. For each resource, the "dedicated" field can be set during POST.
taskCount	**Integer**	Yes	No	The number of tasks (processors) required by this job.
tasksPerNode	**Integer**	Yes	No	The number of tasks to map to each node.

JobRequirementAttribute

Field Name	Type	POST	PUT	Description
comparator	**String**	Yes	No	The comparison operator. Valid values: • >= - Greater than or equal to • > - Greater than • <= - Less than • < - Less than • %= - Equals • %! - Not equals • Null - Defaults to %= • = - (Deprecated) Equivalent to %=

Field Name	Type	POST	PUT	Description
displayValue	**String**	Yes	No	The display value for the required attribute.
restriction	**JobRequirementAttributeRestriction**	Yes	No	The restriction of this attribute. May be null, but defaults to JobRequirementAttributeRestriction.must.
value	**String**	Yes	No	The value of the required attribute. During POST, if value is missing, blank, or null, do not provide a comparator.

JobRequirementAttributeRestriction

Represents a restriction for a job requirement attribute.

Value	Description
must	

JobRequirementFeaturesMode

Value	Description
OR	
AND	

NodeAccessPolicy

This enumeration describes how node resources will be shared by various tasks.

Value	Description
NONE	

Value	Description
SHARED	Tasks from any combination of jobs may utilize available resources.
SHAREDONLY	Only jobs requesting shared node access may utilize available resources.
SINGLEJOB	Tasks from a single job may utilize available resourceses.
SINGLETASK	A single task from a single job may run on the node.
SINGLEUSER	Tasks from any jobs owned by the same user may utilize available resources.
UNIQUEUSER	Any number of tasks from a single job may allocate resources from a node but only if the user has no other jobs running on that node.
SINGLEGROUP	Any number of tasks from the same group may utilize node.
SINGLEACCOUNT	Any number of tasks from the same account may utilize node.
SINGLECLASS	Any number of tasks from the same class may utilize node.
SINGLEQOS	Any number of tasks from the same QOS (quality of service) may utilize node.

NodeAllocationPolicy

Node Allocation enumeration.

Value	Description
FIRSTSET	
MINGLOBAL	
MINLOCAL	
PLUGIN	
NONE	No node allocation policy is specified. Moab defaults to MINRESOURCE when this is the case.

Value	Description
FIRSTAVAILABLE	Simple first come, first served algorithm where nodes are allocated in the order they are presented by the resource manager. This is a very simple, very fast algorithm.
LASTAVAILABLE	This algorithm selects resources so as to minimize the amount of time after the job and before the trailing reservation. This algorithm is a best fit in time algorithm which minimizes the impact of reservation based node-time fragmentation. It is useful in systems where a large number of reservations (job, standing, or administrative) are in place.
MINRESOURCE	This algorithm prioritizes nodes according to the configured resources on each node. Those nodes with the fewest configured resources which still meet the job's resource constraints are selected.
CPULOAD	Nodes are selected which have the maximum amount of available, unused cpu power, i.e. [# of CPU's] - [CPU load]. Good algorithm for timesharing node systems. This algorithm is only applied to jobs starting immediately. For the purpose of future reservations, the **MINRESOURCE** algorithm is used.
LOCAL	This will call the locally created **contrib** node allocation algorithm.
CONTIGUOUS	This algorithm will allocate nodes in contiguous (linear) blocks as required by the Compaq RMS system.
MAXBALANCE	This algorithm will attempt to allocate the most 'balanced' set of nodes possible to a job. In most cases, but not all, the metric for balance of the nodes is node speed. Thus, if possible, nodes with identical speeds will be allocated to the job. If identical speed nodes cannot be found, the algorithm will allocate the set of nodes with the minimum node speed 'span' or range.
PRIORITY	This algorithm allows a site to specify the priority of various static and dynamic aspects of compute nodes and allocate them with preference for higher priority nodes. It is highly flexible allowing node attribute and usage information to be combined with reservation affinity.
FASTEST	This algorithm will select nodes in 'fastest node first' order. Nodes will be selected by node speed if specified. If node speed is not specified, nodes will be selected by processor speed. If neither is specified, nodes will be selected in a random order.

Value	Description
PROCESSORLOAD	Alias for CPULOAD.
NODESPEED	Alias for FASTEST.
INREPORTEDORDER	Alias for FIRSTAVAILABLE.
INREVERSEREPORTEDORDER	Alias for LASTAVAILABLE.
CUSTOMPRIORITY	Alias for PRIORITY.
PROCESSORSPEEDBALANCE	Alias for MAXBALANCE.
MINIMUMCONFIGUREDRESOURCES	Alias for MINRESOURCE.
CRAY3DTORUS	Enable topology awareness scheduling algorithm.

AllocatedNode

Field Name	Type	POST	PUT	Description
name	**String**	No	No	
taskCount	**Integer**	No	No	

JobResource

Represents counts of dedicated and utilized resources.

Field Name	Type	POST	PUT	Description
dedicated	**Integer**	No	No	The amount of this resource that has been allocated for running workload.
utilized	**Integer**	No	No	The amount of this resource that is currently reported as utilized by resource managers.

JobResourceFailPolicyType

Value	Description
CANCEL	
FAIL	
HOLD	
IGNORE	
NOTIFY	
REQUEUE	

ResourceManager

Field Name	Type	POST	PUT	Description
isDestination	**Boolean**	No	No	
isSource	**Boolean**	No	No	
jobName	**String**	No	No	
name	**String**	No	No	

JobStateInformation

Field Name	Type	POST	PUT	Description
state	JobState	No	No	
stateExpected	JobState	No	No	
stateLastUpdatedDate	**Date**	No	No	
subState	JobSubState	No	No	

JobState

Value	Description
Idle	
Starting	
Running	
Removed	
Completed	
Hold	
Deferred	
Vacated	
NotQueued	
Unknown	
Staging	
Suspended	
Blocked	

JobSubState

Value	Description
Epilogue	
Migrated	
Preempted	

Value	Description
Prologue	

JobSystemJobType

Value	Description
generic	
osprovision	
osprovision2	
poweroff	
poweron	
reset	
storage	
vmmap	
vmmigrate	
vmtracking	

JobActionType

Value	Description
DESTROY	
MIGRATE	
MODIFY	

VMUsagePolicy

This enumeration describes the virtual machine requirements of a job

Value	Description
REQUIREPM	Requires a physical machine.
PREFPM	Prefers a physical machine.
CREATEVM	Creates a virtual machine.
CREATEPERSISTENTVM	Creates a virtual machine that doesn't go away after the job is done.
REQUIREVM	Requires a virtual machine.
PREFVM	Prefers a virtual machine.

API version 2

Job

This class represents a job in the Moab Workload Manager. A job is a request for compute resources (CPUs, memory, storage) with which the requester can do work for a given amount of time. In an HPC environment, this might be a batch script to perform a Monte Carlo simulation. In a cloud environment, this would be a virtual machine and its associated storage. Moab will evaluate the request and assign the requested resources to the requester based on policies, current demand, and other factors in the data center. A job will also usually have some process that Moab starts automatically at the assigned start time. In an HPC environment, this can be starting a batch script on the assigned nodes. In a cloud environment, this can be starting provisioning processes to create the virtual machine and storage and install software on it.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique identifier of this job. Note: this field is not user-assigned and is generated by the database.
arrayIndex	**Long**	No	No	If this job is a sub-job of a JobArray, this field contains the index of this job in the array. For example, if this job is `Moab.1[2]`, the array index would be `2`.
arrayMasterName	**String**	No	No	If this job is a sub-job of a JobArray, this field contains the name of the job array master. For example, if this job is `Moab.1[2]`, the array master name would be `Moab.1`.
attributes	**Set<String>**	Yes	No	The list of generic attributes associated with this job.
blocks	**Set<JobBlock>**	No	No	Reasons the job is blocked from running.

Field Name	Type	POST	PUT	Description
bypassCount	**Integer**	No	No	The number of times the job has been backfilled.
cancelCount	**Integer**	No	No	The number of times a job has received a cancel request.
commandFile	**String**	Yes	No	The name of the job script file (absolute path). If commandFile is set and commandScript is not set, then MWS must have read access to the file. If commandFile and commandScript are both set, then MWS does not read the contents of the file, but it does provide the name of the file to Moab.
commandLineArguments	**String**	Yes	No	The command line arguments passed to the job script specified by commandFile or commandScript.
commandScript	**String**	Yes	No	The contents of the job script. This field must be Base64-encoded.
completionCode	**Integer**	No	No	The exit code from this job.
cpuTime	**Long**	No	No	CPU usage time in seconds as reported by the resource manager.
credentials	JobCredentials	Yes	Yes	The credentials (user and group, for example) associated with this job.
customName	**String**	Yes	Yes	The user-specified name of this job.

Field Name	Type	POST	PUT	Description
dates	JobDates	Yes	Yes	Various dates associated with this job.
deferCount	**Integer**	No	No	The number of times a job has been deferred.
dependencies	Set<JobDependency>	Yes	No	Dependencies that must be fulfilled before the job can start.
description	**String**	No	No	The description of the job. Can be set only in a job template.
duration	**Long**	Yes	Yes	The length of time in seconds requested for the job. Note that it is possible to set duration to "INFINITY" if the AllowInfiniteJobs flag is set on the scheduler in the moab.cfg.
durationActive	**Long**	No	No	The length of time in seconds the job has been active or running.
durationQueued	**Long**	No	No	The length of time in seconds the job has been eligible to run in the queue.
durationRemaining	**Long**	No	No	An estimate of the time remaining, in seconds, before the job will complete.
durationSuspended	**Long**	No	No	The length of time in seconds the job has been suspended.
emailNotifyAddresses	**Set<String>**	Yes	No	The list of addresses to whom email is sent by the execution server.

Field Name	Type	POST	PUT	Description
emailNotifyTypes	Set<JobEmailNotifyType>	Yes	No	The list of email notify types attached to the job.
environmentRequested	**Boolean**	Yes	No	Setting this field to true tells the Moab Workload Manager to set various variables, if populated, in the job's environment.
environmentVariables	**Map<String, Map>**	Yes	No	The list of environment variables for this job.
epilogScript	**String**	Yes	No	The path to the TORQUE epilog script.
flags	Set<JobFlag>	Yes	Yes	The flags that are set on this job.
holdDate	**Date**	No	No	The date the most recent hold was placed on the job.
holdReason	JobHoldReason	No	No	The reason the job is on hold.
holds	Set<JobHoldType>	Yes	Yes	The holds that are set on the job. The "User" hold type is valid during POST.
initialWorkingDirectory	**String**	Yes	No	The path to the directory in which the job will be started.
isActive	**Boolean**	No	No	True if the job is active, false if the job is complete.
jobGroup	**String**	Yes	No	The job group to which this job belongs (different from credentials.group).

Field Name	Type	POST	PUT	Description
masterNode	DomainProxy	No	No	The first node in the list of allocated nodes for this job. For TORQUE jobs, this represents the "mother superior."
memorySecondsDedicated	**Double**	No	No	The memory seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.
memorySecondsUtilized	**Double**	No	No	The memory seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
messages	Set<Message>	No	Yes	The list of messages associated with the job. The "message" field is valid during PUT.
migrateCount	**Integer**	No	No	The number of times the job has been migrated.
minimumPreemptTime	**Long**	No	No	The minimum length of time, in seconds, an active job must be running before it is eligible for preemption.
mwmName	**String**	No	No	The name of the Moab Workload Manager instance that owns this job.
name	**String**	No	No	The name of this job. This name is unique *per instance* of Moab Workload Manager (i.e. not globally).

Field Name	Type	POST	PUT	Description
nodesExcluded	**Set<DomainProxy>**	Yes	No	The list of nodes that should not be considered for this job.
nodesRequested	**Set<DomainProxy>**	Yes	No	The exact set, superset, or subset of nodes on which this job must run. (See also: nodesRequestedPolicy.)
nodesRequestedPolicy	**JobHostListMode**	Yes	No	Indicates an exact set, superset, or subset of nodes on which the job must run. Only relevant if nodesRequested is provided. (See also: nodesRequested.)
partitionAccessList	**Set<String>**	No	No	The list of partitions that this job can access.
partitionAccessListRequested	**Set<String>**	Yes	Yes	The list of partitions that this job has requested.
preemptCount	**Integer**	No	No	The number of times the job has been preempted.
priorities	**JobPriority**	Yes	Yes	The list of priorities for the job.
processorSecondsDedicated	**Double**	No	No	The processor seconds dedicated to the job as reported by its resource manager. Not all resource managers provide this information.
processorSecondsLimit	**Double**	No	No	The limit for processorSecondsUtilized.

Field Name	Type	POST	PUT	Description
processorSecondsUtilized	**Double**	No	No	The processor seconds utilized by the job as reported by its resource manager. Not all resource managers provide this information.
prologScript	**String**	Yes	No	The path to the TORQUE prolog script.
queueStatus	**JobQueueStatus**	No	No	The status of the job in its queue.
rejectPolicies	**Set<JobRejectPolicy>**	No	No	The list of policies enabled when a job is rejected.
requirements	**Set<JobRequirement>**	Yes	Yes	The list of items required for this job to run. Only JobRequirement.features is valid during PUT.
reservationRequested	**DomainProxy**	Yes	Yes	The reservation that the job requested.
resourceFailPolicy	**JobResourceFailPolicyType**	Yes	No	The policy that dictates what should happen to the job if it is running and at least one of the resources it is using fails.
resourceManagerExtension	**String**	Yes	No	If provided during POST, this string will be added to the resource manager extension section of the job submission. Example: "bandwidth=120;queuejob=false" Note that the delimiter between resourceManagerExtension elements is the semicolon.

Field Name	Type	POST	PUT	Description
resourceManagers	Set<ResourceManager>	No	No	The list of resource managers associated with this job.
rmStandardErrorFilePath	**String**	No	No	The path to the remote file containing the standard error of the job.
rmStandardOutputFilePath	**String**	No	No	The path to the remote file containing the standard output of the job.
shellName	**String**	Yes	No	Declares the shell that interprets the job script.
standardErrorFilePath	**String**	Yes	No	The path to the file containing the standard error of the job.
standardOutputFilePath	**String**	Yes	No	The path to the file containing the standard output of the job.
startCount	**Integer**	No	No	The number of times the job has been started.
states	JobStateInformation	No	No	Information about the state of the job.
submitHost	**String**	Yes	No	The host from which the job was submitted.
systemJobAction	**String**	No	No	The action the system job will take.
systemJobType	JobSystemJobType	No	No	The type of system job. In the Moab Cloud Suite, this will usually be "vmtracking" or "generic."

Field Name	Type	POST	PUT	Description
targetedJobAction	JobActionType	No	No	The action that this job is performing on another job.
targetedJobName	**String**	No	No	The name of the job on which this job is performing the targetedJobAction.
templates	Set<DomainProxy>	Yes	No	The list of all job templates to be set on this job.
triggers	**Set<String>**	No	No	The list of triggers associated with this job.
variables	**Map<String, Map>**	Yes	Yes	The list of variables that this job owns or sets on completion.
virtualContainers	Set<DomainProxy>	Yes	No	When submitting this job, add it to the specified existing virtual container. Valid during POST, but only one virtual container can be specified.
virtualMachines	Set<DomainProxy>	No	No	The list of virtual machines that are allocated to this job.
vmUsagePolicy	VMUsagePolicy	Yes	No	The requested Virtual Machine Usage Policy for this job.

JobBlock

Field Name	Type	POST	PUT	Description
category	JobBlockCategory	No	No	
message	**String**	No	No	
type	JobBlockType	No	No	

JobBlockCategory

Value	Description
depend	
jobBlock	
migrate	

JobBlockType

Value	Description
ActivePolicy	
BadUser	
Dependency	
EState	
FairShare	
Hold	
IdlePolicy	
LocalPolicy	
NoClass	
NoData	
NoResource	
NoTime	
PartitionAccess	

Value	Description
Priority	
RMSubmissionFailure	
StartDate	
State	
SysLimits	

JobCredentials

Moab Workload Manager supports the concept of credentials, which provide a means of attributing policy and resource access to entities such as users and groups. These credentials allow specification of job ownership, tracking of resource usage, enforcement of policies, and many other features.

Field Name	Type	POST	PUT	Description
account	**String**	Yes	Yes	The account credential is also referred to as the project. This credential is generally associated with a group of users along the lines of a particular project for accounting and billing purposes.
group	**String**	Yes	No	The group credential represents an aggregation of users. User-to-group mappings are often specified by the operating system or resource manager and typically map to a user's UNIX group ID. However, user-to-group mappings may also be provided by a security and identity management service, or you can specify such directly within Moab.
jobClass	**String**	Yes	Yes	The concept of the class credential is derived from the resource manager class or queue object. Classes differ from other credentials in that they more directly impact job attributes. In standard HPC usage, a user submits a job to a class and this class imposes a number of factors on the job. The attributes of a class may be specified within the resource manager or directly within Moab.

Field Name	Type	POST	PUT	Description
qos	**String**	No	No	The quality of service assigned to this job. The concept of a quality of service (QoS) credential is unique to Moab and is not derived from any underlying concept or peer service. In most cases, the QoS credential is used to allow a site to set up a selection of service levels for end-users to choose from on a long-term or job-by-job basis. QoS's differ from other credentials in that they are centered around special access where this access may allow use of additional services, additional resources, or improved responsiveness. Unique to this credential, organizations may also choose to apply different charge rates to the varying levels of service available within each QoS. As QoS is an internal credential, all QoS configuration occurs within Moab.
qosRequested	**String**	Yes	Yes	The quality of service requested for this job.
user	**String**	Yes	No	The user credential is the fundamental credential within a workload manager; each job requires an association with exactly one user. In fact, the user credential is the only required credential in Moab; all others are optional. In most cases, the job's user credential is configured within or managed by the operating system itself, although Moab may be configured to obtain this information from an independent security and identity management service.

JobDates

Field Name	Type	POST	PUT	Description
completedDate	**Date**	No	No	
createdDate	**Date**	No	No	
deadlineDate	**Date**	Yes	No	The deadline for completion of the job.
dispatchedDate	**Date**	No	No	
earliestRequestedStartDate	**Date**	Yes	Yes	The job will start no sooner than this date.

Field Name	Type	POST	PUT	Description
earliestStartDate	**Date**	No	No	
eligibleDate	**Date**	No	No	
lastCanceledDate	**Date**	No	No	
lastChargedDate	**Date**	No	No	
lastPreemptedDate	**Date**	No	No	
lastUpdatedDate	**Date**	No	No	
startDate	**Date**	No	No	
submitDate	**Date**	No	No	
terminationDate	**Date**	No	No	

JobDependency

Field Name	Type	POST	PUT	Description
name	**String**	Yes	No	The name of the object on on which the job is dependent.
type	**JobDependencyType**	Yes	No	The type of job dependency. Only set is valid for POST.
value	**String**	No	No	

JobDependencyType

Represents the type of a job dependency. For now, only the "set" type is supported.

Value	Description
set	

JobEmailNotifyType

Value	Description
JobStart	
JobEnd	
JobFail	
All	

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.
ADVRES	The job requires the use of a reservation.
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.

Value	Description
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.
INTERACTIVE	The job needs to interactive input from the user to run.
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.

Value	Description
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.

Value	Description
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

JobHoldReason

Value	Description
Admin	
NoResources	
SystemLimitsExceeded	
BankFailure	
CannotDebitAccount	
InvalidAccount	
RMFailure	
RMReject	
PolicyViolation	
CredAccess	

Value	Description
CredHold	
PreReq	
Data	
Security	
MissingDependency	

JobHoldType

Value	Description
User	
System	
Batch	
Defer	
All	

DomainProxy

A reference to an object contained within an object. For example, a Virtual Machine object contains a reference to the Node on which it is running. That reference is represented by this class.

Field Name	Type	POST	PUT	Description
name	**String**	Yes	No	The name of the object.

Message

Field Name	Type	POST	PUT	Description
count	**Integer**	No	No	The number of times this message has occurred.
createdDate	**Date**	No	No	The date this message was created.
expireDate	**Date**	No	No	The date this message expires.
message	**String**	No	Yes	The message itself.

JobHostListMode

Value	Description
superset	
subset	
exactset	

JobPriority

Field Name	Type	POST	PUT	Description
run	**Long**	No	No	
start	**Long**	No	No	
system	**Long**	No	No	
user	**Long**	Yes	Yes	The user-requested priority for the job. By default, the range is between -1024 and 0. To enable priority range from -1024 to +1023, set `ENABLEPOSUSERPRIORITY` in the `moab.cfg` file.

JobQueueStatus

Value	Description
active	
blocked	
completed	
eligible	

JobRejectPolicy

Value	Description
CANCEL	
HOLD	
IGNORE	
MAIL	
RETRY	

JobRequirement

Field Name	Type	POST	PUT	Description
architecture	**String**	Yes	No	The architecture required by the job.
attributes	**Map<String, JobRequirementAttribute>**	Yes	No	Required node attributes with version number support.

Field Name	Type	POST	PUT	Description
features	**Set<String>**	No	Yes	The list of node features the job is scheduled against.
featuresExcluded	**Set<String>**	Yes	No	Excluded node features. That is, do not select nodes with these features. (See also: featuresExcludedMode.)
featuresExcludedMode	**JobRequirementFeaturesMode**	Yes	No	Indicates whether excluded features should be AND'ed or OR'd. The default is AND. Only relevant if featuresExcluded is provided. (See also: featuresExcluded.)
featuresRequested	**Set<String>**	Yes	No	Requested node features. (See also: featuresRequestedMode.)
featuresRequestedMode	**JobRequirementFeaturesMode**	Yes	No	Indicates whether requested features should be AND'ed or OR'd. The default is AND. Only relevant if featuresRequested is provided. (See also: featuresRequested.)
image	**String**	Yes	No	The image required by the job.
index	**Integer**	No	No	The index of the requirement, starting with 0.
metrics	**Map<String, Double>**	No	No	Generic metrics associated with the job as reported by the resource manager.
nodeAccessPolicy	**NodeAccessPolicy**	Yes	No	Specifies how node resources should be accessed. Note: If the job requirements array has more than one element that contains nodeAccessPolicy, only the first occurrence will be used.

Field Name	Type	POST	PUT	Description
nodeAllocationPolicy	NodeAllocationPolicy	Yes	No	Specifies how node resources should be selected and allocated to the job. Note: If the job requirements array has more than one element that contains nodeAllocationPolicy, only the first occurrence will be used.
nodeCount	**Integer**	Yes	No	The number of nodes required by the job.
nodeSet	**String**	Yes	No	The requested node set of the job. This must follow the format `SETSELECTION:SETTYPE[:SETLIST]` • SETSELECTION - `ANYOF`, `ONEOF`, or `FIRSTOF` • SETTYPE - `FEATURE` or `VARATTR` • SETLIST - For `FEATURE`, a comma-separated list of features. For `VARATTR`, a key=value pair. Examples: • `ONEOF:FEATURE:fastos,hiprio,bigmem` • `FIRSTOF:VARATTR:datacenter=Provo:datacenter=SaltLake`
nodes	Set<AllocatedNode>	No	No	Nodes that have been allocated to meet this requirement.
reservation	DomainProxy	No	No	The allocated reservation (assigned after the job has a reservation).

Field Name	Type	POST	PUT	Description
resourcesPerTask	Map<String, JobResource>	Yes	No	Contains requirements for disk, memory, processors, swap, and generic resources. For disk, memory, and swap, the unit is MB. For each resource, the "dedicated" field can be set during POST.
taskCount	**Integer**	Yes	No	The number of tasks (processors) required by this job.
tasksPerNode	**Integer**	Yes	No	The number of tasks to map to each node.

JobRequirementAttribute

Field Name	Type	POST	PUT	Description
comparator	**String**	Yes	No	The comparison operator. Valid values: • >= - Greater than or equal to • > - Greater than • <= - Less than • < - Less than • %= - Equals • %! - Not equals • Null - Defaults to %= • = - (Deprecated) Equivalent to %=

Field Name	Type	POST	PUT	Description
displayValue	**String**	Yes	No	The display value for the required attribute.
restriction	**JobRequirementAttributeRestriction**	Yes	No	The restriction of this attribute. May be null, but defaults to JobRequirementAttributeRestriction.must.
value	**String**	Yes	No	The value of the required attribute. During POST, if value is missing, blank, or null, do not provide a comparator.

JobRequirementAttributeRestriction

Represents a restriction for a job requirement attribute.

Value	Description
must	

JobRequirementFeaturesMode

Value	Description
OR	
AND	

NodeAccessPolicy

This enumeration describes how node resources will be shared by various tasks.

Value	Description
NONE	

Value	Description
SHARED	Tasks from any combination of jobs may utilize available resources.
SHAREDONLY	Only jobs requesting shared node access may utilize available resources.
SINGLEJOB	Tasks from a single job may utilize available resourceses.
SINGLETASK	A single task from a single job may run on the node.
SINGLEUSER	Tasks from any jobs owned by the same user may utilize available resources.
UNIQUEUSER	Any number of tasks from a single job may allocate resources from a node but only if the user has no other jobs running on that node.
SINGLEGROUP	Any number of tasks from the same group may utilize node.
SINGLEACCOUNT	Any number of tasks from the same account may utilize node.
SINGLECLASS	Any number of tasks from the same class may utilize node.
SINGLEQOS	Any number of tasks from the same QOS (quality of service) may utilize node.

NodeAllocationPolicy

Node Allocation enumeration.

Value	Description
FIRSTSET	
MINGLOBAL	
MINLOCAL	
PLUGIN	
NONE	No node allocation policy is specified. Moab defaults to MINRESOURCE when this is the case.

Value	Description
FIRSTAVAILABLE	Simple first come, first served algorithm where nodes are allocated in the order they are presented by the resource manager. This is a very simple, very fast algorithm.
LASTAVAILABLE	This algorithm selects resources so as to minimize the amount of time after the job and before the trailing reservation. This algorithm is a best fit in time algorithm which minimizes the impact of reservation based node-time fragmentation. It is useful in systems where a large number of reservations (job, standing, or administrative) are in place.
MINRESOURCE	This algorithm prioritizes nodes according to the configured resources on each node. Those nodes with the fewest configured resources which still meet the job's resource constraints are selected.
CPULOAD	Nodes are selected which have the maximum amount of available, unused cpu power, i.e. [# of CPU's] - [CPU load]. Good algorithm for timesharing node systems. This algorithm is only applied to jobs starting immediately. For the purpose of future reservations, the **MINRESOURCE** algorithm is used.
LOCAL	This will call the locally created **contrib** node allocation algorithm.
CONTIGUOUS	This algorithm will allocate nodes in contiguous (linear) blocks as required by the Compaq RMS system.
MAXBALANCE	This algorithm will attempt to allocate the most 'balanced' set of nodes possible to a job. In most cases, but not all, the metric for balance of the nodes is node speed. Thus, if possible, nodes with identical speeds will be allocated to the job. If identical speed nodes cannot be found, the algorithm will allocate the set of nodes with the minimum node speed 'span' or range.
PRIORITY	This algorithm allows a site to specify the priority of various static and dynamic aspects of compute nodes and allocate them with preference for higher priority nodes. It is highly flexible allowing node attribute and usage information to be combined with reservation affinity.
FASTEST	This algorithm will select nodes in 'fastest node first' order. Nodes will be selected by node speed if specified. If node speed is not specified, nodes will be selected by processor speed. If neither is specified, nodes will be selected in a random order.

Value	Description
PROCESSORLOAD	Alias for CPULOAD.
NODESPEED	Alias for FASTEST.
INREPORTEDORDER	Alias for FIRSTAVAILABLE.
INREVERSEREPORTEDORDER	Alias for LASTAVAILABLE.
CUSTOMPRIORITY	Alias for PRIORITY.
PROCESSORSPEEDBALANCE	Alias for MAXBALANCE.
MINIMUMCONFIGUREDRESOURCES	Alias for MINRESOURCE.
CRAY3DTORUS	Enable topology awareness scheduling algorithm.

AllocatedNode

Field Name	Type	POST	PUT	Description
name	**String**	No	No	
taskCount	**Integer**	No	No	

JobResource

Represents counts of dedicated and utilized resources.

Field Name	Type	POST	PUT	Description
dedicated	**Integer**	No	No	The amount of this resource that has been allocated for running workload.
utilized	**Integer**	No	No	The amount of this resource that is currently reported as utilized by resource managers.

JobResourceFailPolicyType

Value	Description
CANCEL	
FAIL	
HOLD	
IGNORE	
NOTIFY	
REQUEUE	

ResourceManager

Field Name	Type	POST	PUT	Description
isDestination	**Boolean**	No	No	
isSource	**Boolean**	No	No	
jobName	**String**	No	No	
name	**String**	No	No	

JobStateInformation

Field Name	Type	POST	PUT	Description
state	**JobState**	No	No	
stateExpected	**JobState**	No	No	
stateLastUpdatedDate	**Date**	No	No	
subState	**JobSubState**	No	No	

JobState

Value	Description
Idle	
Starting	
Running	
Removed	
Completed	
Hold	
Deferred	
Vacated	
NotQueued	
Unknown	
Staging	
Suspended	
Blocked	

JobSubState

Value	Description
Epilogue	
Migrated	
Preempted	

Value	Description
Prologue	

JobSystemJobType

Value	Description
generic	
osprovision	
osprovision2	
poweroff	
poweron	
reset	
storage	
vmmap	
vmmigrate	
vmtracking	

JobActionType

Value	Description
DESTROY	
MIGRATE	
MODIFY	

VMUsagePolicy

This enumeration describes the virtual machine requirements of a job

Value	Description
REQUIREPM	Requires a physical machine.
PREFPM	Prefers a physical machine.
CREATEVM	Creates a virtual machine.
CREATEPERSISTENTVM	Creates a virtual machine that doesn't go away after the job is done.
REQUIREVM	Requires a virtual machine.
PREFVM	Prefers a virtual machine.

Related Topics

- Jobs

Fields: Job Templates

See the associated Job Templates resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`job-templates`	Permissions
Hooks filename	`job-templates.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

JobTemplate

This class represents a job template in the Moab Workload Manager. Job templates are used for two primary purposes: (1) to provide a means of generically matching and categorizing jobs, and (2) to provide a means of setting arbitrary default or forced attributes for certain jobs.

Field Name	Type	Description
id	**String**	The unique identifier for this job template.
account	**String**	The account under which this job will run for billing purposes.
args	**String**	Command-line arguments that get passed to commandFile.
commandFile	**String**	The path to the file that is executed when the job runs. This is the script that will actually call all the work of the job. Can be null.
description	**String**	The description of the job.
durationRequested	**Long**	The amount of time (in seconds) requested for the job.
genericSystemJob	**Boolean**	True if this template will instantiate a generic system job.
inheritResources	**Boolean**	True if jobs instantiated from this template inherit resources.
jobDependencies	**Set<JobTemplateDependency>**	The list of dependencies for this job template.
jobFlags	**Set<JobFlag>**	Job flags for this template.
jobTemplateFlags	**Set<JobTemplateFlag>**	Job template flags for this template.
jobTemplateRequirements	**Set<JobTemplateRequirement>**	The requirements for this job template.

Field Name	Type	Description
priority	**Long**	Relative job priority.
qos	**String**	The Quality of Service for the job.
queue	**String**	The class or queue in which the job will run.
select	**Boolean**	True if job template can be directly requested by job at submission.
trigger	**Trigger**	The trigger that is typically assigned to generic system jobs.
vmUsagePolicy	**VMUsagePolicy**	The virtual machine usage policy.

JobTemplateDependency

Field Name	Type	Description
name	**String**	The name of the template on which this template depends.
type	**JobDependencyTypeVersion1**	The type of the dependency.

JobDependencyTypeVersion1

Value	Description
JOBSTART	Job may start at any time after specified jobs have started execution.
JOBSUCCESSFULCOMPLETE	Job may be start at any time after all specified jobs have successfully completed.
JOBFAILEDCOMPLETE	Job may start at any time after any specified jobs have completed unsuccessfully.

Value	Description
JOBCOMPLETE	Job may start at any time after all specified jobs have completed regardless of completion status.
BEFORE	Job may start at any time before specified jobs have started execution. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
BEFOREANY	Job may start at any time before all specified jobs have completed regardless of completion status. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
BEFOREOK	Job may start at any time before all specified jobs have successfully completed. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
BEFORENOTOK	Job may start at any time before any specified jobs have completed unsuccessfully. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
HIBERNATE	Job was set to Hibernate mode.
SYNCWITH	Job will wait until it can start simultaneously with a master job
SYNCCOUNT	This job will wait until it can start simultaneously with synccount jobs of type syncwith that have all specified this synccount job is their master job.
SET	Job will wait until a variable on a Moab object is set before starting.

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.

Value	Description
ADVRES	The job requires the use of a reservation.
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.

Value	Description
INTERACTIVE	The job needs to interactive input from the user to run.
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).

Value	Description
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

JobTemplateFlag

This enumeration specifies the flag types of a job template.

Value	Description
GLOBALRSVACCESS	
HIDDEN	

Value	Description
HWJOB	
PRIVATE	
SYNCJOBID	
TEMPLATEISDYNAMIC	True if the template is dynamic (not specified via moab.cfg).
SELECT	True if a job can select this template.

JobTemplateRequirement

Field Name	Type	Description
architecture	**String**	The architecture requirement.
diskRequirement	**Integer**	The amount of disk space required (in MB).
genericResources	**Map<String, Integer>**	Consumable generic attributes associated with individual nodes or the special pseudo-node global, which provides shared cluster (floating) consumable resources.
nodeAccessPolicy	**NodeAccessPolicy**	The node access policy. Specifies how node resources will be shared by a job.
operatingSystem	**String**	The operating system requirement.
requiredDiskPerTask	**Integer**	Disk space (in MB).
requiredFeatures	**Set<String>**	The features required by this template.
requiredMemoryPerTask	**Integer**	Memory (in MB).
requiredProcessorsPerTask	**Integer**	Number of processors.
requiredSwapPerTask	**Integer**	Swap space (in MB).

Field Name	Type	Description
taskCount	**Integer**	The number of tasks required.

NodeAccessPolicy

This enumeration describes how node resources will be shared by various tasks.

Value	Description
NONE	
SHARED	Tasks from any combination of jobs may utilize available resources.
SHAREDONLY	Only jobs requesting shared node access may utilize available resources.
SINGLEJOB	Tasks from a single job may utilize available resourceses.
SINGLETASK	A single task from a single job may run on the node.
SINGLEUSER	Tasks from any jobs owned by the same user may utilize available resources.
UNIQUEUSER	Any number of tasks from a single job may allocate resources from a node but only if the user has no other jobs running on that node.
SINGLEGROUP	Any number of tasks from the same group may utilize node.
SINGLEACCOUNT	Any number of tasks from the same account may utilize node.
SINGLECLASS	Any number of tasks from the same class may utilize node.
SINGLEQOS	Any number of tasks from the same QOS (quality of service) may utilize node.

Trigger

Field Name	Type	Description
id	**String**	Trigger id - internal ID used by moab to track triggers
action	**String**	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	
blockTime	**Date**	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	
eventType	**TriggerEventType**	
expireTime	**Date**	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	
interval	**Boolean**	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	Specifies the number of times Action will be attempted before the trigger is designated a failure.
multiFire	**Boolean**	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length

Field Name	Type	Description
objectId	**String**	The ID of the object which this is attached to.
objectType	**String**	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.
sets	**String**	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	The type of the trigger.

Field Name	Type	Description
unsets	**String**	Variable this trigger destroys upon success or failure.

TriggerActionType

This enumeration specifies the action type of a trigger.

Value	Description
CANCEL	Only apply to reservation triggers.
CHANGE_ PARAM	Run changeparam (NOT PERSISTENT).
JOB_ PREEMPT	Indicates that the trigger should preempt all jobs currently allocating resources assigned to the trigger's parent object. Only apply to reservation triggers.
MAIL	Sends an e-mail.
INTERNAL	Modifies an object internally in Moab. This can be used to set a job hold for example.
EXEC	Execute the trigger action. Typically used to run a script.
MODIFY	Can modify object that trigger is attached to.
QUERY	
RESERVE	
SUBMIT	

TriggerEventType

This enumeration specifies the event type of a trigger.

Value	Description
CANCEL	

Value	Description
CHECKPOINT	
CREATE	
END	
EPOCH	
FAIL	
HOLD	
MIGRATE	
MODIFY	
PREEMPT	
STANDING	
START	
THRESHOLD	
DISCOVER	
LOGROLL	

TriggerFlag

This enumeration specifies a flag belonging to a trigger.

Value	Description
ATTACH_ERROR	If the trigger outputs anything to stderr, Moab will attach this as a message to the trigger object.
CLEANUP	If the trigger is still running when the parent object completes or is canceled, the trigger will be killed.

Value	Description
CHECKPOINT	Moab should always checkpoint this trigger. See Checkpointing a Trigger in the Moab Workload Manager documentation for more information.
GLOBAL_VARS	The trigger will look in the name space of all nodes with the globalvars flag in addition to its own name space. A specific node to search can be specified using the following format: globalvars+node_id
INTERVAL	Trigger is periodic.
MULTIFIRE	Trigger can fire multiple times.
OBJECT_XML_ STDIN	Trigger passes its parent's object XML information into the trigger's stdin. This only works for exec triggers with reservation type parents.
USER	The trigger will execute under the user ID of the object's owner. If the parent object is sched, the user to run under may be explicitly specified using the format user+<username>, for example flags=user+john:
GLOBAL_ TRIGGER	The trigger will be (or was) inserted into the global trigger list.
ASYNCHRONOUS	An asynchronous trigger.
LEAVE_FILES	Do not remove stderr and stdout files.
PROBE	The trigger's stdout will be monitored.
PROBE_ALL	The trigger's stdout will be monitored.
GENERIC_ SYSTEM_JOB	The trigger belongs to a generic system job (for checkpointing).
REMOVE_STD_ FILES	The trigger will delete stdout/stderr files after it has been reset.
RESET_ON_ MODIFY	The trigger resets if the object it is attached to is modified, even if multifire is not set.

Value	Description
SOFT_KILL	By default, a `SIGKILL` (kill -9) signal is sent to the kill the script when a trigger times out. This flag will instead send a `SIGTERM` (kill -15) signal to kill the script. The `SIGTERM` signal will allow the script to trap the signal so that the script can clean up any residual information on the system (instead of just dying, as with the `SIGKILL` signal). NOTE: A timed-out trigger will only receive one kill signal. This means that if you specify this flag, a timed-out trigger will only receive the `SIGTERM` signal, and never the `SIGKILL` signal.

TriggerPeriod

This enumeration specifies the period of a trigger.

Value	Description
MINUTE	
HOUR	
DAY	
WEEK	
MONTH	

TriggerType

This enumeration specifies the type of the trigger.

Value	Description
generic	
elastic	

VMUsagePolicy

This enumeration describes the virtual machine requirements of a job

Value	Description
REQUIREPM	Requires a physical machine.
PREFPM	Prefers a physical machine.
CREATEVM	Creates a virtual machine.
CREATEPERSISTENTVM	Creates a virtual machine that doesn't go away after the job is done.
REQUIREVM	Requires a virtual machine.
PREFVM	Prefers a virtual machine.

API version 2

JobTemplate

This class represents a job template in the Moab Workload Manager. Job templates are used for two primary purposes: (1) to provide a means of generically matching and categorizing jobs, and (2) to provide a means of setting arbitrary default or forced attributes for certain jobs.

Field Name	Type	Description
id	**String**	The unique identifier for this job template.
account	**String**	The account under which this job will run for billing purposes.
args	**String**	Command-line arguments that get passed to commandFile.
commandFile	**String**	The path to the file that is executed when the job runs. This is the script that will actually call all the work of the job. Can be null.
description	**String**	The description of the job.
durationRequested	**Long**	The amount of time (in seconds) requested for the job.
genericSystemJob	**Boolean**	True if this template will instantiate a generic system job.
inheritResources	**Boolean**	True if jobs instantiated from this template inherit resources.
jobDependencies	**Set<JobTemplateDependency>**	The list of dependencies for this job template.
jobFlags	**Set<JobFlag>**	Job flags for this template.
jobTemplateFlags	**Set<JobTemplateFlag>**	Job template flags for this template.
jobTemplateRequirements	**Set<JobTemplateRequirement>**	The requirements for this job template.

Field Name	Type	Description
priority	**Long**	Relative job priority.
qos	**String**	The Quality of Service for the job.
queue	**String**	The class or queue in which the job will run.
select	**Boolean**	True if job template can be directly requested by job at submission.
trigger	**Trigger**	The trigger that is typically assigned to generic system jobs.
vmUsagePolicy	**VMUsagePolicy**	The virtual machine usage policy.

JobTemplateDependency

Field Name	Type	Description
name	**String**	The name of the template on which this template depends.
type	**JobDependencyTypeVersion1**	The type of the dependency.

JobDependencyTypeVersion1

Value	Description
JOBSTART	Job may start at any time after specified jobs have started execution.
JOBSUCCESSFULCOMPLETE	Job may be start at any time after all specified jobs have successfully completed.
JOBFAILEDCOMPLETE	Job may start at any time after any specified jobs have completed unsuccessfully.

Value	Description
JOBCOMPLETE	Job may start at any time after all specified jobs have completed regardless of completion status.
BEFORE	Job may start at any time before specified jobs have started execution. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
BEFOREANY	Job may start at any time before all specified jobs have completed regardless of completion status. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
BEFOREOK	Job may start at any time before all specified jobs have successfully completed. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
BEFORENOTOK	Job may start at any time before any specified jobs have completed unsuccessfully. NOTE: Only reported to Moab and then reported back. Moab currently cannot internally handle this type of dependency.
HIBERNATE	Job was set to Hibernate mode.
SYNCWITH	Job will wait until it can start simultaneously with a master job
SYNCCOUNT	This job will wait until it can start simultaneously with synccount jobs of type syncwith that have all specified this synccount job is their master job.
SET	Job will wait until a variable on a Moab object is set before starting.

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.

Value	Description
ADVRES	The job requires the use of a reservation.
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.

Value	Description
INTERACTIVE	The job needs to interactive input from the user to run.
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).

Value	Description
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

JobTemplateFlag

This enumeration specifies the flag types of a job template.

Value	Description
GLOBALRSVACCESS	
HIDDEN	

Value	Description
HWJOB	
PRIVATE	
SYNCJOBID	
TEMPLATEISDYNAMIC	True if the template is dynamic (not specified via moab.cfg).
SELECT	True if a job can select this template.

JobTemplateRequirement

Field Name	Type	Description
architecture	**String**	The architecture requirement.
diskRequirement	**Integer**	The amount of disk space required (in MB).
genericResources	**Map<String, Integer>**	Consumable generic attributes associated with individual nodes or the special pseudo-node global, which provides shared cluster (floating) consumable resources.
nodeAccessPolicy	NodeAccessPolicy	The node access policy. Specifies how node resources will be shared by a job.
operatingSystem	**String**	The operating system requirement.
requiredDiskPerTask	**Integer**	Disk space (in MB).
requiredFeatures	**Set<String>**	The features required by this template.
requiredMemoryPerTask	**Integer**	Memory (in MB).
requiredProcessorsPerTask	**Integer**	Number of processors.
requiredSwapPerTask	**Integer**	Swap space (in MB).

Field Name	Type	Description
taskCount	**Integer**	The number of tasks required.

NodeAccessPolicy

This enumeration describes how node resources will be shared by various tasks.

Value	Description
NONE	
SHARED	Tasks from any combination of jobs may utilize available resources.
SHAREDONLY	Only jobs requesting shared node access may utilize available resources.
SINGLEJOB	Tasks from a single job may utilize available resourceses.
SINGLETASK	A single task from a single job may run on the node.
SINGLEUSER	Tasks from any jobs owned by the same user may utilize available resources.
UNIQUEUSER	Any number of tasks from a single job may allocate resources from a node but only if the user has no other jobs running on that node.
SINGLEGROUP	Any number of tasks from the same group may utilize node.
SINGLEACCOUNT	Any number of tasks from the same account may utilize node.
SINGLECLASS	Any number of tasks from the same class may utilize node.
SINGLEQOS	Any number of tasks from the same QOS (quality of service) may utilize node.

Trigger

Field Name	Type	Description
id	**String**	Trigger id - internal ID used by moab to track triggers
action	**String**	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	
blockTime	**Date**	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	
eventType	**TriggerEventType**	
expireTime	**Date**	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	
interval	**Boolean**	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	Specifies the number of times Action will be attempted before the trigger is designated a failure.
multiFire	**Boolean**	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length

Field Name	Type	Description
objectId	**String**	The ID of the object which this is attached to.
objectType	**String**	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.
sets	**String**	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	The type of the trigger.

Field Name	Type	Description
unsets	**String**	Variable this trigger destroys upon success or failure.

TriggerActionType

This enumeration specifies the action type of a trigger.

Value	Description
CANCEL	Only apply to reservation triggers.
CHANGE_ PARAM	Run changeparam (NOT PERSISTENT).
JOB_ PREEMPT	Indicates that the trigger should preempt all jobs currently allocating resources assigned to the trigger's parent object. Only apply to reservation triggers.
MAIL	Sends an e-mail.
INTERNAL	Modifies an object internally in Moab. This can be used to set a job hold for example.
EXEC	Execute the trigger action. Typically used to run a script.
MODIFY	Can modify object that trigger is attached to.
QUERY	
RESERVE	
SUBMIT	

TriggerEventType

This enumeration specifies the event type of a trigger.

Value	Description
CANCEL	

Value	Description
CHECKPOINT	
CREATE	
END	
EPOCH	
FAIL	
HOLD	
MIGRATE	
MODIFY	
PREEMPT	
STANDING	
START	
THRESHOLD	
DISCOVER	
LOGROLL	

TriggerFlag

This enumeration specifies a flag belonging to a trigger.

Value	Description
ATTACH_ERROR	If the trigger outputs anything to stderr, Moab will attach this as a message to the trigger object.
CLEANUP	If the trigger is still running when the parent object completes or is canceled, the trigger will be killed.

Value	Description
CHECKPOINT	Moab should always checkpoint this trigger. See Checkpointing a Trigger in the Moab Workload Manager documentation for more information.
GLOBAL_VARS	The trigger will look in the name space of all nodes with the globalvars flag in addition to its own name space. A specific node to search can be specified using the following format: globalvars+node_id
INTERVAL	Trigger is periodic.
MULTIFIRE	Trigger can fire multiple times.
OBJECT_XML_ STDIN	Trigger passes its parent's object XML information into the trigger's stdin. This only works for exec triggers with reservation type parents.
USER	The trigger will execute under the user ID of the object's owner. If the parent object is sched, the user to run under may be explicitly specified using the format user+<username>, for example flags=user+john:
GLOBAL_ TRIGGER	The trigger will be (or was) inserted into the global trigger list.
ASYNCHRONOUS	An asynchronous trigger.
LEAVE_FILES	Do not remove stderr and stdout files.
PROBE	The trigger's stdout will be monitored.
PROBE_ALL	The trigger's stdout will be monitored.
GENERIC_ SYSTEM_JOB	The trigger belongs to a generic system job (for checkpointing).
REMOVE_STD_ FILES	The trigger will delete stdout/stderr files after it has been reset.
RESET_ON_ MODIFY	The trigger resets if the object it is attached to is modified, even if multifire is not set.

Value	Description
SOFT_KILL	By default, a `SIGKILL` (kill -9) signal is sent to the kill the script when a trigger times out. This flag will instead send a `SIGTERM` (kill -15) signal to kill the script. The `SIGTERM` signal will allow the script to trap the signal so that the script can clean up any residual information on the system (instead of just dying, as with the `SIGKILL` signal). NOTE: A timed-out trigger will only receive one kill signal. This means that if you specify this flag, a timed-out trigger will only receive the `SIGTERM` signal, and never the `SIGKILL` signal.

TriggerPeriod

This enumeration specifies the period of a trigger.

Value	Description
MINUTE	
HOUR	
DAY	
WEEK	
MONTH	

TriggerType

This enumeration specifies the type of the trigger.

Value	Description
generic	
elastic	

VMUsagePolicy

This enumeration describes the virtual machine requirements of a job

Value	Description
REQUIREPM	Requires a physical machine.
PREFPM	Prefers a physical machine.
CREATEVM	Creates a virtual machine.
CREATEPERSISTENTVM	Creates a virtual machine that doesn't go away after the job is done.
REQUIREVM	Requires a virtual machine.
PREFVM	Prefers a virtual machine.

Related Topics

- Job Templates

Fields: Metric Types

See the associated Metric Types resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`metric-types`	Permissions
Hooks filename	`metric-types.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

MetricType

Represents a metric visible and known to Moab Workload Manager.

Field Name	Type	Description
id	**String**	The unique ID of this metric type.

API version 2

MetricType

Represents a metric visible and known to Moab Workload Manager.

Field Name	Type	Description
id	**String**	The unique ID of this metric type.

Related Topics

- Metric Types

Fields: Nodes

See the associated Nodes resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`nodes`	Permissions
Hooks filename	`nodes.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Node

This class represents a node in the Moab Workload Manager. Moab recognizes a node as a collection of resources with a particular set of associated attributes. This definition is similar to the traditional notion of a node found in a Linux cluster or supercomputer wherein a node is defined as one or more CPUs, associated memory, and possibly other compute resources such as local disk, swap, network adapters, and software licenses. Additionally, this node is described by various attributes such as an architecture type or operating system. Nodes range in size from small uniprocessor PCs to large symmetric multiprocessing (SMP) systems where a single node may consist of hundreds of CPUs and massive amounts of memory.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier of this node. Note: this field is not user-assigned and is generated by the database.
aclRules	**Set<NodeAclRule>**	No	The set of access control rules associated with this node.
architecture	**String**	No	This node's processor architecture.
attributes	**Map<String, Map>**	No	Attributes is a map of attribute names to tuples (maps) that describe the scheduling attributes of a node. Each tuple should contain the following entries: • **value** - the attribute value • **displayValue** - the attribute display value
classes	**Set<String>**	No	The classes that this node can be scheduled for.

Field Name	Type	PUT	Description
featuresCustom	**Set<String>**	Yes	The features this node advertises which are customizable at run-time. This can be used to define node sets. (See also: featuresReported.)
featuresReported	**Set<String>**	No	The features this node advertises which are reported by resource managers or are present in the Moab Workload Manager configuration. This can be used to define node sets. (See also: featuresCustom.)
index	**Integer**	No	The index for this node as reported by the resource manager.
ipAddress	**String**	No	This node's IPv4 address.
isHypervisor	**Boolean**	No	True if the node is a hypervisor, false otherwise. This is based on the NodeOperatingSystemInformation.hypervisorType field. If hypervisorType is present, the node is a hypervisor. If it is null, then it is not a hypervisor.
jobs	**Set<DomainProxy>**	No	Jobs associated with this node.
lastUpdatedDate	**Date**	No	The timestamp of the last moment when this node was updated. There is no guarantee that all user modifications to a node would be picked up. This will also be changed every RMPOLLINTERVAL even if a resource manager does not report information on this node.
messages	**Set<Message>**	Yes	The list of messages attached to this node. They can be attached by admins, the resource manager layer, or triggers.

Field Name	Type	PUT	Description
metrics	**Map<String, Double>**	Yes	Metrics are the measurable, quantitative, and changing aspects of this node. They are used to define workload placement, attach triggers, etc. There are some built-in metrics: • **speed** - A number from 0.0 to 1.0 describing the relative speed of the system for computational tasks. This is a composite metric, and is defined on a per-site basis. • **cpuLoad** - This is the CPU load on this node. This value is defined at the resource manager layer, but is generally defined on a per-operating system basis. For example, Unix-based OS's use some aspect of the Unix load average, as reported by the resource manager layer, while Windows-based OS's use CPU utilization.
migrationDisabled	**Boolean**	No	True if VM migration is disabled on this node.
name	**String**	No	The name of this node. This name is unique *per instance* of Moab Workload Manager (i.e. not globally).
operatingSystem	NodeOperatingSystemInformation	Yes	Describes the current or expected operating system image information for this node. The operatingSystem.image field can be changed using PUT.
partition	**String**	Yes	The partition this node belongs to.
processorSpeed	**Integer**	No	The speed, in MHz, or the processors on this node.

Field Name	Type	PUT	Description
profilingEnabled	**Boolean**	No	Indicates whether historical data gathering and reporting is enabled for this node. This is also controlled by the same setting on the default node (i.e. all nodes). If set to false (default), node statistics are not gathered.
rack	**Integer**	No	The rack where this node is located in the datacenter/cluster.
requestId	**String**	No	An ID that can be used to track the request that created the node.
reservations	**Set<DomainProxy>**	No	Reservations associated with this node.
resourceManagerMessages	**Map<String, Map>**	No	The resource manager messages for this node. Each key is the name of a resource manager, and the value is the message that the resource manager has posted onto the node.
resourceManagers	**Set<NodeResourceManager>**	No	The resource managers that are reporting or have previously reported this node. Each object also contains information on the resource manager reports.
resources	**Map<String, Resource>**	No	Contains references of a string representing a resource name to a resource object detailing the amount of the resource that is available, configured, etc. Each key is the name of the resource, which equates to the generic resource identifier or one of "processors", "memory", "disk", or "swap". This name may be used as an id in the resource types web service.
slot	**Integer**	No	The slot in the rack where this node is located.

Field Name	Type	PUT	Description
states	NodeStateInformation	Yes	This node's state. The states.powerState and states.state fields can be changed using PUT.
timeToLive	**Date**	No	Specifies the time that the node is supposed to be retired by Moab. Moab will not schedule any jobs on a node after its time to live has passed.
triggers	Set<DomainProxy>	No	Triggers associated with this node.
type	NodeType	No	The type of this node is governed by the types of resources it offers.
variables	**Map<String, Map>**	Yes	Variables is a map of key-value pairs, synonymous, but not directly related to, environment variables. They provide the mechanism to store arbitrary metdata which is useful to external systems in memory on this node.
virtualContainers	Set<DomainProxy>	No	The set of virtual containers that directly (not recursively) contain this node.
virtualMachines	Set<DomainProxy>	No	Virtual machines associated with this node.

NodeAclRule

This class represents a rule that can be in Moab's access control list (ACL) for a node.

The basic NodeAclRule information is the object's name and type. The type directly maps to an NodeAclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	PUT	Description
affinity	AclAffinity	Yes	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	ComparisonOperator	Yes	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
credentialLock	**Boolean**	No	Matching jobs will be required to run on the resources reserved by this reservation. You can use this modifier on accounts, classes, groups, qualities of service, and users.
excludeFromAcl	**Boolean**	No	If attribute is met, the requestor is denied access regardless of any other satisfied ACLs.
hardPolicyOnly	**Boolean**	No	ACLs marked with this modifier are ignored during soft policy scheduling and are only considered for hard policy scheduling once all eligible soft policy jobs start.
requireAll	**Boolean**	No	All required ACLs must be satisfied for requestor access to be granted.
type	NodeAclType	Yes	The type of the object that is being granted (or denied) access.

Field Name	Type	PUT	Description
value	**String**	Yes	The name of the object that is being granted (or denied) access.
xorWithAcl	**Boolean**	No	All attributes of the type specified other than the ones listed in the ACL satisfy the ACL.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"

Value	Description
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

NodeAclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template

Value	Description
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
TASKSPERNODE	Not supported
PROC	Not supported
QTIME	Not supported
QUEUE	Not supported
RACK	Not supported
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

DomainProxy

A reference to an object contained within an object. For example, a Virtual Machine object contains a reference to the Node on which it is running. That reference is represented by this class.

Field Name	Type	PUT	Description
name	**String**	No	The name of the object.

Message

Field Name	Type	PUT	Description
count	**Integer**	No	The number of times this message has occurred.
createdDate	**Date**	No	The date this message was created.
expireDate	**Date**	No	The date this message expires.
message	**String**	Yes	The message itself.

NodeOperatingSystemInformation

Describes the current or expected operating system image information for a node.

Field Name	Type	PUT	Description
hypervisorType	**String**	No	The hypervisor technology that this node uses. May be null if the node is not a hypervisor.
image	**String**	Yes	The name of the operating system currently running on this node. In cloud mode, this corresponds to the ID or name of an image in the image management API in MWS. (See also: Image.id, Image.name.)
imageExpected	**String**	No	The name of the image that was requested to run on this node (i.e. with `mnodectl -m os=myOs`). In cloud mode, this corresponds to the ID or name of an image in the image management API in MWS. (See also: Image.id, Image.name.)
imageLastUpdatedDate	**Date**	No	The last time the image of this node was modified.

Field Name	Type	PUT	Description
imagesAvailable	**Set<String>**	No	The list of image names that can be applied to this node. In cloud mode, this corresponds to IDs or names of images in the image management API in MWS. (See also: Image.id, Image.name.)
virtualMachineImages	**Set<String>**	No	The list of virtual machine image names the node is capable of supporting. In cloud mode, this corresponds to IDs or names of images in the image management API in MWS. (See also: Image.id, Image.name.)

NodeResourceManager

Field Name	Type	PUT	Description
isMaster	**Boolean**	No	Indicates whether this resource manager is the "master" of this Node. If true, it means that this resource manager has the final say on all properties reported about this Node. Note that the first resource manager to report a node is the master resource manager.
name	**String**	No	The name of the resource manager, according to Moab. This name appears in both the RMCFG parameter, and when diagnosing resource managers (e.g. `mdiag -R`).
stateReported	**NodeState**	No	The state reported by this resource manager. See the State section for more details.

NodeState

This enumeration tracks the state of a node.

Value	Description
NONE	The node is set to none by the resource manager.
DOWN	The node is not available for workload.
IDLE	The node is available for workload but is not running anything.

Value	Description
BUSY	The node is running workload and cannot accept more.
RUNNING	The node is running workload and can accept more.
DRAINED	The node has been sent the drain request and has no workload on it.
DRAINING	The node has been sent the drain request, but still has workload on it.
FLUSH	The node is being reprovisioned.
RESERVED	The node is being reserved. This is an internal Moab state.
UNKNOWN	The state of the node is unknown.

Resource

Represents counts of resources available, configured, etc.

Field Name	Type	PUT	Description
available	**Integer**	No	The amount of this resource that is currently available for allocation to workload.
configured	**Integer**	No	The amount of this resource that is considered possible to schedule. Overcommit specifically applies to this, in other words, `configured = overcommitFactor * real`.
dedicated	**Integer**	No	The amount of this resource that has been allocated for running workload. When used in a job submission, this number is the amount of the resource required by the job.
real	**Integer**	No	The amount of this resource that physically exists on the node. Overcommit specifically doesn't apply to this. Note that overcommit currently only applies to "processors" and "memory", and so, for most cases, real and configured will always be the same.
utilized	**Integer**	No	The amount of this resource that is currently reported as utilized by resource managers.

NodeStateInformation

Field Name	Type	PUT	Description
powerState	**NodePower**	Yes	The state of the node's power system, as reported by the RM layer. Modifying the powerState is possible, and, if Moab is configured properly, a request will be made to modify the power state accordingly.
powerStateExpected	**NodePower**	No	The expected state of the node's power system. If a user has requested that a node be powered off (e.g. by modifying the powerState attribute to NodePower.OFF), the requested state will be shown in this field until the state change is completed. If there is no pending power change request, this will be null.
state	**NodeState**	Yes	The scheduling state of the Node, as reported by the resource management layer.
stateExpected	**NodeState**	No	The scheduling state of the Node, as expected by Moab. For example, Moab may think that a Node is "Busy" because it has allocated all configured resources, but a resource manager may report the state as "Running" based on actual utilization of the resources.
stateLastUpdatedDate	**Date**	No	A timestamp recording when the state of the Node was last modified.
subState	**String**	No	A text description of the state of the Node, with the intention of giving more details. Resource Managers may use this field to further describe the state being reported. Resource Managers should provide documented meaning to the possible sub-states that they can report.
subStateLast	**String**	No	The previous sub-state of the Node as reported by the resource management layer.
subStateLastUpdatedDate	**Date**	No	A timestamp recording when the sub-state was last modified.

NodePower

Represents the various options for a Node's power state.

Value	Description
NONE	
ON	
OFF	

NodeType

Represents the type of node as reported by a resource manager.

Value	Description
Compute	
License	
Network	
Storage	

API version 2

Node

This class represents a node in the Moab Workload Manager. Moab recognizes a node as a collection of resources with a particular set of associated attributes. This definition is similar to the traditional notion of a node found in a Linux cluster or supercomputer wherein a node is defined as one or more CPUs, associated memory, and possibly other compute resources such as local disk, swap, network adapters, and software licenses. Additionally, this node is described by various attributes such as an architecture type or operating system. Nodes range in size from small uniprocessor PCs to large symmetric multiprocessing (SMP) systems where a single node may consist of hundreds of CPUs and massive amounts of memory.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier of this node. Note: this field is not user-assigned and is generated by the database.
aclRules	**Set<NodeAclRule>**	No	The set of access control rules associated with this node.
architecture	**String**	No	This node's processor architecture.
attributes	**Map<String, Map>**	No	Attributes is a map of attribute names to tuples (maps) that describe the scheduling attributes of a node. Each tuple should contain the following entries: • **value** - the attribute value • **displayValue** - the attribute display value
classes	**Set<String>**	No	The classes that this node can be scheduled for.

Field Name	Type	PUT	Description
featuresCustom	**Set<String>**	Yes	The features this node advertises which are customizable at run-time. This can be used to define node sets. (See also: featuresReported.)
featuresReported	**Set<String>**	No	The features this node advertises which are reported by resource managers or are present in the Moab Workload Manager configuration. This can be used to define node sets. (See also: featuresCustom.)
index	**Integer**	No	The index for this node as reported by the resource manager.
ipAddress	**String**	No	This node's IPv4 address.
isHypervisor	**Boolean**	No	True if the node is a hypervisor, false otherwise. This is based on the NodeOperatingSystemInformation.hypervisorType field. If hypervisorType is present, the node is a hypervisor. If it is null, then it is not a hypervisor.
jobs	**Set<DomainProxy>**	No	Jobs associated with this node.
lastUpdatedDate	**Date**	No	The timestamp of the last moment when this node was updated. There is no guarantee that all user modifications to a node would be picked up. This will also be changed every RMPOLLINTERVAL even if a resource manager does not report information on this node.
messages	**Set<Message>**	Yes	The list of messages attached to this node. They can be attached by admins, the resource manager layer, or triggers.

Field Name	Type	PUT	Description
metrics	**Map<String, Double>**	Yes	Metrics are the measurable, quantitative, and changing aspects of this node. They are used to define workload placement, attach triggers, etc. There are some built-in metrics: • **speed** - A number from 0.0 to 1.0 describing the relative speed of the system for computational tasks. This is a composite metric, and is defined on a per-site basis. • **cpuLoad** - This is the CPU load on this node. This value is defined at the resource manager layer, but is generally defined on a per-operating system basis. For example, Unix-based OS's use some aspect of the Unix load average, as reported by the resource manager layer, while Windows-based OS's use CPU utilization.
migrationDisabled	**Boolean**	No	True if VM migration is disabled on this node.
name	**String**	No	The name of this node. This name is unique *per instance* of Moab Workload Manager (i.e. not globally).
operatingSystem	NodeOperatingSystemInformation	Yes	Describes the current or expected operating system image information for this node. The operatingSystem.image field can be changed using PUT.
partition	**String**	Yes	The partition this node belongs to.
processorSpeed	**Integer**	No	The speed, in MHz, or the processors on this node.

Field Name	Type	PUT	Description
profilingEnabled	**Boolean**	No	Indicates whether historical data gathering and reporting is enabled for this node. This is also controlled by the same setting on the default node (i.e. all nodes). If set to false (default), node statistics are not gathered.
rack	**Integer**	No	The rack where this node is located in the datacenter/cluster.
requestId	**String**	No	An ID that can be used to track the request that created the node.
reservations	**Set<DomainProxy>**	No	Reservations associated with this node.
resourceManagerMessages	**Map<String, Map>**	No	The resource manager messages for this node. Each key is the name of a resource manager, and the value is the message that the resource manager has posted onto the node.
resourceManagers	**Set<NodeResourceManager>**	No	The resource managers that are reporting or have previously reported this node. Each object also contains information on the resource manager reports.
resources	**Map<String, Resource>**	No	Contains references of a string representing a resource name to a resource object detailing the amount of the resource that is available, configured, etc. Each key is the name of the resource, which equates to the generic resource identifier or one of "processors", "memory", "disk", or "swap". This name may be used as an id in the resource types web service.
slot	**Integer**	No	The slot in the rack where this node is located.

Field Name	Type	PUT	Description
states	NodeStateInformation	Yes	This node's state. The states.powerState and states.state fields can be changed using PUT.
timeToLive	**Date**	No	Specifies the time that the node is supposed to be retired by Moab. Moab will not schedule any jobs on a node after its time to live has passed.
triggers	Set<DomainProxy>	No	Triggers associated with this node.
type	NodeType	No	The type of this node is governed by the types of resources it offers.
variables	**Map<String, Map>**	Yes	Variables is a map of key-value pairs, synonymous, but not directly related to, environment variables. They provide the mechanism to store arbitrary metdata which is useful to external systems in memory on this node.
virtualContainers	Set<DomainProxy>	No	The set of virtual containers that directly (not recursively) contain this node.
virtualMachines	Set<DomainProxy>	No	Virtual machines associated with this node.

NodeAclRule

This class represents a rule that can be in Moab's access control list (ACL) for a node.

The basic NodeAclRule information is the object's name and type. The type directly maps to an NodeAclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	PUT	Description
affinity	**AclAffinity**	Yes	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	**ComparisonOperator**	Yes	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
credentialLock	**Boolean**	No	Matching jobs will be required to run on the resources reserved by this reservation. You can use this modifier on accounts, classes, groups, qualities of service, and users.
excludeFromAcl	**Boolean**	No	If attribute is met, the requestor is denied access regardless of any other satisfied ACLs.
hardPolicyOnly	**Boolean**	No	ACLs marked with this modifier are ignored during soft policy scheduling and are only considered for hard policy scheduling once all eligible soft policy jobs start.
requireAll	**Boolean**	No	All required ACLs must be satisfied for requestor access to be granted.
type	**NodeAclType**	Yes	The type of the object that is being granted (or denied) access.

Field Name	Type	PUT	Description
value	**String**	Yes	The name of the object that is being granted (or denied) access.
xorWithAcl	**Boolean**	No	All attributes of the type specified other than the ones listed in the ACL satisfy the ACL.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"

Value	Description
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

NodeAclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template

Value	Description
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
TASKSPERNODE	Not supported
PROC	Not supported
QTIME	Not supported
QUEUE	Not supported
RACK	Not supported
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

DomainProxy

A reference to an object contained within an object. For example, a Virtual Machine object contains a reference to the Node on which it is running. That reference is represented by this class.

Field Name	Type	PUT	Description
name	**String**	No	The name of the object.

Message

Field Name	Type	PUT	Description
count	**Integer**	No	The number of times this message has occurred.
createdDate	**Date**	No	The date this message was created.
expireDate	**Date**	No	The date this message expires.
message	**String**	Yes	The message itself.

NodeOperatingSystemInformation

Describes the current or expected operating system image information for a node.

Field Name	Type	PUT	Description
hypervisorType	**String**	No	The hypervisor technology that this node uses. May be null if the node is not a hypervisor.
image	**String**	Yes	The name of the operating system currently running on this node. In cloud mode, this corresponds to the ID or name of an image in the image management API in MWS. (See also: Image.id, Image.name.)
imageExpected	**String**	No	The name of the image that was requested to run on this node (i.e. with `mnodectl -m os=myOs`). In cloud mode, this corresponds to the ID or name of an image in the image management API in MWS. (See also: Image.id, Image.name.)
imageLastUpdatedDate	**Date**	No	The last time the image of this node was modified.

Field Name	Type	PUT	Description
imagesAvailable	**Set<String>**	No	The list of image names that can be applied to this node. In cloud mode, this corresponds to IDs or names of images in the image management API in MWS. (See also: Image.id, Image.name.)
virtualMachineImages	**Set<String>**	No	The list of virtual machine image names the node is capable of supporting. In cloud mode, this corresponds to IDs or names of images in the image management API in MWS. (See also: Image.id, Image.name.)

NodeResourceManager

Field Name	Type	PUT	Description
isMaster	**Boolean**	No	Indicates whether this resource manager is the "master" of this Node. If true, it means that this resource manager has the final say on all properties reported about this Node. Note that the first resource manager to report a node is the master resource manager.
name	**String**	No	The name of the resource manager, according to Moab. This name appears in both the RMCFG parameter, and when diagnosing resource managers (e.g. `mdiag -R`).
stateReported	**NodeState**	No	The state reported by this resource manager. See the State section for more details.

NodeState

This enumeration tracks the state of a node.

Value	Description
NONE	The node is set to none by the resource manager.
DOWN	The node is not available for workload.
IDLE	The node is available for workload but is not running anything.

Value	Description
BUSY	The node is running workload and cannot accept more.
RUNNING	The node is running workload and can accept more.
DRAINED	The node has been sent the drain request and has no workload on it.
DRAINING	The node has been sent the drain request, but still has workload on it.
FLUSH	The node is being reprovisioned.
RESERVED	The node is being reserved. This is an internal Moab state.
UNKNOWN	The state of the node is unknown.

Resource

Represents counts of resources available, configured, etc.

Field Name	Type	PUT	Description
available	**Integer**	No	The amount of this resource that is currently available for allocation to workload.
configured	**Integer**	No	The amount of this resource that is considered possible to schedule. Overcommit specifically applies to this, in other words, `configured = overcommitFactor * real`.
dedicated	**Integer**	No	The amount of this resource that has been allocated for running workload. When used in a job submission, this number is the amount of the resource required by the job.
real	**Integer**	No	The amount of this resource that physically exists on the node. Overcommit specifically doesn't apply to this. Note that overcommit currently only applies to "processors" and "memory", and so, for most cases, real and configured will always be the same.
utilized	**Integer**	No	The amount of this resource that is currently reported as utilized by resource managers.

NodeStateInformation

Field Name	Type	PUT	Description
powerState	**NodePower**	Yes	The state of the node's power system, as reported by the RM layer. Modifying the powerState is possible, and, if Moab is configured properly, a request will be made to modify the power state accordingly.
powerStateExpected	**NodePower**	No	The expected state of the node's power system. If a user has requested that a node be powered off (e.g. by modifying the powerState attribute to NodePower.OFF), the requested state will be shown in this field until the state change is completed. If there is no pending power change request, this will be null.
state	**NodeState**	Yes	The scheduling state of the Node, as reported by the resource management layer.
stateExpected	**NodeState**	No	The scheduling state of the Node, as expected by Moab. For example, Moab may think that a Node is "Busy" because it has allocated all configured resources, but a resource manager may report the state as "Running" based on actual utilization of the resources.
stateLastUpdatedDate	**Date**	No	A timestamp recording when the state of the Node was last modified.
subState	**String**	No	A text description of the state of the Node, with the intention of giving more details. Resource Managers may use this field to further describe the state being reported. Resource Managers should provide documented meaning to the possible sub-states that they can report.
subStateLast	**String**	No	The previous sub-state of the Node as reported by the resource management layer.
subStateLastUpdatedDate	**Date**	No	A timestamp recording when the sub-state was last modified.

NodePower

Represents the various options for a Node's power state.

Value	Description
NONE	
ON	
OFF	

NodeType

Represents the type of node as reported by a resource manager.

Value	Description
Compute	
License	
Network	
Storage	

Related Topics

- Nodes

Fields: Notification Conditions

See the associated Notification Conditions resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`notification-conditions`	Permissions
Hooks filename	`notification-conditions.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

NotificationCondition

A notification condition is related to an Event, but differs in three distinct areas:

- Notification conditions are a persistent condition of the system or a component rather than a single occurrence.
  - They are ongoing rather than reoccurring, which is why they are generated from NotificationConditions.
  - They may be observed many times, but the condition is always the same.
  - A good test for this is if something "is" wrong rather than something "went" wrong.
- Notification conditions can be acted on to result in a resolved state, mean the administrator or user can and must take actions to "fix" the condition or problem.
- Notification conditions contain state information based on administrator or user input, meaning that they contain information about the condition (similar to events), but also contain the "status" of the administrator's view of the notification, whether it is currently open, dismissed, or ignored.

In general, questions may be asked to ascertain whether an Event or a Notification Condition is the right fit for an occurrence. These questions, along with some sample situations, are provided below.

- Is the occurrence the root cause of a potentially ongoing condition?
  - A VM migration failed because the VM's state was unknown. The root cause was that the state was unknown, not that the VM migration failed. Therefore, VM migration failed would be an event, while the unknown state would be a notification condition.

- A VM service provision fails because there are no hypervisors that satisfy the requirements. This would be an event. Note that there may be a notification related to this failure, such as a service template requires a feature that does not exist on *any* hypervisors in the system, but this would be distinctly detected and managed from the provision failure event.

- A request to MWS failed because the connection between MWM and MongoDB was misconfigured. The failed request may be represented as an event, but a notification condition should exists that the connection between MWM and MongoDB was down.

- Can an administrator or user affect the outcome of the occurrence?

  - The outcome of a VM migration failing is in the past and cannot be changed by the administrator. However, the outcome of a *future* VM migration may be changed when the administrator resolves the root problem (i.e. VM state is unknown).

A notification condition is an observed condition for which Notifications are created. These conditions are created or updated on every PUT request based on the NotificationCondition.escalationLevel, NotificationCondition.origin, NotificationCondition.message, NotificationCondition.objectType, and NotificationCondition.objectId fields. When notifications are requested, these observed conditions are used to create the notifications for the requesting user.

While notification conditions may not be deleted, they "expire" after a specified amount of time and are no longer considered as active conditions for which notifications are created.

Field Name	Type	PUT	Description
id	**String**	No	The identifier of the condition.
createdDate	**Date**	No	The date that the condition first started appearing.
details	**Map<String, Map>**	No	Arbitrary storage of details for this notification. This could include "pluginType", "pluginId", etc.

Field Name	Type	PUT	Description
escalationLevel	**EscalationLevel**	No	The escalation level of the condition. This indicates who should care about the condition or who can respond to it. This may NOT be EscalationLevel.INTERNAL.
expirationDate	**Date**	No	The date at which the condition is considered "expired" and notifications are no longer created for it. This is typically set using the expirationDuration field.
expirationDuration	**Long**	No	The duration in seconds that may pass before a notification will not be created for a user. Effectively this can disable notifications from being created if they are too old. When this field is set, it will set the expirationDate field automatically each time the condition is updated or on creation. This field must be set to 1 or greater or else set to null.
message	**String**	No	A message detailing the notification and why it exists, with possible action items.
objectId	**String**	No	The identifier of the object which this notification affects, such as "node1" or "vm1".
objectType	**String**	No	The object type that this notification affects, such as "Node", "VM", "System", etc.
observedDate	**Date**	No	The latest date that the condition was observed. If this field is not set in an update request, it will automatically be set to the current date.
origin	**String**	No	The origin of the notification.
tenant	**Map<String, Map>**	No	The tenant that this notification came from. (contains tenant id and name)

EscalationLevel

Value	Description
USER	

Value	Description
POWER_USER	
ADMIN	
INTERNAL	

Related Topics

- Notification Conditions

Fields: Notifications

See the associated Notifications resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`notifications`	Permissions
Hooks filename	`notifications.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Notification

Notifications, while related to Events, are used for different purposes. See NotificationCondition for more information on when notifications should be used as opposed to events.

Notifications are a per-user representation of all notification conditions present in the system at any one time. When an administrator or user requests this resource, notifications are automatically created from the notification conditions that they have access to (determined by the Notification.tenant or the NotificationCondition.escalationLevel fields).

Notifications are expected to contain messages and details that may be understood by a user or admin depending on the escalation level, and contain fields that control whether the user or admin will be notified of future updates to their corresponding condition.

Notifications cannot be deleted, but they can be marked as ignored (see Notification.ignoredDate or dismissed (see Notification.dismissedDate).

Field Name	Type	PUT	Description
id	**String**	No	The identifier of the notification.
conditionId	**String**	No	The identifier of the NotificationCondition from which this notification was created.
createdDate	**Date**	No	The date that the notification condition first appeared.
details	**Map<String, Map>**	No	Arbitrary storage of details for this notification. This could include "pluginType", "pluginId", etc.
dismissedDate	**Date**	No	The date that the notification was dismissed by a user or admin, meaning that they acknowledged the notification and wanted to know of future updates to this notification. This field is cleared every time the attached notification condition is updated/observed again. (See also: conditionId.)
ignoredDate	**Date**	No	The date that the notification was ignored by a user or admin, meaning that they acknowledged the notification now and in the future and did not wish to know of any updates. This field is never cleared, even if the attached notification condition is updated/observed again.

Field Name	Type	PUT	Description
message	**String**	No	A message detailing the notification and why it exists, with possible action items.
objectId	**String**	No	The identifier of the object which this notification affects, such as "node1" or "vm1".
objectType	**String**	No	The object type that this notification affects, such as "Node", "VM", "System", etc.
observedDate	**Date**	No	The latest date that the notification condition was observed. If this field, ignoredDate, and dismissedDate are not set during an update (i.e. a user/admin is not ignoring or dismissing the notification), this field will automatically be set to the current date.
origin	**String**	No	The origin of the notification.
user	**String**	No	The user that this notification was created for.

Related Topics

- Notifications

Fields: Plugins

See the associated Plugins resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`plugins`	Permissions
Hooks filename	`plugins.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

PluginInstance

This class represents a configured plugin created from a plugin type.

Field Name	Type	POST	PUT	Description
id	**String**	Yes	No	Unique identifier for the plugin. Must contain at least one letter and must also start with a letter. Reserved IDs are "all" and "moab". If these are used an error with be returned.
autoStart	**Boolean**	Yes	Yes	Whether the plugin should start automatically when created.
config	**Map<String, Map>**	Yes	Yes	The configuration of the plugin. Plugin types may define constraints on the configuration, therefore it is recommended to view the plugin type's documentation for more information on required and optional fields. Regardless, the plugin configuration supports arbitrary keys and values.
dateCreated	**Date**	No	No	The date that this plugin was created.
lastPollDate	**Date**	No	No	The date of the last polling event that occurred. This may be null if the plugin is in the STOPPED state or has not yet been polled.
lastUpdated	**Date**	No	No	The date that this plugin was last updated.
nextPollDate	**Date**	No	No	The date of the next polling event that is scheduled to occur. This may be null if the plugin is in the STOPPED state.
pluginType	**String**	Yes	No	The plugin name as in Native or Example for the plugin called ExamplePlugin.
pollInterval	**Integer**	Yes	Yes	The polling interval to use for the plugin in seconds. This is ignored if the plugin type does not support polling.

Field Name	Type	POST	PUT	Description
precedence	**Long**	Yes	Yes	The precedence of this plugin, with the lowest value being the highest precedence. Minimum of 1. This is used when doing data consolidation when reporting current state data. Lower numbers results in a higher precedence (i.e. 1 is higher precedence than 10). If not specified during creation, this will be automatically set to 1 for the first plugin created, then 1 greater for each subsequently created plugin (i.e. 1 for plugin1, 2 for plugin2, etc). It is always set to 1 greater than the plugin with the greatest precedence number (i.e. 11 if two plugins exist with precedence 1 and 10).
state	**PluginState**	No	No	The current state of the plugin. Defaults to PluginState.STOPPED.

PluginState

Represents the current state of a plugin.

Value	Description
STOPPED	The plugin is created and ready for use, but is not currently receiving any events
STARTED	The plugin is currently receiving events and is working correctly.
PAUSED	The plugin is currently not receiving any events but is also not stopped. This should be used when polling or other events should stop only temporarily without firing the stop events.
ERRORED	MWS has detected an error with the plugin and has automatically stopped it. Errors could be due to the following reasons: 1. An invalid configuration was detected when running the AbstractPlugin.configure method. 2. An unexpected exception was thrown during an event, such as during polling.

API version 2

PluginInstance

This class represents a configured plugin created from a plugin type.

Field Name	Type	POST	PUT	Description
id	**String**	Yes	No	Unique identifier for the plugin. Must contain at least one letter and must also start with a letter. Reserved IDs are "all" and "moab". If these are used an error with be returned.
autoStart	**Boolean**	Yes	Yes	Whether the plugin should start automatically when created.
config	**Map<String, Map>**	Yes	Yes	The configuration of the plugin. Plugin types may define constraints on the configuration, therefore it is recommended to view the plugin type's documentation for more information on required and optional fields. Regardless, the plugin configuration supports arbitrary keys and values.
dateCreated	**Date**	No	No	The date that this plugin was created.
lastPollDate	**Date**	No	No	The date of the last polling event that occurred. This may be null if the plugin is in the STOPPED state or has not yet been polled.
lastUpdated	**Date**	No	No	The date that this plugin was last updated.
nextPollDate	**Date**	No	No	The date of the next polling event that is scheduled to occur. This may be null if the plugin is in the STOPPED state.
pluginType	**String**	Yes	No	The plugin name as in Native or Example for the plugin called ExamplePlugin.
pollInterval	**Integer**	Yes	Yes	The polling interval to use for the plugin in seconds. This is ignored if the plugin type does not support polling.

Field Name	Type	POST	PUT	Description
precedence	**Long**	Yes	Yes	The precedence of this plugin, with the lowest value being the highest precedence. Minimum of 1. This is used when doing data consolidation when reporting current state data. Lower numbers results in a higher precedence (i.e. 1 is higher precedence than 10). If not specified during creation, this will be automatically set to 1 for the first plugin created, then 1 greater for each subsequently created plugin (i.e. 1 for plugin1, 2 for plugin2, etc). It is always set to 1 greater than the plugin with the greatest precedence number (i.e. 11 if two plugins exist with precedence 1 and 10).
state	**PluginState**	No	No	The current state of the plugin. Defaults to PluginState.STOPPED.

PluginState

Represents the current state of a plugin.

Value	Description
STOPPED	The plugin is created and ready for use, but is not currently receiving any events
STARTED	The plugin is currently receiving events and is working correctly.
PAUSED	The plugin is currently not receiving any events but is also not stopped. This should be used when polling or other events should stop only temporarily without firing the stop events.
ERRORED	MWS has detected an error with the plugin and has automatically stopped it. Errors could be due to the following reasons: 1. An invalid configuration was detected when running the AbstractPlugin.configure method. 2. An unexpected exception was thrown during an event, such as during polling.

Related Topics

- Plugins

Fields: Plugin Types

See the associated Plugin Types resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`plugin-types`	Permissions
Hooks filename	`plugin-types.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

PluginType

Represents a MWS plugin type. All fields in this class are generated from plugin project and type metadata and cannot be modified directly. Consequentially, all fields are only valid for list/show/GET operations.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier of the plugin type. This is based on the class name of the plugin. Ex: **Plugin Class Name -> ID** NativePlugin -> Native MSMPlugin -> MSM MyExamplePlugin -> MyExample
author	**String**	No	The main author (company or person) of the plugin type.
commonsVersion	**String**	No	A string representing the restriction on which version of the plugin framework (plugins-commons dependency) is required for the plugin type. In the format 'COMMONS_VERSION > *', meaning that any version greater or equal to COMMONS_VERSION is valid.
description	**String**	No	The full description of the plugin type.
documentationLink	**String**	No	A full URL to the complete documentation for the plugin type.
email	**String**	No	The email of the author.
eventComponent	**Integer**	No	The event component ID of the plugin type. This should be unique for each plugin type and should be 1 or greater.
initialPlugins	**Map<String, Map>**	No	Represents the plugins that are initially configured when the plugin type is loaded. Each key represents the plugin ID.
instances	**List<List>**	No	The list of plugin instances created from this plugin type.

Field Name	Type	PUT	Description
issueManagementLink	**String**	No	A full URL to the issue management system or project for the plugin type.
license	**String**	No	The license of this plugin type, typically APACHE.
mwsVersion	**String**	No	A string representing the restriction on which version of MWS is required for the plugin type. In the format 'MWS_VERSION > *', meaning that any version greater or equal to MWS_VERSION is valid.
pollMethod	**boolean**	No	Indicates whether the plugin type has a defined 'poll' method (event handler) or not.
realizedEventComponent	**Integer**	No	The fully realized event component ID of the plugin type, including the MWS bits. This should take the form of 0x201. If the eventComponent is not set, this will be 0x2FF, meaning the component ID is an unknown plugin type.
scmLink	**String**	No	A full URL to the Source Control Management (SCM) system or project for the plugin type.
title	**String**	No	A short name describing the plugin type.
webServices	**List<List>**	No	The list of web service IDs that can be called for this plugin type.
website	**String**	No	The website of the author.

API version 2

PluginType

Represents a MWS plugin type. All fields in this class are generated from plugin project and type metadata and cannot be modified directly. Consequentially, all fields are only valid for list/show/GET operations.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier of the plugin type. This is based on the class name of the plugin. Ex: **Plugin Class Name -> ID** NativePlugin -> Native MSMPlugin -> MSM MyExamplePlugin -> MyExample
author	**String**	No	The main author (company or person) of the plugin type.
commonsVersion	**String**	No	A string representing the restriction on which version of the plugin framework (plugins-commons dependency) is required for the plugin type. In the format 'COMMONS_VERSION > *', meaning that any version greater or equal to COMMONS_VERSION is valid.
description	**String**	No	The full description of the plugin type.
documentationLink	**String**	No	A full URL to the complete documentation for the plugin type.
email	**String**	No	The email of the author.
eventComponent	**Integer**	No	The event component ID of the plugin type. This should be unique for each plugin type and should be 1 or greater.
initialPlugins	**Map<String, Map>**	No	Represents the plugins that are initially configured when the plugin type is loaded. Each key represents the plugin ID.
instances	**List<List>**	No	The list of plugin instances created from this plugin type.

Field Name	Type	PUT	Description
issueManagementLink	**String**	No	A full URL to the issue management system or project for the plugin type.
license	**String**	No	The license of this plugin type, typically APACHE.
mwsVersion	**String**	No	A string representing the restriction on which version of MWS is required for the plugin type. In the format 'MWS_VERSION > *', meaning that any version greater or equal to MWS_VERSION is valid.
pollMethod	**boolean**	No	Indicates whether the plugin type has a defined 'poll' method (event handler) or not.
realizedEventComponent	**Integer**	No	The fully realized event component ID of the plugin type, including the MWS bits. This should take the form of 0x201. If the eventComponent is not set, this will be 0x2FF, meaning the component ID is an unknown plugin type.
scmLink	**String**	No	A full URL to the Source Control Management (SCM) system or project for the plugin type.
title	**String**	No	A short name describing the plugin type.
webServices	**List<List>**	No	The list of web service IDs that can be called for this plugin type.
website	**String**	No	The website of the author.

Related Topics

- Plugin Types

Fields: Policies

See the associated Policies resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`policies`	Permissions
Hooks filename	`policies.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Policy

A Moab Workload Manager policy which can affect scheduling decisions such as resource allocation. A policy contains state, identifying information, a priority, and metadata about the policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

AutoVMMigrationPolicy

The Moab policy used to enabled and configure policy-based VM migration. Using information about data center applications and server load, Moab can aim to keep VMs in the data center optimally distributed across all hypervisors.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
genericMetricThresholds	**Map<String, Double>**	Yes	A map of generic metric pairings where each value must be greater than or equal to 0 such as: `METRIC1 => 5.6 METRIC2 => 0.0 METRIC3 => 102.4`
memoryUtilizationThreshold	**Double**	Yes	Defines the utilization threshold for memory. This must be greater than 0 and less than or equal to 1. A value of 1 effectively disables the threshold.

Field Name	Type	PUT	Description
migrationAlgorithmType	**AutoVMMigrationPolicyType**	Yes	Configures the VM migration algorithm utilized when the policy is active. Defaults to NONE. When ENABLED, this must **not** be set to NONE.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
processorUtilizationThreshold	**Double**	Yes	Defines the load utilization threshold for processors. This must be greater than 0 and less than or equal to 1. A value of 1 effectively disables the threshold.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

AutoVMMigrationPolicyType

Represents the algorithm used to migrate VMs when the AutoVMMigrationPolicy is used.

Value	Description
NONE	Used when the Auto VM Migration policy is currently disabled in Moab and before any settings are saved the first time. For example, if the policy is disabled on the first read of Moab policies, the AutoVMMigrationPolicy.migrationAlgorithmType will be set to NONE. If the policy is enabled and the type is set to OVERCOMMIT, followed by a disabling of the policy, it will then be represented as having a state of DISABLED with a migrationAlgorithmType of OVERCOMMIT.
OVERCOMMIT	Use the "overcommit" algorithm for migration. The goal of this algorithm is to equalize loads across hypervisors as migrations are queued due to overcommit conditions. This places VMs to be migrated on the least-loaded hypervisor available.
CONSOLIDATION	Use the "consolidation" algorithm for migration. The goal of this algorithm is to load hypervisors as close to thresholds as possible, without exceeding them. This policy places VMs to be migrated on the most loaded hypervisor possible, within these constraints. A second loop of this policy will select lightly-loaded hypervisors to be evacuated completely.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

HVAllocationOvercommitPolicy

The Hypervisor Allocation Overcommit policy controls how many virtual machines can be placed on a hypervisor. By enabling this policy, you are allowing Moab to allocate more resoures to a set of virtual machines than a hypervisor may actually have. This is possible due to virtualization. In other words, this policy allows you to set the high-water mark for virtual machine allocation for hypervisors. At least one of these limits must be greater than 1.0, or the policy will not be able to set to a state of PolicyState.ENABLED.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
memoryAllocationLimit	**Double**	Yes	Setting this to 1 effectively disables the allocation overcommit based on memory. If this and processorAllocationLimit are both set to 1.0 (the default), the policy state cannot be set to PolicyState.ENABLED.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.

Field Name	Type	PUT	Description
processorAllocationLimit	**Double**	Yes	The Allocation Limit defines the upper bound or maximum amount of VCPUs that can be created on any given hypervisor (HV). For example, if you have a hypervisor with 12 processors or cores (Moab sees them as 12 processors), and have an Allocation Limit of 2.0 for procs, then Moab will not allow, under any condition, more than 24 VCPU's to be allocated on this hypervisor. Remember: a VM can have one or more VCPU's. So, in this example, the HV could only support 8 VM's if they all had 3 VPCU's each. It could support 4 VM's if they had 6 VPCU's each, and so forth From http://www.adaptivecomputing.com/resources/docs/mwm/7-1-1/Content/topics/vm/allocation_limits_and_utilization_threshold.html Setting this to 1 effectively disables the allocation overcommit based on processors. If this and memoryAllocationLimit are both set to 1.0 (the default), the policy state cannot be set to PolicyState.ENABLED.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

NodeAllocationPolicy

Node allocation is the process of selecting the best resources to allocate to a job from a list of available resources. Moab contains a number of allocation algorithms that address this in the NodeAllocationPolicy.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
customPriorityFunction	**String**	Yes	Defines the priority function when the CustomPriority algorithm is used.
description	**String**	No	The user friendly description of the policy.
name	**String**	No	The user friendly name of the policy.
nodeAllocationAlgorithm	**NodeAllocationAlgorithm**	Yes	Configures the node allocation algorithm utilized when the policy is active. Defaults to NONE. When ENABLED, this must **not** be set to NONE.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.

Field Name	Type	PUT	Description
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

NodeAllocationAlgorithm

Represents the algorithm used to allocate Nodes when the NodeAllocationPolicy is used.

Value	Description
NONE	
InReportedOrder	
InReverseReportedOrder	
CustomPriority	
ProcessorLoad	
MinimumConfiguredResources	
Contiguous	
ProcessorSpeedBalance	
NodeSpeed	

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

MigrationExclusionListPolicy

Specify which virtual machines and hypervisors to exclude from automatic migration operations.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
hvExclusionList	**List<String>**	Yes	The list of hypervisor IDs on the exclusion list.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.

Field Name	Type	PUT	Description
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.
vmExclusionList	**List<String>**	Yes	The list of VM IDs on the exclusion list.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

FairsharePolicy

Fairshare allows historical resource utilization information to be incorporated into job feasibility and priority decisions. This feature allows site administrators to set system utilization targets for users, groups, accounts, classes, and QoS levels. Administrators can also specify the time frame over which resource utilization is evaluated in determining whether the goal is being reached. Parameters allow sites to specify the utilization metric, how historical information is aggregated, and the effect of fairshare state on scheduling behavior. You can specify fairshare targets for any credentials (such as user, group, and class) that administrators want such information to affect. http://docs.adaptivecomputing.com/mwm/archive/6-0-4/6.3fairshare.php

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.

Field Name	Type	PUT	Description
decayFactor	**Double**	Yes	Specifies decay rate applied to past fairshare interval when computing effective fairshare usage. Values may be in the range of 0.01 to 1.0. A smaller value causes more rapid decay causing aged usage to contribute less to the overall effective fairshare usage. A value of 1.0 indicates that no decay will occur and all fairshare intervals will be weighted equally when determining effective fairshare usage.
depth	**Integer**	Yes	Number of fairshare windows factored into current fairshare utilization. Note: The number of available fairshare windows is bounded by the MAX_FSDEPTH value (32 in Moab).
description	**String**	No	The user friendly description of the policy.
intervalSeconds	**Long**	Yes	Specifies the length of each fairshare window in seconds.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

Field Name	Type	PUT	Description
usageMetric	FairshareUsageMetric	Yes	As Moab runs, it records how available resources are used. Each iteration it updates fairshare resource utilization statistics. Resource utilization is tracked in accordance with the usage metric allowing various aspects of resource consumption information to be measured. The usage metric allows selection of both the types of resources to be tracked as well as the method of tracking. It provides the option of tracking usage by dedicated or consumed resources, where dedicated usage tracks what the scheduler assigns to the job and consumed usage tracks what the job actually uses.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

FairshareUsageMetric

Specifies the unit of tracking FairsharePolicy usage. http://docs.adaptivecomputing.com/mwm/archive/6-0-4/6.3fairshare.php#fspolicy

Value	Description
NONE	
DEDICATED_PROCESSOR_SECONDS_DELIVERED	Usage tracked by processor seconds dedicated to each job relative to other processor seconds dedicated to other jobs on the system. (Useful in dedicated node environments.)
DEDICATED_PROCESSOR_SECONDS_AVAILABLE	Usage tracked by processor seconds dedicated to each job relative to all available processor seconds dedicated to other jobs on the system. (Useful in dedicated node environments.)

Value	Description
DEDICATED_ PROCESSOR_ EQUIVALENT_ SECONDS_DELIVERED	Usage tracked by processor-equivalent seconds dedicated to each job relative to other processor-equivalent seconds dedicated to other jobs on the system. (Useful in dedicated and shared nodes environments).
UTILIZED_ PROCESSOR_ SECONDS_DELIVERED	Usage tracked by processor seconds used by each job relative to other processor seconds used by other jobs on the system. (Useful in shared node/SMP environments.)

API version 2

Policy

A Moab Workload Manager policy which can affect scheduling decisions such as resource allocation. A policy contains state, identifying information, a priority, and metadata about the policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

AutoVMMigrationPolicy

The Moab policy used to enabled and configure policy-based VM migration. Using information about data center applications and server load, Moab can aim to keep VMs in the data center optimally distributed across all hypervisors.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
genericMetricThresholds	**Map<String, Double>**	Yes	A map of generic metric pairings where each value must be greater than or equal to 0 such as: `METRIC1 => 5.6 METRIC2 => 0.0 METRIC3 => 102.4`
memoryUtilizationThreshold	**Double**	Yes	Defines the utilization threshold for memory. This must be greater than 0 and less than or equal to 1. A value of 1 effectively disables the threshold.

Field Name	Type	PUT	Description
migrationAlgorithmType	**AutoVMMigrationPolicyType**	Yes	Configures the VM migration algorithm utilized when the policy is active. Defaults to NONE. When ENABLED, this must **not** be set to NONE.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
processorUtilizationThreshold	**Double**	Yes	Defines the load utilization threshold for processors. This must be greater than 0 and less than or equal to 1. A value of 1 effectively disables the threshold.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

AutoVMMigrationPolicyType

Represents the algorithm used to migrate VMs when the AutoVMMigrationPolicy is used.

Value	Description
NONE	Used when the Auto VM Migration policy is currently disabled in Moab and before any settings are saved the first time. For example, if the policy is disabled on the first read of Moab policies, the AutoVMMigrationPolicy.migrationAlgorithmType will be set to NONE. If the policy is enabled and the type is set to OVERCOMMIT, followed by a disabling of the policy, it will then be represented as having a state of DISABLED with a migrationAlgorithmType of OVERCOMMIT.
OVERCOMMIT	Use the "overcommit" algorithm for migration. The goal of this algorithm is to equalize loads across hypervisors as migrations are queued due to overcommit conditions. This places VMs to be migrated on the least-loaded hypervisor available.
CONSOLIDATION	Use the "consolidation" algorithm for migration. The goal of this algorithm is to load hypervisors as close to thresholds as possible, without exceeding them. This policy places VMs to be migrated on the most loaded hypervisor possible, within these constraints. A second loop of this policy will select lightly-loaded hypervisors to be evacuated completely.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

HVAllocationOvercommitPolicy

The Hypervisor Allocation Overcommit policy controls how many virtual machines can be placed on a hypervisor. By enabling this policy, you are allowing Moab to allocate more resoures to a set of virtual machines than a hypervisor may actually have. This is possible due to virtualization. In other words, this policy allows you to set the high-water mark for virtual machine allocation for hypervisors. At least one of these limits must be greater than 1.0, or the policy will not be able to set to a state of PolicyState.ENABLED.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
memoryAllocationLimit	**Double**	Yes	Setting this to 1 effectively disables the allocation overcommit based on memory. If this and processorAllocationLimit are both set to 1.0 (the default), the policy state cannot be set to PolicyState.ENABLED.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.

Field Name	Type	PUT	Description
processorAllocationLimit	**Double**	Yes	The Allocation Limit defines the upper bound or maximum amount of VCPUs that can be created on any given hypervisor (HV). For example, if you have a hypervisor with 12 processors or cores (Moab sees them as 12 processors), and have an Allocation Limit of 2.0 for procs, then Moab will not allow, under any condition, more than 24 VCPU's to be allocated on this hypervisor. Remember: a VM can have one or more VCPU's. So, in this example, the HV could only support 8 VM's if they all had 3 VPCU's each. It could support 4 VM's if they had 6 VPCU's each, and so forth From http://www.adaptivecomputing.com/resources/docs/mwm/7-1-1/Content/topics/vm/allocation_limits_and_utilization_threshold.html Setting this to 1 effectively disables the allocation overcommit based on processors. If this and memoryAllocationLimit are both set to 1.0 (the default), the policy state cannot be set to PolicyState.ENABLED.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

NodeAllocationPolicy

Node allocation is the process of selecting the best resources to allocate to a job from a list of available resources. Moab contains a number of allocation algorithms that address this in the NodeAllocationPolicy.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
customPriorityFunction	**String**	Yes	Defines the priority function when the CustomPriority algorithm is used.
description	**String**	No	The user friendly description of the policy.
name	**String**	No	The user friendly name of the policy.
nodeAllocationAlgorithm	**NodeAllocationAlgorithm**	Yes	Configures the node allocation algorithm utilized when the policy is active. Defaults to NONE. When ENABLED, this must **not** be set to NONE.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.

Field Name	Type	PUT	Description
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

NodeAllocationAlgorithm

Represents the algorithm used to allocate Nodes when the NodeAllocationPolicy is used.

Value	Description
NONE	
InReportedOrder	
InReverseReportedOrder	
CustomPriority	
ProcessorLoad	
MinimumConfiguredResources	
Contiguous	
ProcessorSpeedBalance	
NodeSpeed	

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

MigrationExclusionListPolicy

Specify which virtual machines and hypervisors to exclude from automatic migration operations.

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.
description	**String**	No	The user friendly description of the policy.
hvExclusionList	**List<String>**	Yes	The list of hypervisor IDs on the exclusion list.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.

Field Name	Type	PUT	Description
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.
vmExclusionList	**List<String>**	Yes	The list of VM IDs on the exclusion list.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

FairsharePolicy

Fairshare allows historical resource utilization information to be incorporated into job feasibility and priority decisions. This feature allows site administrators to set system utilization targets for users, groups, accounts, classes, and QoS levels. Administrators can also specify the time frame over which resource utilization is evaluated in determining whether the goal is being reached. Parameters allow sites to specify the utilization metric, how historical information is aggregated, and the effect of fairshare state on scheduling behavior. You can specify fairshare targets for any credentials (such as user, group, and class) that administrators want such information to affect. http://docs.adaptivecomputing.com/mwm/archive/6-0-4/6.3fairshare.php

This class inherits fields from Policy.

Field Name	Type	PUT	Description
id	**String**	No	The unique identifier for the policy. Must contain only lowercase letters and dashes, such as 'auto-vm-migration'.
conflicted	**Boolean**	No	Signifies whether any other policies are currently activated that potentially conflict with this policy. If true, it signifies a **potential** conflict.

Field Name	Type	PUT	Description
decayFactor	**Double**	Yes	Specifies decay rate applied to past fairshare interval when computing effective fairshare usage. Values may be in the range of 0.01 to 1.0. A smaller value causes more rapid decay causing aged usage to contribute less to the overall effective fairshare usage. A value of 1.0 indicates that no decay will occur and all fairshare intervals will be weighted equally when determining effective fairshare usage.
depth	**Integer**	Yes	Number of fairshare windows factored into current fairshare utilization. Note: The number of available fairshare windows is bounded by the MAX_FSDEPTH value (32 in Moab).
description	**String**	No	The user friendly description of the policy.
intervalSeconds	**Long**	Yes	Specifies the length of each fairshare window in seconds.
name	**String**	No	The user friendly name of the policy.
potentialConflicts	**Set<String>**	No	A set of policy IDs that may potentially conflict with this policy.
priority	**Integer**	No	Indicates the absolute priority of the policy with respect to others. It is possible that more than one policy has the same priority. The higher the number, the greater the priority. Minimum is 0.
state	**PolicyState**	Yes	Defines the current state of the policy: enabled or disabled. Defaults to PolicyState.DISABLED.
tags	**Set<String>**	No	A set of strings that can be used to aid in filtering or querying policies.
types	**Set<String>**	No	A set of categories or types that the policy is included in. This may be used to filter or query on groups of policies.

Field Name	Type	PUT	Description
usageMetric	FairshareUsageMetric	Yes	As Moab runs, it records how available resources are used. Each iteration it updates fairshare resource utilization statistics. Resource utilization is tracked in accordance with the usage metric allowing various aspects of resource consumption information to be measured. The usage metric allows selection of both the types of resources to be tracked as well as the method of tracking. It provides the option of tracking usage by dedicated or consumed resources, where dedicated usage tracks what the scheduler assigns to the job and consumed usage tracks what the job actually uses.

PolicyState

Represents the state of a policy. A policy may only be enabled or disabled.

Value	Description
ENABLED	The policy is enabled or active.
DISABLED	The policy is disabled or inactive.

FairshareUsageMetric

Specifies the unit of tracking FairsharePolicy usage. http://docs.adaptivecomputing.com/mwm/archive/6-0-4/6.3fairshare.php#fspolicy

Value	Description
NONE	
DEDICATED_PROCESSOR_SECONDS_DELIVERED	Usage tracked by processor seconds dedicated to each job relative to other processor seconds dedicated to other jobs on the system. (Useful in dedicated node environments.)
DEDICATED_PROCESSOR_SECONDS_AVAILABLE	Usage tracked by processor seconds dedicated to each job relative to all available processor seconds dedicated to other jobs on the system. (Useful in dedicated node environments.)

Value	Description
DEDICATED_ PROCESSOR_ EQUIVALENT_ SECONDS_DELIVERED	Usage tracked by processor-equivalent seconds dedicated to each job relative to other processor-equivalent seconds dedicated to other jobs on the system. (Useful in dedicated and shared nodes environments).
UTILIZED_ PROCESSOR_ SECONDS_DELIVERED	Usage tracked by processor seconds used by each job relative to other processor seconds used by other jobs on the system. (Useful in shared node/SMP environments.)

Related Topics

- Policies

Fields: Principals

See the associated Principals resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`principals`	Permissions
Hooks filename	`principals.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Principal

A principal maps to a set of ldap users, ldap groups, pam users, and/or pam groups. MWS roles are attached to the principals to authorize the group to use the specific MWS roles.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this principal.
attachedRoles	**Set<Role>**	Yes	Yes	The MWS roles this principal is authorized to use.
description	**String**	Yes	Yes	The principal description.
groups	**List<Map>**	Yes	Yes	The groups associated with this principal. Each group has a name and a type. The valid types of groups are LDAPOU, LDAPGROUP, PAMGROUP, and SSO. Example group: `{"name": "CN=Engineering,CN=Users,DC=corp,DC=cloud,DC=dev", "type": "LDAPGROUP"}` or `{"name": "engineering", "type": "PAMGROUP"}`
name	**String**	Yes	Yes	The unique human-readable name of this principal. Required during POST.
users	**List<Map>**	Yes	Yes	The users associated with this principal. Each user has a name and type. The valid types of users are LDAP and PAM. Example user: `{"name": "jhammon", "type": "LDAP"}` or `{"name": "jhammon", "type": "PAM"}`

Role

A role defines a set of permissions that are based on the proxy-user. If no proxy user is specified then access to objects in MWS are limited to its application permissions. For example if the application has permission to update all resources in MWS and no proxy-user is specified in the request then the request can access all resources in MWS.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
description	**String**	Yes	Yes	The role description.
name	**String**	Yes	Yes	The unique human-readable name of this role. Required during POST.
permissions	**List<Permission>**	Yes	Yes	The set of permissions enforced based on the proxy-user.
scope	**PrivilegeScope**	No	No	

Permission

Represents a permission

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
action	**String**	No	No	The action that can be performed on the resource.
administrator	**Boolean**	No	No	If true, grants full rights over the given resource for the given action. For example, if resource is "jobs" and action is "update" and administrator is true, then this permission allows the user to update any job, not just jobs owned by the user.
description	**String**	No	No	A description of this permission.
fieldPath	**String**	No	No	Field level ACL control, if null or '*', all fields are accessible, otherwise requests must match dot delimited path. Currently only checked when doing writable actions. Example - attributes.*: create|update
label	**String**	No	No	A human readable label for this permission.

Field Name	Type	POST	PUT	Description
resource	**String**	No	No	The resource the permission applies to.
resourceFilter	**Map<String, Map>**	No	No	A map used to limit which resource instances this permission applies to. If this is null then the permission will apply to all instances of the resource. For api permissions the filter uses mongo query syntax.
scope	**PrivilegeScope**	No	No	Whether this permission applies to the principal's tenant-associated resources or globally
type	**String**	No	No	The type of the permission. Only 'api' type permissions are enforced.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.

Value	Description
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

API version 2

Principal

A principal maps to a set of ldap users, ldap groups, pam users, and/or pam groups. MWS roles are attached to the principals to authorize the group to use the specific MWS roles.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this principal.
attachedRoles	**Set<Role>**	Yes	Yes	The MWS roles this principal is authorized to use.
description	**String**	Yes	Yes	The principal description.
groups	**List<Map>**	Yes	Yes	The groups associated with this principal. Each group has a name and a type. The valid types of groups are LDAPOU, LDAPGROUP, PAMGROUP, and SSO. Example group: `{"name": "CN=Engineering,CN=Users,DC=corp,DC=cloud,DC=dev", "type": "LDAPGROUP"}` or `{"name": "engineering", "type": "PAMGROUP"}`
name	**String**	Yes	Yes	The unique human-readable name of this principal. Required during POST.
users	**List<Map>**	Yes	Yes	The users associated with this principal. Each user has a name and type. The valid types of users are LDAP and PAM. Example user: `{"name": "jhammon", "type": "LDAP"}` or `{"name": "jhammon", "type": "PAM"}`

Role

A role defines a set of permissions that are based on the proxy-user. If no proxy user is specified then access to objects in MWS are limited to its application permissions. For example if the application has permission to update all resources in MWS and no proxy-user is specified in the request then the request can access all resources in MWS.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
description	**String**	Yes	Yes	The role description.
name	**String**	Yes	Yes	The unique human-readable name of this role. Required during POST.
permissions	**List<Permission>**	Yes	Yes	The set of permissions enforced based on the proxy-user.
scope	**PrivilegeScope**	No	No	

Permission

Represents a permission

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
action	**String**	No	No	The action that can be performed on the resource.
administrator	**Boolean**	No	No	If true, grants full rights over the given resource for the given action. For example, if resource is "jobs" and action is "update" and administrator is true, then this permission allows the user to update any job, not just jobs owned by the user.
description	**String**	No	No	A description of this permission.
fieldPath	**String**	No	No	Field level ACL control, if null or '*', all fields are accessible, otherwise requests must match dot delimited path. Currently only checked when doing writable actions. Example - attributes.*: create|update
label	**String**	No	No	A human readable label for this permission.

Field Name	Type	POST	PUT	Description
resource	**String**	No	No	The resource the permission applies to.
resourceFilter	**Map<String, Map>**	No	No	A map used to limit which resource instances this permission applies to. If this is null then the permission will apply to all instances of the resource. For api permissions the filter uses mongo query syntax.
scope	**PrivilegeScope**	No	No	Whether this permission applies to the principal's tenant-associated resources or globally
type	**String**	No	No	The type of the permission. Only 'api' type permissions are enforced.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.

Value	Description
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

Related Topics

- Principals

Fields: Report Datapoints

See the associated Reports resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`reports/datapoints`	Permissions
Hooks filename	`reports.datapoints.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Datapoint

A metric that measures system state over a specified period of time. For example, a datapoint may contain data on CPU utilization by specific users. A datapoint is generated by the consolidation of zero or more Samples. It could be said that a datapoint represents a smoothing of samples.

Field Name	Type	Description
id	**Long**	
data	**Map<String, Map>**	The actual consolidated sample data. This property may be 'null' if the Report.minimumSampleSize was not met when consolidating the datapoint.
endDate	**Date**	The ending date that the datapoint covers.
firstSampleDate	**Date**	The date of the first sample consolidated in this datapoint. (See also: Sample.timestamp.)
lastSampleDate	**Date**	The date of the last sample consolidated in this datapoint. (See also: Sample.timestamp.)
startDate	**Date**	The beginning date that the datapoint covers.

API version 2

Datapoint

A metric that measures system state over a specified period of time. For example, a datapoint may contain data on CPU utilization by specific users. A datapoint is generated by the consolidation of zero or more Samples. It could be said that a datapoint represents a smoothing of samples.

Field Name	Type	Description
id	**Long**	
data	**Map<String, Map>**	The actual consolidated sample data. This property may be 'null' if the Report.minimumSampleSize was not met when consolidating the datapoint.
endDate	**Date**	The ending date that the datapoint covers.
firstSampleDate	**Date**	The date of the first sample consolidated in this datapoint. (See also: Sample.timestamp.)
lastSampleDate	**Date**	The date of the last sample consolidated in this datapoint. (See also: Sample.timestamp.)
startDate	**Date**	The beginning date that the datapoint covers.

Related Topics

- Reports

Fields: Reports

See the associated Reports resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`reports`	Permissions
Hooks filename	`reports.groovy`	Pre- and Post-Processing Hooks

Type	Value	Additional information
Distinct query-supported	Yes	Distinct

API version 3

Report

A set of time-based values that share similar context. For example, a report may contain data on CPU or power utilization for all nodes in a cluster.

A report is composed of metadata and a collection of Datapoints. Samples are also associated with reports, but these are consolidated using the Report.consolidationFunction to create Datapoints.

If the datapoint documents are being truncated in any way or there are warnings about documents being too large, it may be necessary to increase the Report.reportDocumentSize.

Field Name	Type	POST	Description
id	**String**	No	The unique identifier for the report. This is automatically assigned and will be ignored if specified duration creation.
consolidationFunction	**String**	Yes	The consolidation function is the process used to convert a set of samples into a datapoint. Currently the only supported function is "average", which is used if none is specified.
datapointDuration	**Long**	Yes	Required. How long the datapoints are, in seconds.
datapoints	**List<Datapoint>**	Yes	This is the set of datapoints that have been consolidated for the report or are desired to be included in the report during creation time. In the latter case, these represent historical data created outside of the reporting framework. Only present when getting a single report.
description	**String**	Yes	A description of the report.
keepSamples	**Boolean**	Yes	Controls if samples are retained after consolidation. Defaults to false, which means that after consolidation, samples are discarded.
minimumSampleSize	**Integer**	Yes	If number of samples is below this number, the datapoint data field is "null". Defaults to 1.

Field Name	Type	POST	Description
name	**String**	Yes	Required. A unique name identifying the report. Valid characters are all alphanumeric characters, dashes (-), periods (.), and underscores (_).
reportDocumentSize	**Long**	Yes	The maximum size in bytes of each datapoint document stored for this report. This option is provided to maximize the amount of disk space used for a single report. The default value for this option is 100*1024, or 100 KB. The maximum value of this option is 16*1024*1024 (16777216) or 16 MB, which represents the maximum document size in MongoDB. See also http://www.mongodb.org/display/DOCS/Documents. Keep in mind that when creating a new report, MongoDB will initialize all needed space for all possible datapoint documents up front. This can easily fill a disk unless this parameter is modified.
reportSize	**Long**	Yes	Required. The size of the report in datapoints. After this number of datapoints is reached, the old datapoints will be discarded. WARNING: On report creation, a Mongo collection will be initialized that is the maximum size of a single entry (currently 16 MB) multiplied by the report size. Be careful in setting a large report size as this will quickly allocate the entire disk if many reports with large report sizes are created.

Datapoint

A metric that measures system state over a specified period of time. For example, a datapoint may contain data on CPU utilization by specific users. A datapoint is generated by the consolidation of zero or more Samples. It could be said that a datapoint represents a smoothing of samples.

Field Name	Type	POST	Description
id	**Long**	No	

Field Name	Type	POST	Description
data	**Map<String, Map>**	No	The actual consolidated sample data. This property may be 'null' if the Report.minimumSampleSize was not met when consolidating the datapoint.
endDate	**Date**	No	The ending date that the datapoint covers.
firstSampleDate	**Date**	No	The date of the first sample consolidated in this datapoint. (See also: Sample.timestamp.)
lastSampleDate	**Date**	No	The date of the last sample consolidated in this datapoint. (See also: Sample.timestamp.)
startDate	**Date**	No	The beginning date that the datapoint covers.

API version 2

Report

A set of time-based values that share similar context. For example, a report may contain data on CPU or power utilization for all nodes in a cluster.

A report is composed of metadata and a collection of Datapoints. Samples are also associated with reports, but these are consolidated using the Report.consolidationFunction to create Datapoints.

If the datapoint documents are being truncated in any way or there are warnings about documents being too large, it may be necessary to increase the Report.reportDocumentSize.

Field Name	Type	POST	Description
id	**String**	No	The unique identifier for the report. This is automatically assigned and will be ignored if specified duration creation.
consolidationFunction	**String**	Yes	The consolidation function is the process used to convert a set of samples into a datapoint. Currently the only supported function is "average", which is used if none is specified.
datapointDuration	**Long**	Yes	Required. How long the datapoints are, in seconds.
datapoints	**List<Datapoint>**	Yes	This is the set of datapoints that have been consolidated for the report or are desired to be included in the report during creation time. In the latter case, these represent historical data created outside of the reporting framework. Only present when getting a single report.
description	**String**	Yes	A description of the report.
keepSamples	**Boolean**	Yes	Controls if samples are retained after consolidation. Defaults to false, which means that after consolidation, samples are discarded.
minimumSampleSize	**Integer**	Yes	If number of samples is below this number, the datapoint data field is "null". Defaults to 1.

Field Name	Type	POST	Description
name	**String**	Yes	Required. A unique name identifying the report. Valid characters are all alphanumeric characters, dashes (-), periods (.), and underscores (_).
reportDocumentSize	**Long**	Yes	The maximum size in bytes of each datapoint document stored for this report. This option is provided to maximize the amount of disk space used for a single report. The default value for this option is 100*1024, or 100 KB. The maximum value of this option is 16*1024*1024 (16777216) or 16 MB, which represents the maximum document size in MongoDB. See also http://www.mongodb.org/display/DOCS/Documents. Keep in mind that when creating a new report, MongoDB will initialize all needed space for all possible datapoint documents up front. This can easily fill a disk unless this parameter is modified.
reportSize	**Long**	Yes	Required. The size of the report in datapoints. After this number of datapoints is reached, the old datapoints will be discarded. WARNING: On report creation, a Mongo collection will be initialized that is the maximum size of a single entry (currently 16 MB) multiplied by the report size. Be careful in setting a large report size as this will quickly allocate the entire disk if many reports with large report sizes are created.

Datapoint

A metric that measures system state over a specified period of time. For example, a datapoint may contain data on CPU utilization by specific users. A datapoint is generated by the consolidation of zero or more Samples. It could be said that a datapoint represents a smoothing of samples.

Field Name	Type	POST	Description
id	**Long**	No	

Field Name	Type	POST	Description
data	**Map<String, Map>**	No	The actual consolidated sample data. This property may be 'null' if the Report.minimumSampleSize was not met when consolidating the datapoint.
endDate	**Date**	No	The ending date that the datapoint covers.
firstSampleDate	**Date**	No	The date of the first sample consolidated in this datapoint. (See also: Sample.timestamp.)
lastSampleDate	**Date**	No	The date of the last sample consolidated in this datapoint. (See also: Sample.timestamp.)
startDate	**Date**	No	The beginning date that the datapoint covers.

Related Topics

- Reports

Fields: Reservations

See the associated Reservations resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`reservations`	Permissions
Hooks filename	`reservations.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

Reservation

A reservation is the mechanism by which Moab guarantees the availability of a set of resources at a particular time. Each reservation consists of three major components: (1) a set of resources, (2) a time frame, and (3) an access control list. It is a scheduler role to ensure that the access control list is not violated during the reservation's lifetime (that is, its time frame) on the resources listed. For example, a reservation may specify that node002 is reserved for user Tom on Friday. The scheduler is thus constrained to make certain that only Tom's jobs can use node002 at any time on Friday.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of the reservation.
accountingAccount	**String**	Yes	No	Accountable Account.
accountingGroup	**String**	Yes	No	Accountable Group.
accountingQOS	**String**	Yes	No	Accountable QOS.
accountingUser	**String**	Yes	No	Accountable User.
aclRules	**Set<AclRule>**	Yes	No	The set of access control rules associated with this reservation.
allocatedNodeCount	**Integer**	No	No	The number of allocated nodes for this reservation.
allocatedNodes	**Set<DomainProxyVersion1>**	No	No	The nodes allocated to the reservation.
allocatedProcessorCount	**Integer**	No	No	The number of allocated processors.
allocatedTaskCount	**Integer**	No	No	The number of allocated tasks.

Field Name	Type	POST	PUT	Description
comments	**String**	Yes	No	Reservation's comments or description.
creationDate	**Date**	No	No	Creation date. Automatically set by Moab when a user creates the reservation.
duration	**Long**	Yes	No	The duration of the reservation (in seconds).
endDate	**Date**	Yes	No	The end date of the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation ends.
excludeJobs	**Set<String>**	Yes	No	The list of jobs to exclude. Client must also set the IGNJOBRSV reservation flag. Otherwise, results are undefined. Used only during reservation creation.
expireDate	**Date**	No	No	The date/time when the reservation expires and vacates.
flags	**Set<ReservationFlag>**	Yes	No	The flags associated with the reservation.
globalId	**String**	No	No	Global reservation ID.

Field Name	Type	POST	PUT	Description
hostListExpression	**String**	Yes	No	The list of nodes a user can select to reserve. This may or may not be the nodes that are currently allocated to this reservation. Note: Either hostListExpression or taskCount must be set to create a reservation.
idPrefix	**String**	Yes	No	The user-specified prefix for this reservation. If provided, Moab combines the idPrefix with an integer, and the combination is the unique identifier for this reservation.
isActive	**Boolean**	No	No	State whether or not this reservation is currently active.
isTracked	**Boolean**	No	No	States whether reservation resource usage is tracked.
label	**String**	Yes	No	When a label is assigned to a reservation, the reservation can then be referenced by that label as well as by the reservation name.
maxTasks	**Integer**	No	No	The maximum number of tasks for this reservation.
messages	**Set<MessageVersion1>**	No	No	Messages for the reservation.

Field Name	Type	POST	PUT	Description
owner	EmbeddedCredential	Yes	No	The owner of the reservation
partitionId	**String**	Yes	No	The ID of the partition this reservation is for.
profile	**String**	Yes	No	The profile that this reservation is using. A profile is a specification of attributes that all reservations share. Used only during reservation creation.
requirements	ReservationRequirement	Yes	No	The reservation's requirements.
reservationGroup	**String**	Yes	No	The reservation group to which the reservation belongs.
resources	**Map<String, Integer>**	Yes	No	The reservation's resources. This field is a map, where the key is PROCS, MEM DISK, SWAP, or one or more user-defined keys.
startDate	**Date**	Yes	No	The start time for the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation starts.
statistics	ReservationStatistics	No	No	The reservation's statistical information.
subType	**String**	Yes	No	The reservation sub-type.

Field Name	Type	POST	PUT	Description
taskCount	**Integer**	No	No	The number of tasks that must be allocated to satisfy the reservation request. Note: Either hostListExpression or taskCount must be set to create a reservation.
trigger	**Trigger**	Yes	No	Trigger for reservation. Used only during reservation creation.
triggerIds	**Set<String>**	No	No	The IDs of the triggers attached to this reservation.
uniqueIndex	**String**	No	No	The globally-unique reservation index.
variables	**Map<String, Map>**	Yes	Yes	The set of variables for this reservation.

AclRule

This class represents a rule that can be in Moab's access control list (ACL) mechanism.

The basic AclRule information is the object's name and type. The type directly maps to an AclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	POST	PUT	Description
affinity	AclAffinity	No	Yes	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	ComparisonOperator	No	Yes	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
type	AclType	No	Yes	The type of the object that is being granted (or denied) access.
value	**String**	No	Yes	The name of the object that is being granted (or denied) access.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.

Value	Description
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

AclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
PROC	Not supported
QTIME	Not supported
QUEUE	Not supported

Value	Description
RACK	Not supported
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

DomainProxyVersion1

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The id of the object.

ReservationFlag

The flag types of a reservation.

Value	Description
ALLOWJOBOVERLAP	Allows jobs to overlap this Reservation, but not start during it (unless they have ACL access).
APPLYPROFRESOURCES	Only apply resource allocation info from profile.
DEADLINE	Reservation should be scheduled against a deadline.
IGNIDLEJOBS	Ignore idle job reservations.
IGNJOBRSV	Ignore job reservations, but not user or other reservations.
CHARGE	Charge the idle cycles in the accounting manager.

Value	Description
NOVMMIGRATIONS	Override the VM Migration Policy and don't migrate VMs that overlap this reservation.
OWNERPREEMPTIGNOREMINTIME	Owner ignores preemptmintime for this reservation.
PROVISION	Reservation should be capable of provisioning.
NOACLOVERLAP	Reservation will not look at ACLs to overlap job (when using exclusive).
ADVRES	If set, the reservation is created in advance of needing it.
ADVRESJOBDESTROY	Cancel any jobs associated with the reservation when it is released.
ALLOWGRID	The reservation is set up for use in a grid environment.
ALLOWPRSV	Personal reservations can be created within the space of this standing reservation (and ONLY this standing reservation). By default, when a standing reservation is given the flag ALLOWPRSV, it is given the ACL rule USER==ALL+ allowing all jobs and all users access.
BYNAME	Reservation only allows access to jobs that meet reservation ACLs and explicitly request the resources of this reservation using the job ADVRES flag.
DEDICATEDNODE	If set, only one active reservation is allowed on a node.
OWNEREXCLUSIVEBF	When an owner job is idle, other jobs are not allowed to backfill.
DEDICATEDRESOURCE	The reservation is only placed on resources that are not reserved by any other reservation, including jobs and other reservations.
EXCLUDEJOBS	Makes a reservation job exclusive, where only one job can run in the reservation.
ENDTRIGHASFIRED	A trigger has finished firing.
ENFORCENODESET	Enforce node sets when creating reservation.

Value	Description
EXCLUDEALLBUTSB	Reservation only shares resources with sandboxes.
EXCLUDEMYGROUP	Exclude reservations within the same group.
IGNRSV	Forces the reservation onto nodes regardless of whether there are other reservations currently residing on the nodes.
IGNSTATE	Request ignores existing resource reservations, allowing the reservation to be forced onto available resources even if this conflicts with other reservations.
ISACTIVE	If set, the reservation is currently active.
ISCLOSED	If set, the reservation is closed.
ISGLOBAL	If set the reservation applies to all resources.
OWNERPREEMPT	The owner of the reservation is given preemptor status for resources contained in the reservation.
PARENTLOCK	The reservation can only be destroyed by destroying its parent.
PREEMPTEE	The reservation is preemptible.
PLACEHOLDER	The reservation is a placeholder for resources.
PRSV	The reservation is a non-adminstrator, non-standing reservation, user-created reservation.
REQFULL	The reservation will fail if all resources requested cannot be allocated.
SCHEDULEVCRSV	The reservation was created as part of a schedule VC command. This pertains to reservations creating while scheduling MWS Services, and these are filtered from the MWS output of reservations.
SINGLEUSE	The reservation is automatically removed after completion of the first job to use the reserved resources.

Value	Description
SPACEFLEX	The reservation is allowed to adjust resources allocated over time in an attempt to optimize resource utilization.
STANDINGRSV	If set, the reservation was created by a standing reservation instance.
STATIC	Makes a reservation ineligible to modified or canceled by an administrator.
SYSTEMJOB	The reservation was created by a system job.
TIMEFLEX	The reservation is allowed to adjust the reserved time frame in an attempt to optimize resource utilization.
TRIGHASFIRED	The reservation has one or more triggers that have fired on it.
WASACTIVE	The reservation was previously active.
EVACVMS	Evacuate virtual machines on the node when the reservation starts.

MessageVersion1

Field Name	Type	POST	PUT	Description
author	**String**	No	No	The author of the message.
creationTime	**Date**	No	No	The time the message was created in epoch time.
expireTime	**Date**	No	No	The time the message will be deleted in epoch time.
index	**Integer**	No	No	The index of the message relative to other messages in Moab's memory.
message	**String**	No	Yes	The comment information itself.
messageCount	**Integer**	No	No	The number of times this message has been displayed.
priority	**Double**	No	No	An optional priority that can be attached to the comment.

EmbeddedCredential

Field Name	Type	POST	PUT	Description
name	**String**	No	No	
type	**CredentialType**	No	No	

CredentialType

Value	Description
USER	
GROUP	
ACCOUNT	
CLASS	
QOS	
NOT_SPECIFIED	

ReservationRequirement

Represents all the types of requirements a user can request while creating a reservation.

Field Name	Type	POST	PUT	Description
architecture	**String**	Yes	No	Required architecture.
featureList	**Set<String>**	Yes	No	The list of features required for this reservation.
featureMode	**String**	No	No	Required feature mode.
memory	**Integer**	Yes	No	Required node memory, in MB.

Field Name	Type	POST	PUT	Description
nodeCount	**Integer**	No	No	Required number of nodes.
nodeIds	**Set<String>**	No	No	The list of node IDs required for this reservation.
os	**String**	Yes	No	Required Operating System.
taskCount	**Integer**	Yes	No	Required task count.

ReservationStatistics

Represents some basic statistical information that is kept about the usage of reservations. All metrics that are kept track relate to processor-seconds usage.

Field Name	Type	POST	PUT	Description
caps	**Long**	No	No	The current active processor-seconds in the last reported iteration.
cips	**Long**	No	No	The current idle processor-seconds in the last reported iteration.
taps	**Long**	No	No	The total active processor-seconds over the life of the reservation.
tips	**Long**	No	No	The total idle processor-seconds over the life of the reservation.

Trigger

Field Name	Type	POST	PUT	Description
id	**String**	No	No	Trigger id - internal ID used by moab to track triggers

Field Name	Type	POST	PUT	Description
action	**String**	No	No	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	No	No	
blockTime	**Date**	No	No	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	No	No	
eventType	**TriggerEventType**	No	No	
expireTime	**Date**	No	No	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	No	No	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	No	No	
interval	**Boolean**	No	No	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	No	No	Specifies the number of times Action will be attempted before the trigger is designated a failure.

Field Name	Type	POST	PUT	Description
multiFire	**Boolean**	No	No	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	No	No	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length
objectId	**String**	No	No	The ID of the object which this is attached to.
objectType	**String**	No	No	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	No	No	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	No	No	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	No	No	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	No	No	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.

Field Name	Type	POST	PUT	Description
sets	**String**	No	No	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	No	No	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	No	No	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	No	No	The type of the trigger.
unsets	**String**	No	No	Variable this trigger destroys upon success or failure.

TriggerActionType

This enumeration specifies the action type of a trigger.

Value	Description
CANCEL	Only apply to reservation triggers.
CHANGE_PARAM	Run changeparam (NOT PERSISTENT).

Value	Description
JOB_ PREEMPT	Indicates that the trigger should preempt all jobs currently allocating resources assigned to the trigger's parent object. Only apply to reservation triggers.
MAIL	Sends an e-mail.
INTERNAL	Modifies an object internally in Moab. This can be used to set a job hold for example.
EXEC	Execute the trigger action. Typically used to run a script.
MODIFY	Can modify object that trigger is attached to.
QUERY	
RESERVE	
SUBMIT	

TriggerEventType

This enumeration specifies the event type of a trigger.

Value	Description
CANCEL	
CHECKPOINT	
CREATE	
END	
EPOCH	
FAIL	
HOLD	
MIGRATE	

Value	Description
MODIFY	
PREEMPT	
STANDING	
START	
THRESHOLD	
DISCOVER	
LOGROLL	

TriggerFlag

This enumeration specifies a flag belonging to a trigger.

Value	Description
ATTACH_ERROR	If the trigger outputs anything to stderr, Moab will attach this as a message to the trigger object.
CLEANUP	If the trigger is still running when the parent object completes or is canceled, the trigger will be killed.
CHECKPOINT	Moab should always checkpoint this trigger. See Checkpointing a Trigger in the Moab Workload Manager documentation for more information.
GLOBAL_VARS	The trigger will look in the name space of all nodes with the globalvars flag in addition to its own name space. A specific node to search can be specified using the following format: globalvars+node_id
INTERVAL	Trigger is periodic.
MULTIFIRE	Trigger can fire multiple times.
OBJECT_XML_STDIN	Trigger passes its parent's object XML information into the trigger's stdin. This only works for exec triggers with reservation type parents.

Value	Description
USER	The trigger will execute under the user ID of the object's owner. If the parent object is sched, the user to run under may be explicitly specified using the format user+<username>, for example flags=user+john:
GLOBAL_ TRIGGER	The trigger will be (or was) inserted into the global trigger list.
ASYNCHRONOUS	An asynchronous trigger.
LEAVE_FILES	Do not remove stderr and stdout files.
PROBE	The trigger's stdout will be monitored.
PROBE_ALL	The trigger's stdout will be monitored.
GENERIC_ SYSTEM_JOB	The trigger belongs to a generic system job (for checkpointing).
REMOVE_STD_ FILES	The trigger will delete stdout/stderr files after it has been reset.
RESET_ON_ MODIFY	The trigger resets if the object it is attached to is modified, even if multifire is not set.
SOFT_KILL	By default, a `SIGKILL` (kill -9) signal is sent to the kill the script when a trigger times out. This flag will instead send a `SIGTERM` (kill -15) signal to kill the script. The `SIGTERM` signal will allow the script to trap the signal so that the script can clean up any residual information on the system (instead of just dying, as with the `SIGKILL` signal). NOTE: A timed-out trigger will only receive one kill signal. This means that if you specify this flag, a timed-out trigger will only receive the `SIGTERM` signal, and never the `SIGKILL` signal.

TriggerPeriod

This enumeration specifies the period of a trigger.

Value	Description
MINUTE	

Value	Description
HOUR	
DAY	
WEEK	
MONTH	

TriggerType

This enumeration specifies the type of the trigger.

Value	Description
generic	
elastic	

API version 2

Reservation

A reservation is the mechanism by which Moab guarantees the availability of a set of resources at a particular time. Each reservation consists of three major components: (1) a set of resources, (2) a time frame, and (3) an access control list. It is a scheduler role to ensure that the access control list is not violated during the reservation's lifetime (that is, its time frame) on the resources listed. For example, a reservation may specify that node002 is reserved for user Tom on Friday. The scheduler is thus constrained to make certain that only Tom's jobs can use node002 at any time on Friday.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of the reservation.
accountingAccount	**String**	Yes	No	Accountable Account.
accountingGroup	**String**	Yes	No	Accountable Group.
accountingQOS	**String**	Yes	No	Accountable QOS.
accountingUser	**String**	Yes	No	Accountable User.
aclRules	**Set<AclRule>**	Yes	No	The set of access control rules associated with this reservation.
allocatedNodeCount	**Integer**	No	No	The number of allocated nodes for this reservation.
allocatedNodes	**Set<DomainProxyVersion1>**	No	No	The nodes allocated to the reservation.
allocatedProcessorCount	**Integer**	No	No	The number of allocated processors.
allocatedTaskCount	**Integer**	No	No	The number of allocated tasks.

Field Name	Type	POST	PUT	Description
comments	**String**	Yes	No	Reservation's comments or description.
creationDate	**Date**	No	No	Creation date. Automatically set by Moab when a user creates the reservation.
duration	**Long**	Yes	No	The duration of the reservation (in seconds).
endDate	**Date**	Yes	No	The end date of the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation ends.
excludeJobs	**Set<String>**	Yes	No	The list of jobs to exclude. Client must also set the IGNJOBRSV reservation flag. Otherwise, results are undefined. Used only during reservation creation.
expireDate	**Date**	No	No	The date/time when the reservation expires and vacates.
flags	**Set<ReservationFlag>**	Yes	No	The flags associated with the reservation.
globalId	**String**	No	No	Global reservation ID.

Field Name	Type	POST	PUT	Description
hostListExpression	**String**	Yes	No	The list of nodes a user can select to reserve. This may or may not be the nodes that are currently allocated to this reservation. Note: Either hostListExpression or taskCount must be set to create a reservation.
idPrefix	**String**	Yes	No	The user-specified prefix for this reservation. If provided, Moab combines the idPrefix with an integer, and the combination is the unique identifier for this reservation.
isActive	**Boolean**	No	No	State whether or not this reservation is currently active.
isTracked	**Boolean**	No	No	States whether reservation resource usage is tracked.
label	**String**	Yes	No	When a label is assigned to a reservation, the reservation can then be referenced by that label as well as by the reservation name.
maxTasks	**Integer**	No	No	The maximum number of tasks for this reservation.
messages	**Set<MessageVersion1>**	No	No	Messages for the reservation.

Field Name	Type	POST	PUT	Description
owner	EmbeddedCredential	Yes	No	The owner of the reservation
partitionId	**String**	Yes	No	The ID of the partition this reservation is for.
profile	**String**	Yes	No	The profile that this reservation is using. A profile is a specification of attributes that all reservations share. Used only during reservation creation.
requirements	ReservationRequirement	Yes	No	The reservation's requirements.
reservationGroup	**String**	Yes	No	The reservation group to which the reservation belongs.
resources	**Map<String, Integer>**	Yes	No	The reservation's resources. This field is a map, where the key is PROCS, MEM DISK, SWAP, or one or more user-defined keys.
startDate	**Date**	Yes	No	The start time for the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation starts.
statistics	ReservationStatistics	No	No	The reservation's statistical information.
subType	**String**	Yes	No	The reservation sub-type.

Field Name	Type	POST	PUT	Description
taskCount	**Integer**	No	No	The number of tasks that must be allocated to satisfy the reservation request. Note: Either hostListExpression or taskCount must be set to create a reservation.
trigger	**Trigger**	Yes	No	Trigger for reservation. Used only during reservation creation.
triggerIds	**Set<String>**	No	No	The IDs of the triggers attached to this reservation.
uniqueIndex	**String**	No	No	The globally-unique reservation index.
variables	**Map<String, Map>**	Yes	Yes	The set of variables for this reservation.

AclRule

This class represents a rule that can be in Moab's access control list (ACL) mechanism.

The basic AclRule information is the object's name and type. The type directly maps to an AclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	POST	PUT	Description
affinity	**AclAffinity**	No	Yes	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	**ComparisonOperator**	No	Yes	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
type	**AclType**	No	Yes	The type of the object that is being granted (or denied) access.
value	**String**	No	Yes	The name of the object that is being granted (or denied) access.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.

Value	Description
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

AclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
PROC	Not supported
QTIME	Not supported
QUEUE	Not supported

Value	Description
RACK	Not supported
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

DomainProxyVersion1

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The id of the object.

ReservationFlag

The flag types of a reservation.

Value	Description
ALLOWJOBOVERLAP	Allows jobs to overlap this Reservation, but not start during it (unless they have ACL access).
APPLYPROFRESOURCES	Only apply resource allocation info from profile.
DEADLINE	Reservation should be scheduled against a deadline.
IGNIDLEJOBS	Ignore idle job reservations.
IGNJOBRSV	Ignore job reservations, but not user or other reservations.
CHARGE	Charge the idle cycles in the accounting manager.

Value	Description
NOVMMIGRATIONS	Override the VM Migration Policy and don't migrate VMs that overlap this reservation.
OWNERPREEMPTIGNOREMINTIME	Owner ignores preemptmintime for this reservation.
PROVISION	Reservation should be capable of provisioning.
NOACLOVERLAP	Reservation will not look at ACLs to overlap job (when using exclusive).
ADVRES	If set, the reservation is created in advance of needing it.
ADVRESJOBDESTROY	Cancel any jobs associated with the reservation when it is released.
ALLOWGRID	The reservation is set up for use in a grid environment.
ALLOWPRSV	Personal reservations can be created within the space of this standing reservation (and ONLY this standing reservation). By default, when a standing reservation is given the flag ALLOWPRSV, it is given the ACL rule USER==ALL+ allowing all jobs and all users access.
BYNAME	Reservation only allows access to jobs that meet reservation ACLs and explicitly request the resources of this reservation using the job ADVRES flag.
DEDICATEDNODE	If set, only one active reservation is allowed on a node.
OWNEREXCLUSIVEBF	When an owner job is idle, other jobs are not allowed to backfill.
DEDICATEDRESOURCE	The reservation is only placed on resources that are not reserved by any other reservation, including jobs and other reservations.
EXCLUDEJOBS	Makes a reservation job exclusive, where only one job can run in the reservation.
ENDTRIGHASFIRED	A trigger has finished firing.
ENFORCENODESET	Enforce node sets when creating reservation.

Value	Description
EXCLUDEALLBUTSB	Reservation only shares resources with sandboxes.
EXCLUDEMYGROUP	Exclude reservations within the same group.
IGNRSV	Forces the reservation onto nodes regardless of whether there are other reservations currently residing on the nodes.
IGNSTATE	Request ignores existing resource reservations, allowing the reservation to be forced onto available resources even if this conflicts with other reservations.
ISACTIVE	If set, the reservation is currently active.
ISCLOSED	If set, the reservation is closed.
ISGLOBAL	If set the reservation applies to all resources.
OWNERPREEMPT	The owner of the reservation is given preemptor status for resources contained in the reservation.
PARENTLOCK	The reservation can only be destroyed by destroying its parent.
PREEMPTEE	The reservation is preemptible.
PLACEHOLDER	The reservation is a placeholder for resources.
PRSV	The reservation is a non-adminstrator, non-standing reservation, user-created reservation.
REQFULL	The reservation will fail if all resources requested cannot be allocated.
SCHEDULEVCRSV	The reservation was created as part of a schedule VC command. This pertains to reservations creating while scheduling MWS Services, and these are filtered from the MWS output of reservations.
SINGLEUSE	The reservation is automatically removed after completion of the first job to use the reserved resources.

Value	Description
SPACEFLEX	The reservation is allowed to adjust resources allocated over time in an attempt to optimize resource utilization.
STANDINGRSV	If set, the reservation was created by a standing reservation instance.
STATIC	Makes a reservation ineligible to modified or canceled by an administrator.
SYSTEMJOB	The reservation was created by a system job.
TIMEFLEX	The reservation is allowed to adjust the reserved time frame in an attempt to optimize resource utilization.
TRIGHASFIRED	The reservation has one or more triggers that have fired on it.
WASACTIVE	The reservation was previously active.
EVACVMS	Evacuate virtual machines on the node when the reservation starts.

MessageVersion1

Field Name	Type	POST	PUT	Description
author	**String**	No	No	The author of the message.
creationTime	**Date**	No	No	The time the message was created in epoch time.
expireTime	**Date**	No	No	The time the message will be deleted in epoch time.
index	**Integer**	No	No	The index of the message relative to other messages in Moab's memory.
message	**String**	No	Yes	The comment information itself.
messageCount	**Integer**	No	No	The number of times this message has been displayed.
priority	**Double**	No	No	An optional priority that can be attached to the comment.

EmbeddedCredential

Field Name	Type	POST	PUT	Description
name	**String**	No	No	
type	**CredentialType**	No	No	

CredentialType

Value	Description
USER	
GROUP	
ACCOUNT	
CLASS	
QOS	
NOT_SPECIFIED	

ReservationRequirement

Represents all the types of requirements a user can request while creating a reservation.

Field Name	Type	POST	PUT	Description
architecture	**String**	Yes	No	Required architecture.
featureList	**Set<String>**	Yes	No	The list of features required for this reservation.
featureMode	**String**	No	No	Required feature mode.
memory	**Integer**	Yes	No	Required node memory, in MB.

Field Name	Type	POST	PUT	Description
nodeCount	**Integer**	No	No	Required number of nodes.
nodeIds	**Set<String>**	No	No	The list of node IDs required for this reservation.
os	**String**	Yes	No	Required Operating System.
taskCount	**Integer**	Yes	No	Required task count.

ReservationStatistics

Represents some basic statistical information that is kept about the usage of reservations. All metrics that are kept track relate to processor-seconds usage.

Field Name	Type	POST	PUT	Description
caps	**Long**	No	No	The current active processor-seconds in the last reported iteration.
cips	**Long**	No	No	The current idle processor-seconds in the last reported iteration.
taps	**Long**	No	No	The total active processor-seconds over the life of the reservation.
tips	**Long**	No	No	The total idle processor-seconds over the life of the reservation.

Trigger

Field Name	Type	POST	PUT	Description
id	**String**	No	No	Trigger id - internal ID used by moab to track triggers

Field Name	Type	POST	PUT	Description
action	**String**	No	No	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	No	No	
blockTime	**Date**	No	No	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	No	No	
eventType	**TriggerEventType**	No	No	
expireTime	**Date**	No	No	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	No	No	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	No	No	
interval	**Boolean**	No	No	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	No	No	Specifies the number of times Action will be attempted before the trigger is designated a failure.

Field Name	Type	POST	PUT	Description
multiFire	**Boolean**	No	No	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	No	No	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length
objectId	**String**	No	No	The ID of the object which this is attached to.
objectType	**String**	No	No	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	No	No	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	No	No	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	No	No	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	No	No	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.

Field Name	Type	POST	PUT	Description
sets	**String**	No	No	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	No	No	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	No	No	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	No	No	The type of the trigger.
unsets	**String**	No	No	Variable this trigger destroys upon success or failure.

TriggerActionType

This enumeration specifies the action type of a trigger.

Value	Description
CANCEL	Only apply to reservation triggers.
CHANGE_PARAM	Run changeparam (NOT PERSISTENT).

Value	Description
JOB_ PREEMPT	Indicates that the trigger should preempt all jobs currently allocating resources assigned to the trigger's parent object. Only apply to reservation triggers.
MAIL	Sends an e-mail.
INTERNAL	Modifies an object internally in Moab. This can be used to set a job hold for example.
EXEC	Execute the trigger action. Typically used to run a script.
MODIFY	Can modify object that trigger is attached to.
QUERY	
RESERVE	
SUBMIT	

TriggerEventType

This enumeration specifies the event type of a trigger.

Value	Description
CANCEL	
CHECKPOINT	
CREATE	
END	
EPOCH	
FAIL	
HOLD	
MIGRATE	

Value	Description
MODIFY	
PREEMPT	
STANDING	
START	
THRESHOLD	
DISCOVER	
LOGROLL	

TriggerFlag

This enumeration specifies a flag belonging to a trigger.

Value	Description
ATTACH_ERROR	If the trigger outputs anything to stderr, Moab will attach this as a message to the trigger object.
CLEANUP	If the trigger is still running when the parent object completes or is canceled, the trigger will be killed.
CHECKPOINT	Moab should always checkpoint this trigger. See Checkpointing a Trigger in the Moab Workload Manager documentation for more information.
GLOBAL_VARS	The trigger will look in the name space of all nodes with the globalvars flag in addition to its own name space. A specific node to search can be specified using the following format: globalvars+node_id
INTERVAL	Trigger is periodic.
MULTIFIRE	Trigger can fire multiple times.
OBJECT_XML_STDIN	Trigger passes its parent's object XML information into the trigger's stdin. This only works for exec triggers with reservation type parents.

Value	Description
USER	The trigger will execute under the user ID of the object's owner. If the parent object is sched, the user to run under may be explicitly specified using the format user+<username>, for example flags=user+john:
GLOBAL_ TRIGGER	The trigger will be (or was) inserted into the global trigger list.
ASYNCHRONOUS	An asynchronous trigger.
LEAVE_FILES	Do not remove stderr and stdout files.
PROBE	The trigger's stdout will be monitored.
PROBE_ALL	The trigger's stdout will be monitored.
GENERIC_ SYSTEM_JOB	The trigger belongs to a generic system job (for checkpointing).
REMOVE_STD_ FILES	The trigger will delete stdout/stderr files after it has been reset.
RESET_ON_ MODIFY	The trigger resets if the object it is attached to is modified, even if multifire is not set.
SOFT_KILL	By default, a `SIGKILL` (kill -9) signal is sent to the kill the script when a trigger times out. This flag will instead send a `SIGTERM` (kill -15) signal to kill the script. The `SIGTERM` signal will allow the script to trap the signal so that the script can clean up any residual information on the system (instead of just dying, as with the `SIGKILL` signal). NOTE: A timed-out trigger will only receive one kill signal. This means that if you specify this flag, a timed-out trigger will only receive the `SIGTERM` signal, and never the `SIGKILL` signal.

TriggerPeriod

This enumeration specifies the period of a trigger.

Value	Description
MINUTE	

Value	Description
HOUR	
DAY	
WEEK	
MONTH	

TriggerType

This enumeration specifies the type of the trigger.

Value	Description
generic	
elastic	

Related Topics

- Reservations

Fields: Resource Types

See the associated Resource Types resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`resource-types`	Permissions
Hooks filename	`resource-types.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

ResourceType

Represents a resource type in Moab Workload Manager.

Field Name	Type	Description
id	**String**	The unique ID of this resource type.

API version 2

ResourceType

Represents a resource type in Moab Workload Manager.

Field Name	Type	Description
id	**String**	The unique ID of this resource type.

Related Topics

- Resource Types

Fields: Roles

See the associated Roles resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`roles`	Permissions
Hooks filename	`roles.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Role

A role defines a set of permissions that are based on the proxy-user. If no proxy user is specified then access to objects in MWS are limited to its application permissions. For example if the application has permission to update all resources in MWS and no proxy-user is specified in the request then the request can access all resources in MWS.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
description	**String**	Yes	Yes	The role description.
name	**String**	Yes	Yes	The unique human-readable name of this role. Required during POST.
permissions	**List<Permission>**	Yes	Yes	The set of permissions enforced based on the proxy-user.
scope	**PrivilegeScope**	No	No	

Permission

Represents a permission

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
action	**String**	No	No	The action that can be performed on the resource.
administrator	**Boolean**	No	No	If true, grants full rights over the given resource for the given action. For example, if resource is "jobs" and action is "update" and administrator is true, then this permission allows the user to update any job, not just jobs owned by the user.
description	**String**	No	No	A description of this permission.

Field Name	Type	POST	PUT	Description
fieldPath	**String**	No	No	Field level ACL control, if null or '*', all fields are accessible, otherwise requests must match dot delimited path. Currently only checked when doing writable actions. Example - attributes.*: create|update
label	**String**	No	No	A human readable label for this permission.
resource	**String**	No	No	The resource the permission applies to.
resourceFilter	**Map<String, Map>**	No	No	A map used to limit which resource instances this permission applies to. If this is null then the permission will apply to all instances of the resource. For api permissions the filter uses mongo query syntax.
scope	**PrivilegeScope**	No	No	Whether this permission applies to the principal's tenant-associated resources or globally
type	**String**	No	No	The type of the permission. Only 'api' type permissions are enforced.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

API version 2

Role

A role defines a set of permissions that are based on the proxy-user. If no proxy user is specified then access to objects in MWS are limited to its application permissions. For example if the application has permission to update all resources in MWS and no proxy-user is specified in the request then the request can access all resources in MWS.

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
description	**String**	Yes	Yes	The role description.
name	**String**	Yes	Yes	The unique human-readable name of this role. Required during POST.
permissions	**List<Permission>**	Yes	Yes	The set of permissions enforced based on the proxy-user.
scope	**PrivilegeScope**	No	No	

Permission

Represents a permission

Field Name	Type	POST	PUT	Description
id	**String**	No	No	The unique ID of this role.
action	**String**	No	No	The action that can be performed on the resource.
administrator	**Boolean**	No	No	If true, grants full rights over the given resource for the given action. For example, if resource is "jobs" and action is "update" and administrator is true, then this permission allows the user to update any job, not just jobs owned by the user.
description	**String**	No	No	A description of this permission.

Field Name	Type	POST	PUT	Description
fieldPath	**String**	No	No	Field level ACL control, if null or '*', all fields are accessible, otherwise requests must match dot delimited path. Currently only checked when doing writable actions. Example - attributes.*: create|update
label	**String**	No	No	A human readable label for this permission.
resource	**String**	No	No	The resource the permission applies to.
resourceFilter	**Map<String, Map>**	No	No	A map used to limit which resource instances this permission applies to. If this is null then the permission will apply to all instances of the resource. For api permissions the filter uses mongo query syntax.
scope	**PrivilegeScope**	No	No	Whether this permission applies to the principal's tenant-associated resources or globally
type	**String**	No	No	The type of the permission. Only 'api' type permissions are enforced.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

Related Topics

- Roles

Fields: Report Samples

See the associated Reports resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`reports/samples`	Permissions
Hooks filename	`reports.samples.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

Sample

A single snapshot of system state. It can contain all the same information as Datapoint.data in the sample's data field.

Field Name	Type	POST	Description
id	**Long**	No	
agent	**String**	No	A unique identifier for the agent that recorded this sample.
data	**Map<String, Map>**	No	Arbitrary data that was recorded for this sample. Defaults to an empty object if none is supplied.
timestamp	**Date**	No	The date and time at which this sample was recorded. Defaults to the current date if none is supplied.

API version 2

Sample

A single snapshot of system state. It can contain all the same information as Datapoint.data in the sample's data field.

Field Name	Type	POST	Description
id	**Long**	No	
agent	**String**	No	A unique identifier for the agent that recorded this sample.
data	**Map<String, Map>**	No	Arbitrary data that was recorded for this sample. Defaults to an empty object if none is supplied.
timestamp	**Date**	No	The date and time at which this sample was recorded. Defaults to the current date if none is supplied.

Related Topics

- Reports

Fields: Standing Reservations

See the associated Standing Reservations resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`standing-reservations`	Permissions
Hooks filename	`standing-reservations.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	No	Distinct

API version 3

StandingReservation

This class represents a standing reservation.

A standing reservation is any reservation that is not a one-time reservation. This includes reservations that recur every day or every week, or infinite reservations.

Field Name	Type	Description
id	**String**	The unique ID of the standing reservation.
access	**ReservationAccess**	If set to ReservationAccess.SHARED, allows a standing reservation to use resources already allocated to other non-job reservations. Otherwise, these other reservations block resource access.
accounts	**Set<String>**	Specifies that jobs with the associated accounts may use the resources contained within this reservation.
aclRules	**Set<AclRule>**	The set of access control rules associated with this standing reservation.
chargeAccount	**String**	Specifies the account to which Moab will charge all idle cycles within the reservation (via the accounting manager).
chargeUser	**String**	Specifies the user to which Moab will charge all idle cycles within the reservation (via the accounting manager). Must be used in conjunction with chargeAccount
classes	**Set<String>**	Specifies that jobs with the associated classes/queues may use the resources contained within this reservation.
clusters	**Set<String>**	Specifies that jobs originating within the listed clusters may use the resources contained within this reservation.
comment	**String**	Specifies a descriptive message associated with the standing reservation and all child reservations

Field Name	Type	Description
days	**Set<String>**	Specifies which days of the week the standing reservation is active. Valid values are Mon, Tue, Wed, Thu, Fri, Sat, Sun, or [ALL].
depth	**Integer**	Specifies the depth of standing reservations to be created, starting at depth 0 (one per period).
disabled	**Boolean**	Specifies if the standing reservation should no longer spawn child reservations.
endOffset	**Long**	The ending offset, in seconds, from the beginning of the current period (DAY or WEEK), for this standing reservation. See examples at startOffset.
flags	**Set<ReservationFlag>**	Specifies special reservation attributes.
groups	**Set<String>**	Specifies the groups allowed access to this standing reservation.
hosts	**Set<String>**	Specifies the set of hosts that the scheduler can search for resources to satisfy the reservation. If specified using the class:X format, Moab only selects hosts that support the specified class. If TASKCOUNT is also specified, only TASKCOUNT tasks are reserved. Otherwise, all matching hosts are reserved.
jobAttributes	**Set<JobFlag>**	Specifies job attributes that grant a job access to the reservation. Values can be specified with a != assignment to only allow jobs NOT requesting a certain feature inside the reservation.
maxJob	**Integer**	Specifies the maximum number of jobs that can run in the reservation.
maxTime	**Integer**	Specifies the maximum time for jobs allowable. Can be used with affinity to attract jobs with same maxTime.
messages	**Set<String>**	Messages associated with the reservation.

Field Name	Type	Description
nodeFeatures	**Set<String>**	Specifies the required node features for nodes that are part of the standing reservation.
os	**String**	Specifies the operating system that should be in place during the reservation. Moab provisions this OS at reservation start and restores the original OS at reservation completion.
owner	**EmbeddedCredential**	Specifies the owner of the reservation. Setting ownership for a reservation grants the user management privileges, including the power to release it. Setting a user as the owner of a reservation gives that user privileges to query and release the reservation. For sandbox reservations, sandboxes are applied to a specific peer only if owner is set to CLUSTER:<PEERNAME>
partition	**String**	Specifies the partition in which to create the standing reservation. Defaults to ALL.
period	**TimeWindow**	Period of the Standing reservation. Defaults to TimeWindow.DAY.
procLimit	**IntLimit**	Specifies the processor limit for jobs requesting access to this standing reservation.
psLimit	**IntLimit**	Specifies the processor-second limit for jobs requesting access to this standing reservation.
qoses	**Set<String>**	Specifies that jobs with the listed QoS names can access the reserved resources.
reservationAccessList	**Set<Reservation>**	A list of reservations to which the specified reservation has access.
reservationGroup	**String**	The group of the reservation.

Field Name	Type	Description
resources	**Map<String, Integer>**	Specifies what resources constitute a single standing reservation task. (Each task must be able to obtain all of its resources as an atomic unit on a single node.) Supported resources currently include the following: • PROCS (number of processors) • MEM (real memory in MB) • DISK (local disk in MB) • SWAP (virtual memory in MB)
rollbackOffset	**Integer**	Specifies the minimum time in the future at which the reservation may start. This offset is rolling meaning the start time of the reservation will continuously roll back into the future to maintain this offset. Rollback offsets are a good way of providing guaranteed resource access to users under the conditions that they must commit their resources in the future or lose dedicated access. See QoS Credential in the Moab Workload Manager documentation for more information on quality of service and service level agreements.
startOffset	**Long**	The starting offset, in seconds, from the beginning of the current period (DAY or WEEK), for this standing reservation. If period is DAY, the offset is from midnight (00:00) of the current day. If period is WEEK, the offset is from midnight Sunday of the current week. Example 1: For a standing reservation that begins at 9:00 and ends at 17:00 every day, period is DAY, startOffset is 32400 (9*60*60), and endOffset is 61200 (17*60*60). Example 2: For a standing reservation that begins at 9:00 Monday and ends at 17:00 Friday every week, period is WEEK, startOffset is 118800 ((24+9) *60*60), and endOffset is 493200 (((5*24)+17) *60*60).

Field Name	Type	Description
taskCount	**Integer**	Specifies how many tasks should be reserved for the reservation Default is 0 (unlimited tasks).
tasksPerNode	**Integer**	Specifies the minimum number of tasks per node that must be available on eligible nodes. Default is 0 (no TPN constraint)
timeLimit	**Integer**	Specifies the maximum allowed overlap between the standing reservation and a job requesting resource access. Default is null (-1 in moab)
triggers	**Set<Trigger>**	Triggers associated with the reservation.
type	**String**	The type of the reservation.
users	**Set<String>**	Specifies which users have access to the resources reserved by this reservation.

ReservationAccess

The access type of a standing reservation. If set to SHARED, allows a standing reservation to use resources already allocated to other non-job reservations. Otherwise, these other reservations block resource access.

Value	Description
DEDICATED	
SHARED	

AclRule

This class represents a rule that can be in Moab's access control list (ACL) mechanism.

The basic AclRule information is the object's name and type. The type directly maps to an AclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	Description
affinity	**AclAffinity**	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	**ComparisonOperator**	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
type	**AclType**	The type of the object that is being granted (or denied) access.
value	**String**	The name of the object that is being granted (or denied) access.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.

Value	Description
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

AclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User

Value	Description
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
PROC	Not supported
QTIME	Not supported
QUEUE	Not supported
RACK	Not supported

Value	Description
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

ReservationFlag

The flag types of a reservation.

Value	Description
ALLOWJOBOVERLAP	Allows jobs to overlap this Reservation, but not start during it (unless they have ACL access).
APPLYPROFRESOURCES	Only apply resource allocation info from profile.
DEADLINE	Reservation should be scheduled against a deadline.
IGNIDLEJOBS	Ignore idle job reservations.
IGNJOBRSV	Ignore job reservations, but not user or other reservations.
CHARGE	Charge the idle cycles in the accounting manager.
NOVMMIGRATIONS	Override the VM Migration Policy and don't migrate VMs that overlap this reservation.
OWNERPREEMPTIGNOREMINTIME	Owner ignores preemptmintime for this reservation.
PROVISION	Reservation should be capable of provisioning.
NOACLOVERLAP	Reservation will not look at ACLs to overlap job (when using exclusive).

Value	Description
ADVRES	If set, the reservation is created in advance of needing it.
ADVRESJOBDESTROY	Cancel any jobs associated with the reservation when it is released.
ALLOWGRID	The reservation is set up for use in a grid environment.
ALLOWPRSV	Personal reservations can be created within the space of this standing reservation (and ONLY this standing reservation). By default, when a standing reservation is given the flag ALLOWPRSV, it is given the ACL rule USER==ALL+ allowing all jobs and all users access.
BYNAME	Reservation only allows access to jobs that meet reservation ACLs and explicitly request the resources of this reservation using the job ADVRES flag.
DEDICATEDNODE	If set, only one active reservation is allowed on a node.
OWNEREXCLUSIVEBF	When an owner job is idle, other jobs are not allowed to backfill.
DEDICATEDRESOURCE	The reservation is only placed on resources that are not reserved by any other reservation, including jobs and other reservations.
EXCLUDEJOBS	Makes a reservation job exclusive, where only one job can run in the reservation.
ENDTRIGHASFIRED	A trigger has finished firing.
ENFORCENODESET	Enforce node sets when creating reservation.
EXCLUDEALLBUTSB	Reservation only shares resources with sandboxes.
EXCLUDEMYGROUP	Exclude reservations within the same group.
IGNRSV	Forces the reservation onto nodes regardless of whether there are other reservations currently residing on the nodes.

Value	Description
IGNSTATE	Request ignores existing resource reservations, allowing the reservation to be forced onto available resources even if this conflicts with other reservations.
ISACTIVE	If set, the reservation is currently active.
ISCLOSED	If set, the reservation is closed.
ISGLOBAL	If set the reservation applies to all resources.
OWNERPREEMPT	The owner of the reservation is given preemptor status for resources contained in the reservation.
PARENTLOCK	The reservation can only be destroyed by destroying its parent.
PREEMPTEE	The reservation is preemptible.
PLACEHOLDER	The reservation is a placeholder for resources.
PRSV	The reservation is a non-adminstrator, non-standing reservation, user-created reservation.
REQFULL	The reservation will fail if all resources requested cannot be allocated.
SCHEDULEVCRSV	The reservation was created as part of a schedule VC command. This pertains to reservations creating while scheduling MWS Services, and these are filtered from the MWS output of reservations.
SINGLEUSE	The reservation is automatically removed after completion of the first job to use the reserved resources.
SPACEFLEX	The reservation is allowed to adjust resources allocated over time in an attempt to optimize resource utilization.
STANDINGRSV	If set, the reservation was created by a standing reservation instance.
STATIC	Makes a reservation ineligible to modified or canceled by an administrator.

Value	Description
SYSTEMJOB	The reservation was created by a system job.
TIMEFLEX	The reservation is allowed to adjust the reserved time frame in an attempt to optimize resource utilization.
TRIGHASFIRED	The reservation has one or more triggers that have fired on it.
WASACTIVE	The reservation was previously active.
EVACVMS	Evacuate virtual machines on the node when the reservation starts.

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.
ADVRES	The job requires the use of a reservation.
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.

Value	Description
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.
INTERACTIVE	The job needs to interactive input from the user to run.
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.

Value	Description
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.

Value	Description
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

EmbeddedCredential

Field Name	Type	Description
name	**String**	
type	**CredentialType**	

CredentialType

Value	Description
USER	
GROUP	
ACCOUNT	
CLASS	
QOS	
NOT_SPECIFIED	

TimeWindow

This enumeration represents some common time windows. It can be used when for many purposes, but was created specifically for statistics.

Value	Description
MINUTE	
HOUR	
DAY	
WEEK	
MONTH	
YEAR	
INFINITY	

IntLimit

Field Name	Type	Description
qualifier	**String**	One of: • < • <= • == • >= • >
value	**Integer**	

Reservation

A reservation is the mechanism by which Moab guarantees the availability of a set of resources at a particular time. Each reservation consists of three major components: (1) a set of resources, (2) a time frame, and (3) an access control list. It is a scheduler role to ensure that the access control list is not violated during the reservation's lifetime (that is, its time frame) on the resources listed. For example, a reservation may specify that node002 is reserved for user Tom on Friday. The scheduler is thus constrained to make certain that only Tom's jobs can use node002 at any time on Friday.

Field Name	Type	Description
id	**String**	The unique ID of the reservation.
accountingAccount	**String**	Accountable Account.
accountingGroup	**String**	Accountable Group.
accountingQOS	**String**	Accountable QOS.
accountingUser	**String**	Accountable User.
aclRules	**Set<AclRule>**	The set of access control rules associated with this reservation.
allocatedNodeCount	**Integer**	The number of allocated nodes for this reservation.
allocatedNodes	**Set<DomainProxyVersion1>**	The nodes allocated to the reservation.
allocatedProcessorCount	**Integer**	The number of allocated processors.
allocatedTaskCount	**Integer**	The number of allocated tasks.
comments	**String**	Reservation's comments or description.
creationDate	**Date**	Creation date. Automatically set by Moab when a user creates the reservation.
duration	**Long**	The duration of the reservation (in seconds).

Field Name	Type	Description
endDate	**Date**	The end date of the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation ends.
excludeJobs	**Set<String>**	The list of jobs to exclude. Client must also set the IGNJOBRSV reservation flag. Otherwise, results are undefined. Used only during reservation creation.
expireDate	**Date**	The date/time when the reservation expires and vacates.
flags	**Set<ReservationFlag>**	The flags associated with the reservation.
globalId	**String**	Global reservation ID.
hostListExpression	**String**	The list of nodes a user can select to reserve. This may or may not be the nodes that are currently allocated to this reservation. Note: Either hostListExpression or taskCount must be set to create a reservation.
idPrefix	**String**	The user-specified prefix for this reservation. If provided, Moab combines the idPrefix with an integer, and the combination is the unique identifier for this reservation.
isActive	**Boolean**	State whether or not this reservation is currently active.
isTracked	**Boolean**	States whether reservation resource usage is tracked.
label	**String**	When a label is assigned to a reservation, the reservation can then be referenced by that label as well as by the reservation name.
maxTasks	**Integer**	The maximum number of tasks for this reservation.

Field Name	Type	Description
messages	Set<MessageVersion1>	Messages for the reservation.
owner	EmbeddedCredential	The owner of the reservation
partitionId	**String**	The ID of the partition this reservation is for.
profile	**String**	The profile that this reservation is using. A profile is a specification of attributes that all reservations share. Used only during reservation creation.
requirements	ReservationRequirement	The reservation's requirements.
reservationGroup	**String**	The reservation group to which the reservation belongs.
resources	**Map<String, Integer>**	The reservation's resources. This field is a map, where the key is PROCS, MEM DISK, SWAP, or one or more user-defined keys.
startDate	**Date**	The start time for the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation starts.
statistics	ReservationStatistics	The reservation's statistical information.
subType	**String**	The reservation sub-type.
taskCount	**Integer**	The number of tasks that must be allocated to satisfy the reservation request. Note: Either hostListExpression or taskCount must be set to create a reservation.
trigger	Trigger	Trigger for reservation. Used only during reservation creation.
triggerIds	**Set<String>**	The IDs of the triggers attached to this reservation.

Field Name	Type	Description
uniqueIndex	**String**	The globally-unique reservation index.
variables	**Map<String, Map>**	The set of variables for this reservation.

DomainProxyVersion1

Field Name	Type	Description
id	**String**	The id of the object.

MessageVersion1

Field Name	Type	Description
author	**String**	The author of the message.
creationTime	**Date**	The time the message was created in epoch time.
expireTime	**Date**	The time the message will be deleted in epoch time.
index	**Integer**	The index of the message relative to other messages in Moab's memory.
message	**String**	The comment information itself.
messageCount	**Integer**	The number of times this message has been displayed.
priority	**Double**	An optional priority that can be attached to the comment.

ReservationRequirement

Represents all the types of requirements a user can request while creating a reservation.

Field Name	Type	Description
architecture	**String**	Required architecture.

Field Name	Type	Description
featureList	**Set<String>**	The list of features required for this reservation.
featureMode	**String**	Required feature mode.
memory	**Integer**	Required node memory, in MB.
nodeCount	**Integer**	Required number of nodes.
nodeIds	**Set<String>**	The list of node IDs required for this reservation.
os	**String**	Required Operating System.
taskCount	**Integer**	Required task count.

ReservationStatistics

Represents some basic statistical information that is kept about the usage of reservations. All metrics that are kept track relate to processor-seconds usage.

Field Name	Type	Description
caps	**Long**	The current active processor-seconds in the last reported iteration.
cips	**Long**	The current idle processor-seconds in the last reported iteration.
taps	**Long**	The total active processor-seconds over the life of the reservation.
tips	**Long**	The total idle processor-seconds over the life of the reservation.

Trigger

Field Name	Type	Description
id	**String**	Trigger id - internal ID used by moab to track triggers

Field Name	Type	Description
action	**String**	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	
blockTime	**Date**	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	
eventType	**TriggerEventType**	
expireTime	**Date**	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	
interval	**Boolean**	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	Specifies the number of times Action will be attempted before the trigger is designated a failure.
multiFire	**Boolean**	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length
objectId	**String**	The ID of the object which this is attached to.

Field Name	Type	Description
objectType	**String**	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.
sets	**String**	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	The type of the trigger.
unsets	**String**	Variable this trigger destroys upon success or failure.

TriggerActionType

This enumeration specifies the action type of a trigger.

Value	Description
CANCEL	Only apply to reservation triggers.
CHANGE_ PARAM	Run changeparam (NOT PERSISTENT).
JOB_ PREEMPT	Indicates that the trigger should preempt all jobs currently allocating resources assigned to the trigger's parent object. Only apply to reservation triggers.
MAIL	Sends an e-mail.
INTERNAL	Modifies an object internally in Moab. This can be used to set a job hold for example.
EXEC	Execute the trigger action. Typically used to run a script.
MODIFY	Can modify object that trigger is attached to.
QUERY	
RESERVE	
SUBMIT	

TriggerEventType

This enumeration specifies the event type of a trigger.

Value	Description
CANCEL	
CHECKPOINT	
CREATE	
END	

Value	Description
EPOCH	
FAIL	
HOLD	
MIGRATE	
MODIFY	
PREEMPT	
STANDING	
START	
THRESHOLD	
DISCOVER	
LOGROLL	

TriggerFlag

This enumeration specifies a flag belonging to a trigger.

Value	Description
ATTACH_ERROR	If the trigger outputs anything to stderr, Moab will attach this as a message to the trigger object.
CLEANUP	If the trigger is still running when the parent object completes or is canceled, the trigger will be killed.
CHECKPOINT	Moab should always checkpoint this trigger. See Checkpointing a Trigger in the Moab Workload Manager documentation for more information.

Value	Description
GLOBAL_VARS	The trigger will look in the name space of all nodes with the globalvars flag in addition to its own name space. A specific node to search can be specified using the following format: globalvars+node_id
INTERVAL	Trigger is periodic.
MULTIFIRE	Trigger can fire multiple times.
OBJECT_XML_ STDIN	Trigger passes its parent's object XML information into the trigger's stdin. This only works for exec triggers with reservation type parents.
USER	The trigger will execute under the user ID of the object's owner. If the parent object is sched, the user to run under may be explicitly specified using the format user+<username>, for example flags=user+john:
GLOBAL_ TRIGGER	The trigger will be (or was) inserted into the global trigger list.
ASYNCHRONOUS	An asynchronous trigger.
LEAVE_FILES	Do not remove stderr and stdout files.
PROBE	The trigger's stdout will be monitored.
PROBE_ALL	The trigger's stdout will be monitored.
GENERIC_ SYSTEM_JOB	The trigger belongs to a generic system job (for checkpointing).
REMOVE_STD_ FILES	The trigger will delete stdout/stderr files after it has been reset.
RESET_ON_ MODIFY	The trigger resets if the object it is attached to is modified, even if multifire is not set.

Value	Description
SOFT_KILL	By default, a `SIGKILL` (kill -9) signal is sent to the kill the script when a trigger times out. This flag will instead send a `SIGTERM` (kill -15) signal to kill the script. The `SIGTERM` signal will allow the script to trap the signal so that the script can clean up any residual information on the system (instead of just dying, as with the `SIGKILL` signal). NOTE: A timed-out trigger will only receive one kill signal. This means that if you specify this flag, a timed-out trigger will only receive the `SIGTERM` signal, and never the `SIGKILL` signal.

TriggerPeriod

This enumeration specifies the period of a trigger.

Value	Description
MINUTE	
HOUR	
DAY	
WEEK	
MONTH	

TriggerType

This enumeration specifies the type of the trigger.

Value	Description
generic	
elastic	

Trigger

Field Name	Type	Description
id	**String**	Trigger id - internal ID used by moab to track triggers
action	**String**	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	
blockTime	**Date**	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	
eventType	**TriggerEventType**	
expireTime	**Date**	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	
interval	**Boolean**	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	Specifies the number of times Action will be attempted before the trigger is designated a failure.
multiFire	**Boolean**	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length

Field Name	Type	Description
objectId	**String**	The ID of the object which this is attached to.
objectType	**String**	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.
sets	**String**	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	The type of the trigger.

Field Name	Type	Description
unsets	**String**	Variable this trigger destroys upon success or failure.

API version 2

StandingReservation

This class represents a standing reservation.

A standing reservation is any reservation that is not a one-time reservation. This includes reservations that recur every day or every week, or infinite reservations.

Field Name	Type	Description
id	**String**	The unique ID of the standing reservation.
access	**ReservationAccess**	If set to ReservationAccess.SHARED, allows a standing reservation to use resources already allocated to other non-job reservations. Otherwise, these other reservations block resource access.
accounts	**Set<String>**	Specifies that jobs with the associated accounts may use the resources contained within this reservation.
aclRules	**Set<AclRule>**	The set of access control rules associated with this standing reservation.
chargeAccount	**String**	Specifies the account to which Moab will charge all idle cycles within the reservation (via the accounting manager).
chargeUser	**String**	Specifies the user to which Moab will charge all idle cycles within the reservation (via the accounting manager). Must be used in conjunction with chargeAccount
classes	**Set<String>**	Specifies that jobs with the associated classes/queues may use the resources contained within this reservation.
clusters	**Set<String>**	Specifies that jobs originating within the listed clusters may use the resources contained within this reservation.
comment	**String**	Specifies a descriptive message associated with the standing reservation and all child reservations

Field Name	Type	Description
days	**Set<String>**	Specifies which days of the week the standing reservation is active. Valid values are Mon, Tue, Wed, Thu, Fri, Sat, Sun, or [ALL].
depth	**Integer**	Specifies the depth of standing reservations to be created, starting at depth 0 (one per period).
disabled	**Boolean**	Specifies if the standing reservation should no longer spawn child reservations.
endOffset	**Long**	The ending offset, in seconds, from the beginning of the current period (DAY or WEEK), for this standing reservation. See examples at startOffset.
flags	**Set<ReservationFlag>**	Specifies special reservation attributes.
groups	**Set<String>**	Specifies the groups allowed access to this standing reservation.
hosts	**Set<String>**	Specifies the set of hosts that the scheduler can search for resources to satisfy the reservation. If specified using the class:X format, Moab only selects hosts that support the specified class. If TASKCOUNT is also specified, only TASKCOUNT tasks are reserved. Otherwise, all matching hosts are reserved.
jobAttributes	**Set<JobFlag>**	Specifies job attributes that grant a job access to the reservation. Values can be specified with a != assignment to only allow jobs NOT requesting a certain feature inside the reservation.
maxJob	**Integer**	Specifies the maximum number of jobs that can run in the reservation.
maxTime	**Integer**	Specifies the maximum time for jobs allowable. Can be used with affinity to attract jobs with same maxTime.
messages	**Set<String>**	Messages associated with the reservation.

Field Name	Type	Description
nodeFeatures	**Set<String>**	Specifies the required node features for nodes that are part of the standing reservation.
os	**String**	Specifies the operating system that should be in place during the reservation. Moab provisions this OS at reservation start and restores the original OS at reservation completion.
owner	**EmbeddedCredential**	Specifies the owner of the reservation. Setting ownership for a reservation grants the user management privileges, including the power to release it. Setting a user as the owner of a reservation gives that user privileges to query and release the reservation. For sandbox reservations, sandboxes are applied to a specific peer only if owner is set to CLUSTER:<PEERNAME>
partition	**String**	Specifies the partition in which to create the standing reservation. Defaults to ALL.
period	**TimeWindow**	Period of the Standing reservation. Defaults to TimeWindow.DAY.
procLimit	**IntLimit**	Specifies the processor limit for jobs requesting access to this standing reservation.
psLimit	**IntLimit**	Specifies the processor-second limit for jobs requesting access to this standing reservation.
qoses	**Set<String>**	Specifies that jobs with the listed QoS names can access the reserved resources.
reservationAccessList	**Set<Reservation>**	A list of reservations to which the specified reservation has access.
reservationGroup	**String**	The group of the reservation.

Field Name	Type	Description
resources	**Map<String, Integer>**	Specifies what resources constitute a single standing reservation task. (Each task must be able to obtain all of its resources as an atomic unit on a single node.) Supported resources currently include the following: • PROCS (number of processors) • MEM (real memory in MB) • DISK (local disk in MB) • SWAP (virtual memory in MB)
rollbackOffset	**Integer**	Specifies the minimum time in the future at which the reservation may start. This offset is rolling meaning the start time of the reservation will continuously roll back into the future to maintain this offset. Rollback offsets are a good way of providing guaranteed resource access to users under the conditions that they must commit their resources in the future or lose dedicated access. See QoS Credential in the Moab Workload Manager documentation for more information on quality of service and service level agreements.
startOffset	**Long**	The starting offset, in seconds, from the beginning of the current period (DAY or WEEK), for this standing reservation. If period is DAY, the offset is from midnight (00:00) of the current day. If period is WEEK, the offset is from midnight Sunday of the current week. Example 1: For a standing reservation that begins at 9:00 and ends at 17:00 every day, period is DAY, startOffset is 32400 (9*60*60), and endOffset is 61200 (17*60*60). Example 2: For a standing reservation that begins at 9:00 Monday and ends at 17:00 Friday every week, period is WEEK, startOffset is 118800 ((24+9) *60*60), and endOffset is 493200 (((5*24)+17) *60*60).

Field Name	Type	Description
taskCount	**Integer**	Specifies how many tasks should be reserved for the reservation Default is 0 (unlimited tasks).
tasksPerNode	**Integer**	Specifies the minimum number of tasks per node that must be available on eligible nodes. Default is 0 (no TPN constraint)
timeLimit	**Integer**	Specifies the maximum allowed overlap between the standing reservation and a job requesting resource access. Default is null (-1 in moab)
triggers	**Set<Trigger>**	Triggers associated with the reservation.
type	**String**	The type of the reservation.
users	**Set<String>**	Specifies which users have access to the resources reserved by this reservation.

ReservationAccess

The access type of a standing reservation. If set to SHARED, allows a standing reservation to use resources already allocated to other non-job reservations. Otherwise, these other reservations block resource access.

Value	Description
DEDICATED	
SHARED	

AclRule

This class represents a rule that can be in Moab's access control list (ACL) mechanism.

The basic AclRule information is the object's name and type. The type directly maps to an AclType value. The default mechanism Moab uses to check the ACL for a particular item is if the user or object coming in has ANY of the values in the ACL, then the user or object is given access. If no values match the user or object in question, the user or object is rejected access.

Field Name	Type	Description
affinity	**AclAffinity**	Reservation ACLs allow or deny access to reserved resources but they may also be configured to affect a job's affinity for a particular reservation. By default, jobs gravitate toward reservations through a mechanism known as positive affinity. This mechanism allows jobs to run on the most constrained resources leaving other, unreserved resources free for use by other jobs that may not be able to access the reserved resources. Normally this is a desired behavior. However, sometimes, it is desirable to reserve resources for use only as a last resort-using the reserved resources only when there are no other resources available. This last resort behavior is known as negative affinity. Defaults to AclAffinity.POSITIVE.
comparator	**ComparisonOperator**	The type of comparison to make against the ACL object. Defaults to ComparisonOperator.EQUAL.
type	**AclType**	The type of the object that is being granted (or denied) access.
value	**String**	The name of the object that is being granted (or denied) access.

AclAffinity

This enumeration describes the values available for describing how a rule is used in establishing access to an object in Moab. Currently, these ACL affinities are used only for granting access to reservations.

Value	Description
NEGATIVE	Access to the object is repelled using this rule until access is the last choice.
NEUTRAL	Access to the object is not affected by affinity.
POSITIVE	Access to the object is looked at as the first choice.
PREEMPTIBLE	Access to the object given the rule gives preemptible status to the accessor. Supported only during GET.

Value	Description
REQUIRED	The rule in question must be satisified in order to gain access to the object. Supported only during GET.
UNAVAILABLE	The rule does not have its affinity available. Supported only during GET.

ComparisonOperator

This enumeration is used when Moab needs to compare items. One such use is in Access Control Lists (ACLs).

Value	Description
GREATER_THAN	Valid values: ">", "gt"
GREATER_THAN_OR_EQUAL	Valid values: ">=", "ge"
LESS_THAN	Valid values: "<", "lt"
LESS_THAN_OR_EQUAL	Valid values: "<=", "le"
EQUAL	Valid values: "==", "eq", "="
NOT_EQUAL	Valid values: "!=", "ne", "<>"
LEXIGRAPHIC_SUBSTRING	Valid value: "%<"
LEXIGRAPHIC_NOT_EQUAL	Valid value: "%!"
LEXIGRAPHIC_EQUAL	Valid value: "%="

AclType

This enumeration describes the values available for the type of an ACL Rule.

Value	Description
USER	User

Value	Description
GROUP	Group
ACCOUNT	Account or Project
CLASS	Class or Queue
QOS	Quality of Service
CLUSTER	Cluster
JOB_ID	Job ID
RESERVATION_ID	Reservation ID
JOB_TEMPLATE	Job Template
JOB_ATTRIBUTE	Job Attribute
DURATION	Duration in Seconds
PROCESSOR_SECONDS	Processor Seconds
JPRIORITY	Not supported
MEMORY	Not supported
NODE	Not supported
PAR	Not supported
PROC	Not supported
QTIME	Not supported
QUEUE	Not supported
RACK	Not supported

Value	Description
SCHED	Not supported
SYSTEM	Not supported
TASK	Not supported
VC	Not supported
XFACTOR	Not supported

ReservationFlag

The flag types of a reservation.

Value	Description
ALLOWJOBOVERLAP	Allows jobs to overlap this Reservation, but not start during it (unless they have ACL access).
APPLYPROFRESOURCES	Only apply resource allocation info from profile.
DEADLINE	Reservation should be scheduled against a deadline.
IGNIDLEJOBS	Ignore idle job reservations.
IGNJOBRSV	Ignore job reservations, but not user or other reservations.
CHARGE	Charge the idle cycles in the accounting manager.
NOVMMIGRATIONS	Override the VM Migration Policy and don't migrate VMs that overlap this reservation.
OWNERPREEMPTIGNOREMINTIME	Owner ignores preemptmintime for this reservation.
PROVISION	Reservation should be capable of provisioning.
NOACLOVERLAP	Reservation will not look at ACLs to overlap job (when using exclusive).

Value	Description
ADVRES	If set, the reservation is created in advance of needing it.
ADVRESJOBDESTROY	Cancel any jobs associated with the reservation when it is released.
ALLOWGRID	The reservation is set up for use in a grid environment.
ALLOWPRSV	Personal reservations can be created within the space of this standing reservation (and ONLY this standing reservation). By default, when a standing reservation is given the flag ALLOWPRSV, it is given the ACL rule USER==ALL+ allowing all jobs and all users access.
BYNAME	Reservation only allows access to jobs that meet reservation ACLs and explicitly request the resources of this reservation using the job ADVRES flag.
DEDICATEDNODE	If set, only one active reservation is allowed on a node.
OWNEREXCLUSIVEBF	When an owner job is idle, other jobs are not allowed to backfill.
DEDICATEDRESOURCE	The reservation is only placed on resources that are not reserved by any other reservation, including jobs and other reservations.
EXCLUDEJOBS	Makes a reservation job exclusive, where only one job can run in the reservation.
ENDTRIGHASFIRED	A trigger has finished firing.
ENFORCENODESET	Enforce node sets when creating reservation.
EXCLUDEALLBUTSB	Reservation only shares resources with sandboxes.
EXCLUDEMYGROUP	Exclude reservations within the same group.
IGNRSV	Forces the reservation onto nodes regardless of whether there are other reservations currently residing on the nodes.

Value	Description
IGNSTATE	Request ignores existing resource reservations, allowing the reservation to be forced onto available resources even if this conflicts with other reservations.
ISACTIVE	If set, the reservation is currently active.
ISCLOSED	If set, the reservation is closed.
ISGLOBAL	If set the reservation applies to all resources.
OWNERPREEMPT	The owner of the reservation is given preemptor status for resources contained in the reservation.
PARENTLOCK	The reservation can only be destroyed by destroying its parent.
PREEMPTEE	The reservation is preemptible.
PLACEHOLDER	The reservation is a placeholder for resources.
PRSV	The reservation is a non-adminstrator, non-standing reservation, user-created reservation.
REQFULL	The reservation will fail if all resources requested cannot be allocated.
SCHEDULEVCRSV	The reservation was created as part of a schedule VC command. This pertains to reservations creating while scheduling MWS Services, and these are filtered from the MWS output of reservations.
SINGLEUSE	The reservation is automatically removed after completion of the first job to use the reserved resources.
SPACEFLEX	The reservation is allowed to adjust resources allocated over time in an attempt to optimize resource utilization.
STANDINGRSV	If set, the reservation was created by a standing reservation instance.
STATIC	Makes a reservation ineligible to modified or canceled by an administrator.

Value	Description
SYSTEMJOB	The reservation was created by a system job.
TIMEFLEX	The reservation is allowed to adjust the reserved time frame in an attempt to optimize resource utilization.
TRIGHASFIRED	The reservation has one or more triggers that have fired on it.
WASACTIVE	The reservation was previously active.
EVACVMS	Evacuate virtual machines on the node when the reservation starts.

JobFlag

This enumeration specifies the flag types of a job.

Value	Description
NONE	
BACKFILL	The job is using backfill to run.
COALLOC	The job can use resources from multiple resource managers and partitions.
ADVRES	The job requires the use of a reservation.
NOQUEUE	The job will attempt to execute immediately or fail.
ARRAYJOB	The job is part of a job array.
ARRAYJOBPARLOCK	This array job will only run in one partition.
ARRAYJOBPARSPAN	This array job will span partitions (default).
ARRAYMASTER	This job is the master of a job array.
BESTEFFORT	The job will succeed if even partial resources are available.
RESTARTABLE	The job is restartable.

Value	Description
SUSPENDABLE	The job is suspendable.
HASPREEMPTED	This job preempted other jobs to start.
PREEMPTEE	The job is a preemptee and therefore can be preempted by other jobs.
PREEMPTOR	The job is a preemptor and therefore can preempt other jobs.
RSVMAP	The job is based on a reservation.
SPVIOLATION	The job was started with a soft policy violation.
IGNNODEPOLICIES	The job will ignore node policies.
IGNPOLICIES	The job will ignore idle, active, class, partition, and system policies.
IGNNODESTATE	The job will ignore node state in order to run.
IGNIDLEJOBRSV	The job can ignore idle job reservations. The job granted access to all idle job reservations.
INTERACTIVE	The job needs to interactive input from the user to run.
FSVIOLATION	The job was started with a fairshare violation.
GLOBALQUEUE	The job is directly submitted without doing any authentication.
NORESOURCES	The job is a system job that does not need any resources.
NORMSTART	The job will not query a resource manager to run.
CLUSTERLOCKED	The job is locked into the current cluster and cannot be migrated elsewhere. This is for grid mode.
FRAGMENT	The job can be run across multiple nodes in individual chunks.

Value	Description
SYSTEMJOB	The job is a system job which simply runs on the same node that Moab is running on. This is usually used for running scripts and other executables in workflows.
ADMINSETIGNPOLICIES	The IGNPOLICIES flag was set by an administrator.
EXTENDSTARTWALLTIME	The job duration (walltime) was extended at job start.
SHAREDMEM	The job will share its memory across nodes.
BLOCKEDBYGRES	The job's generic resource requirement caused the job to start later.
GRESONLY	The job is requesting only generic resources, no compute resources.
TEMPLATESAPPLIED	The job has had all applicable templates applied to it.
META	META job, just a container around resources.
WIDERSVSEARCHALGO	This job prefers the wide search algorithm.
VMTRACKING	The job is a VMTracking job for an externally-created VM (via job template).
DESTROYTEMPLATESUBMITTED	A destroy job has already been created from the template for this job.
PROCSPECIFIED	The job requested processors on the command line.
CANCELONFIRSTFAILURE	Cancel job array on first array job failure.
CANCELONFIRSTSUCCESS	Cancel job array on first array job success.
CANCELONANYFAILURE	Cancel job array on any array job failure.
CANCELONANYSUCCESS	Cancel job array on any array job success.
CANCELONEXITCODE	Cancel job array on a specific exit code.
NOVMMIGRATE	Do not migrate the virtual machine that this job sets up.

Value	Description
JOINSTDERRTOSTDOUT	Join the stderr file to the stdout file.
JOINSTDOUTTOSTDERR	Join the stdout file to the stderr file.
PURGEONSUCCESSONLY	Only purge the job if it completed successfully
ALLPROCS	Each job compute task requests all the procs on its node
COMMLOCAL	Each job communications are localized, with minimal routing outside job shape
COMMTOLERANT	Each job communications are low-intensity and insensitive to interference

EmbeddedCredential

Field Name	Type	Description
name	**String**	
type	**CredentialType**	

CredentialType

Value	Description
USER	
GROUP	
ACCOUNT	
CLASS	
QOS	
NOT_SPECIFIED	

TimeWindow

This enumeration represents some common time windows. It can be used when for many purposes, but was created specifically for statistics.

Value	Description
MINUTE	
HOUR	
DAY	
WEEK	
MONTH	
YEAR	
INFINITY	

IntLimit

Field Name	Type	Description
qualifier	**String**	One of: • < • <= • == • >= • >
value	**Integer**	

Reservation

A reservation is the mechanism by which Moab guarantees the availability of a set of resources at a particular time. Each reservation consists of three major components: (1) a set of resources, (2) a time frame, and (3) an access control list. It is a scheduler role to ensure that the access control list is not violated during the reservation's lifetime (that is, its time frame) on the resources listed. For example, a reservation may specify that node002 is reserved for user Tom on Friday. The scheduler is thus constrained to make certain that only Tom's jobs can use node002 at any time on Friday.

Field Name	Type	Description
id	**String**	The unique ID of the reservation.
accountingAccount	**String**	Accountable Account.
accountingGroup	**String**	Accountable Group.
accountingQOS	**String**	Accountable QOS.
accountingUser	**String**	Accountable User.
aclRules	**Set<AclRule>**	The set of access control rules associated with this reservation.
allocatedNodeCount	**Integer**	The number of allocated nodes for this reservation.
allocatedNodes	**Set<DomainProxyVersion1>**	The nodes allocated to the reservation.
allocatedProcessorCount	**Integer**	The number of allocated processors.
allocatedTaskCount	**Integer**	The number of allocated tasks.
comments	**String**	Reservation's comments or description.
creationDate	**Date**	Creation date. Automatically set by Moab when a user creates the reservation.
duration	**Long**	The duration of the reservation (in seconds).

Field Name	Type	Description
endDate	**Date**	The end date of the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation ends.
excludeJobs	**Set<String>**	The list of jobs to exclude. Client must also set the IGNJOBRSV reservation flag. Otherwise, results are undefined. Used only during reservation creation.
expireDate	**Date**	The date/time when the reservation expires and vacates.
flags	**Set<ReservationFlag>**	The flags associated with the reservation.
globalId	**String**	Global reservation ID.
hostListExpression	**String**	The list of nodes a user can select to reserve. This may or may not be the nodes that are currently allocated to this reservation. Note: Either hostListExpression or taskCount must be set to create a reservation.
idPrefix	**String**	The user-specified prefix for this reservation. If provided, Moab combines the idPrefix with an integer, and the combination is the unique identifier for this reservation.
isActive	**Boolean**	State whether or not this reservation is currently active.
isTracked	**Boolean**	States whether reservation resource usage is tracked.
label	**String**	When a label is assigned to a reservation, the reservation can then be referenced by that label as well as by the reservation name.
maxTasks	**Integer**	The maximum number of tasks for this reservation.

Field Name	Type	Description
messages	Set<MessageVersion1>	Messages for the reservation.
owner	EmbeddedCredential	The owner of the reservation
partitionId	**String**	The ID of the partition this reservation is for.
profile	**String**	The profile that this reservation is using. A profile is a specification of attributes that all reservations share. Used only during reservation creation.
requirements	ReservationRequirement	The reservation's requirements.
reservationGroup	**String**	The reservation group to which the reservation belongs.
resources	**Map<String, Integer>**	The reservation's resources. This field is a map, where the key is PROCS, MEM DISK, SWAP, or one or more user-defined keys.
startDate	**Date**	The start time for the reservation. This is especially useful for one-time reservations, which have an exact time for when a reservation starts.
statistics	ReservationStatistics	The reservation's statistical information.
subType	**String**	The reservation sub-type.
taskCount	**Integer**	The number of tasks that must be allocated to satisfy the reservation request. Note: Either hostListExpression or taskCount must be set to create a reservation.
trigger	Trigger	Trigger for reservation. Used only during reservation creation.
triggerIds	**Set<String>**	The IDs of the triggers attached to this reservation.

Field Name	Type	Description
uniqueIndex	**String**	The globally-unique reservation index.
variables	**Map<String, Map>**	The set of variables for this reservation.

DomainProxyVersion1

Field Name	Type	Description
id	**String**	The id of the object.

MessageVersion1

Field Name	Type	Description
author	**String**	The author of the message.
creationTime	**Date**	The time the message was created in epoch time.
expireTime	**Date**	The time the message will be deleted in epoch time.
index	**Integer**	The index of the message relative to other messages in Moab's memory.
message	**String**	The comment information itself.
messageCount	**Integer**	The number of times this message has been displayed.
priority	**Double**	An optional priority that can be attached to the comment.

ReservationRequirement

Represents all the types of requirements a user can request while creating a reservation.

Field Name	Type	Description
architecture	**String**	Required architecture.

Field Name	Type	Description
featureList	**Set<String>**	The list of features required for this reservation.
featureMode	**String**	Required feature mode.
memory	**Integer**	Required node memory, in MB.
nodeCount	**Integer**	Required number of nodes.
nodeIds	**Set<String>**	The list of node IDs required for this reservation.
os	**String**	Required Operating System.
taskCount	**Integer**	Required task count.

ReservationStatistics

Represents some basic statistical information that is kept about the usage of reservations. All metrics that are kept track relate to processor-seconds usage.

Field Name	Type	Description
caps	**Long**	The current active processor-seconds in the last reported iteration.
cips	**Long**	The current idle processor-seconds in the last reported iteration.
taps	**Long**	The total active processor-seconds over the life of the reservation.
tips	**Long**	The total idle processor-seconds over the life of the reservation.

Trigger

Field Name	Type	Description
id	**String**	Trigger id - internal ID used by moab to track triggers

Field Name	Type	Description
action	**String**	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	
blockTime	**Date**	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	
eventType	**TriggerEventType**	
expireTime	**Date**	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	
interval	**Boolean**	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	Specifies the number of times Action will be attempted before the trigger is designated a failure.
multiFire	**Boolean**	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length
objectId	**String**	The ID of the object which this is attached to.

Field Name	Type	Description
objectType	**String**	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.
sets	**String**	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	The type of the trigger.
unsets	**String**	Variable this trigger destroys upon success or failure.

TriggerActionType

This enumeration specifies the action type of a trigger.

Value	Description
CANCEL	Only apply to reservation triggers.
CHANGE_ PARAM	Run changeparam (NOT PERSISTENT).
JOB_ PREEMPT	Indicates that the trigger should preempt all jobs currently allocating resources assigned to the trigger's parent object. Only apply to reservation triggers.
MAIL	Sends an e-mail.
INTERNAL	Modifies an object internally in Moab. This can be used to set a job hold for example.
EXEC	Execute the trigger action. Typically used to run a script.
MODIFY	Can modify object that trigger is attached to.
QUERY	
RESERVE	
SUBMIT	

TriggerEventType

This enumeration specifies the event type of a trigger.

Value	Description
CANCEL	
CHECKPOINT	
CREATE	
END	

Value	Description
EPOCH	
FAIL	
HOLD	
MIGRATE	
MODIFY	
PREEMPT	
STANDING	
START	
THRESHOLD	
DISCOVER	
LOGROLL	

TriggerFlag

This enumeration specifies a flag belonging to a trigger.

Value	Description
ATTACH_ERROR	If the trigger outputs anything to stderr, Moab will attach this as a message to the trigger object.
CLEANUP	If the trigger is still running when the parent object completes or is canceled, the trigger will be killed.
CHECKPOINT	Moab should always checkpoint this trigger. See Checkpointing a Trigger in the Moab Workload Manager documentation for more information.

Value	Description
GLOBAL_VARS	The trigger will look in the name space of all nodes with the globalvars flag in addition to its own name space. A specific node to search can be specified using the following format: globalvars+node_id
INTERVAL	Trigger is periodic.
MULTIFIRE	Trigger can fire multiple times.
OBJECT_XML_ STDIN	Trigger passes its parent's object XML information into the trigger's stdin. This only works for exec triggers with reservation type parents.
USER	The trigger will execute under the user ID of the object's owner. If the parent object is sched, the user to run under may be explicitly specified using the format user+<username>, for example flags=user+john:
GLOBAL_ TRIGGER	The trigger will be (or was) inserted into the global trigger list.
ASYNCHRONOUS	An asynchronous trigger.
LEAVE_FILES	Do not remove stderr and stdout files.
PROBE	The trigger's stdout will be monitored.
PROBE_ALL	The trigger's stdout will be monitored.
GENERIC_ SYSTEM_JOB	The trigger belongs to a generic system job (for checkpointing).
REMOVE_STD_ FILES	The trigger will delete stdout/stderr files after it has been reset.
RESET_ON_ MODIFY	The trigger resets if the object it is attached to is modified, even if multifire is not set.

Value	Description
SOFT_KILL	By default, a `SIGKILL` (kill -9) signal is sent to the kill the script when a trigger times out. This flag will instead send a `SIGTERM` (kill -15) signal to kill the script. The `SIGTERM` signal will allow the script to trap the signal so that the script can clean up any residual information on the system (instead of just dying, as with the `SIGKILL` signal). NOTE: A timed-out trigger will only receive one kill signal. This means that if you specify this flag, a timed-out trigger will only receive the `SIGTERM` signal, and never the `SIGKILL` signal.

TriggerPeriod

This enumeration specifies the period of a trigger.

Value	Description
MINUTE	
HOUR	
DAY	
WEEK	
MONTH	

TriggerType

This enumeration specifies the type of the trigger.

Value	Description
generic	
elastic	

Trigger

Field Name	Type	Description
id	**String**	Trigger id - internal ID used by moab to track triggers
action	**String**	For exec atype triggers, signifies executable and arguments. For jobpreempt atype triggers, signifies PREEMPTPOLICY to apply to jobs that are running on allocated resources. For changeparam atype triggers, specifies the parameter to change and its new value (using the same syntax and behavior as the changeparam command).
actionType	**TriggerActionType**	
blockTime	**Date**	Time (in seconds) Moab will suspend normal operation to wait for trigger execution to finish. Use caution as Moab will completely stop normal operation until BlockTime expires.
description	**String**	
eventType	**TriggerEventType**	
expireTime	**Date**	Time at which trigger should be terminated if it has not already been activated.
failOffset	**Date**	Specifies the time (in seconds) that the threshold condition must exist before the trigger fires.
flags	**Set<TriggerFlag>**	
interval	**Boolean**	When used in conjunction with MultiFire and RearmTime trigger will fire at regular intervals. Can be used with TriggerEventType.EPOCH to create a Standing Trigger. Defaults to false
maxRetry	**Integer**	Specifies the number of times Action will be attempted before the trigger is designated a failure.
multiFire	**Boolean**	Specifies whether this trigger can fire multiple times. Defaults to false.
name	**String**	Trigger name - can be auto assigned by moab or requested. Alphanumeric up to 16 characters in length

Field Name	Type	Description
objectId	**String**	The ID of the object which this is attached to.
objectType	**String**	The type of object which this is attached to. Possible values: • vm - Virtual Machine
offset	**Date**	Relative time offset from event when trigger can fire.
period	**TriggerPeriod**	Can be used in conjunction with Offset to have a trigger fire at the beginning of the specified period. Can be used with EType epoch to create a standing trigger.
rearmTime	**Date**	Time between MultiFire triggers. Rearm time is enforced from the trigger event time.
requires	**String**	Variables this trigger requires to be set or not set before it will fire. Preceding the string with an exclamation mark (!) indicates this variable must NOT be set. Used in conjunction with Sets to create trigger dependencies.
sets	**String**	Variable values this trigger sets upon success or failure. Preceding the string with an exclamation mark (!) indicates this variable is set upon trigger failure. Preceding the string with a caret (^) indicates this variable is to be exported to the parent object when the current object is destroyed through a completion event. Used in conjunction with Requires to create trigger dependencies.
threshold	**String**	Reservation usage threshold - When reservation usage drops below Threshold, trigger will fire. Threshold usage support is only enabled for reservations and applies to percent processor utilization. gmetric thresholds are supported with job, node, credential, and reservation triggers. See Threshold Triggers in the Moab Workload Manager documentation for more information.
timeout	**Date**	Time allotted to this trigger before it is marked as unsuccessful and its process (if any) killed.
type	**TriggerType**	The type of the trigger.

Field Name	Type	Description
unsets	**String**	Variable this trigger destroys upon success or failure.

Related Topics

- Standing Reservations

Fields: User's Permissions

See the associated Permissions resource section for more information on how to use this resource and supported operations.

Additional references

Type	Value	Additional information
Permissions resource	`permissions/users`	Permissions
Hooks filename	`permissions.users.groovy`	Pre- and Post-Processing Hooks
Distinct query-supported	Yes	Distinct

API version 3

UserPermission

Field Name	Type	Description
id	**String**	The unique ID of the cached user permission.
name	**String**	The unique name of the user.
permissions	**List<Permission>**	The list of permissions.

Permission

Represents a permission

Field Name	Type	Description
id	**String**	The unique ID of this role.
action	**String**	The action that can be performed on the resource.
administrator	**Boolean**	If true, grants full rights over the given resource for the given action. For example, if resource is "jobs" and action is "update" and administrator is true, then this permission allows the user to update any job, not just jobs owned by the user.
description	**String**	A description of this permission.
fieldPath	**String**	Field level ACL control, if null or '*', all fields are accessible, otherwise requests must match dot delimited path. Currently only checked when doing writable actions. Example - attributes.*: create|update
label	**String**	A human readable label for this permission.
resource	**String**	The resource the permission applies to.
resourceFilter	**Map<String, Map>**	A map used to limit which resource instances this permission applies to. If this is null then the permission will apply to all instances of the resource. For api permissions the filter uses mongo query syntax.

Field Name	Type	Description
scope	**PrivilegeScope**	Whether this permission applies to the principal's tenant-associated resources or globally
type	**String**	The type of the permission. Only 'api' type permissions are enforced.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

API version 2

UserPermission

Field Name	Type	Description
id	**String**	The unique ID of the cached user permission.
name	**String**	The unique name of the user.
permissions	**List<Permission>**	The list of permissions.

Permission

Represents a permission

Field Name	Type	Description
id	**String**	The unique ID of this role.
action	**String**	The action that can be performed on the resource.
administrator	**Boolean**	If true, grants full rights over the given resource for the given action. For example, if resource is "jobs" and action is "update" and administrator is true, then this permission allows the user to update any job, not just jobs owned by the user.
description	**String**	A description of this permission.
fieldPath	**String**	Field level ACL control, if null or '*', all fields are accessible, otherwise requests must match dot delimited path. Currently only checked when doing writable actions. Example - attributes.*: create|update
label	**String**	A human readable label for this permission.
resource	**String**	The resource the permission applies to.
resourceFilter	**Map<String, Map>**	A map used to limit which resource instances this permission applies to. If this is null then the permission will apply to all instances of the resource. For api permissions the filter uses mongo query syntax.

Field Name	Type	Description
scope	**PrivilegeScope**	Whether this permission applies to the principal's tenant-associated resources or globally
type	**String**	The type of the permission. Only 'api' type permissions are enforced.

PrivilegeScope

Some permissions and roles ignore tenants and apply globally. Others apply only to the resources associated with the principal's tenants.

Value	Description
GLOBAL	Describes a role or permission that applies globally, irrespective of the principal's tenants. This scope can be applied to any role or permission.
TENANT	Describes a role or permission that applies only to the resources associated with the principal's tenants. This scope can be applied to any role, but only to those permissions associated with tenanted resources (e.g. nodes, services, etc.).
NONE	Scope doesn't apply to some permissions. As of right now, all non-domain permissions (e.g. those created by Viewpoint) don't need a scope. NONE should therefore be assigned to all non-domain permissions.

Related Topics

- Permissions