AirMettle User Guide

Release 1.1.0

AirMettle, Inc.

Dec 16, 2024

USER GUIDE

CHAPTER ONE

OVERVIEW

AirMettle is an analytical object store that provides the accessibility and data protection capabilities expected from an object storage system in addition to an interface for accelerated analytics.

This document describes the user interfaces and APIs provided by AirMettle.

CHAPTER

TWO

AUTHENTICATION

The admin APIs below are used in the airmettle UI and they can also be used by directly making http requests. The UI uses /admin/login API to login and also supports OpenID Connect configuration for logging in. A session token is stored in memory so if the airmettle nodes are bounced, you will be prompted to login again in the UI.

2.1 IP ACL

An IP access control list can be set to automatically allow access.

Set the IP address that will be making request in the Admin Trusted list of IPs in the IP Access Control section of the UI or via the `/admin/addIP` administrative interface.

By default, access is limited to … (todo)

2.2 Internal Auth

2.2.1 `/admin/login`

Login using local authentication. A token is returned that can then be used in the Authorization header as a bearer token for other API requests.

Tokens are stored internally in a distributed hash table, and will be lost if the system is bounced.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`loginId`	username	String
`password`	password	String
`application`	Application name (tenant)	String
`source`	Optional parameter that indicates if request is for UI or API	String value: ui, api

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
```

(continues on next page)

(continued from previous page)

```
        "statusText": "Success",
        "superUser": "SUPERUSER",
        "adminNode": true,
        "userName": "admin",
        "token": "dAFpbnk..."
    }
}
```

2.2.2 /admin/logout

Logout using the token passed in cookie or authorization header.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

2.3 S3 Authentication

CHAPTER

THREE

OBJECT STORAGE

The AirMettle storage system supports an S3-compatible interface.

3.1 S3 API

These are S3 compatible APIs that are supported. The API syntax is identical to the AWS S3 API. Further documentation can be found here: S3 Documentation

3.1.1 `GET /bucket`

List object in a bucket

3.1.2 `GET /bucket?cors`

Get CORs properties set on bucket

3.1.3 `GET /bucket?acl`

Get ACL set on bucket

3.1.4 `GET /bucket?versioning`

Get versioning configuration set on bucket

3.1.5 `GET /bucket/object`

Get an object

3.1.6 `GET /bucket/object?acl`

Get ACL set on object

3.1.7 `GET /bucket/object?uploads`

Get multi upload list of objects

3.1.8 `PUT /bucket`

Create a bucket

3.1.9 `PUT /bucket?cors`

Set the CORs properties on a bucket

3.1.10 `PUT /bucket?acl`

Set the ACL on a bucket

3.1.11 `PUT /bucket/object`

Upload an object

3.1.12 `PUT /bucket/object?acl`

Set the ACL on an object

3.1.13 `PUT /bucket/object?uploadId`

Upload an object part as part of a multipart upload

3.1.14 `POST /bucket?delete`

Process a multi object delete from a bucket

3.1.15 `POST /bucket/object`

Upload an object

3.1.16 `POST /bucket/object?uploadId`

Upload an object part as part of a multipart upload

3.1.17 `POST /bucket/object?select`

Perform Select Query on an object

3.1.18 `DELETE /bucket`

Delete a bucket

3.1.19 `DELETE /bucket/object`

Delete an object

3.1.20 `HEAD /bucket`

Check if bucket exists

3.1.21 `HEAD /bucket/object`

Check if object exists

CHAPTER

FOUR

ANALYTICS

The analytics subsystem supports all features of the S3-Select interface in addition to AirMettle-specific query extensions and data types.

4.1 S3 SelectObjectContent

AirMettle supports the SelectObjectContent command found in AWS S3, with the exception of support for multi part upload.

See SelectObjectContent in the S3 documentation.

4.2 Arrow Flight

Apache Arrow Flight is a high-performance data transfer framework that leverages the Apache Arrow in-memory format to efficiently exchange data between systems, making it well-suited for a variety of applications that require fast and reliable data communication.

For a more complete description see the latest Flight documentation.

AirMettle provides an Arrow Flight interface to allow execution of the SelectObjectContent command using the `DoAction` rpc. The easiest way to use the Flight interface is with one of the SDK's available as part of Apache Arrow. Additionally, AirMettle provides a Python SDK called `pyairmettle` making is simpler again.

The following is an example of using the Flight interface.

- Install `pyairmettle` via pip.

```
pip install <path-to-pyairmettle-distribution>
```

- Connect to AirMettle.

```
import pyarrow as pa

flight_client = pa.flight.FlightClient(pa.flight.Location.for_grpc_tcp(HOST, PORT))
```

- Authenticate. `pyairmettle` handles the vagaries of authentication against AirMettle.

```
import pyairmettle as pam

auth_handler = pam.v1_0.flight.ClientAuthHandler(USERNAME, PASSWORD, APPLICATION)
flight_client.authenticate(auth_handler=auth_handler)
```

- Get FlightInfo using a SelectObjectContent request.

```
options = flight.FlightCallOptions(headers=[
  (b"bucket", str.encode(bucket)),
  (b"object", str.encode(object_))
])
descriptor = pa.flight.FlightDescriptor.for_command(command=json.dumps(
  {
    "select-object-content-request": select_object_content_request.to_object()
  }
))
info = flight_client.get_flight_info(descriptor, options)
```

- Get FlightData by executing FlightAction.

  Arrow Flight assumes all parts of a returned object will have the same schema, but since JSON documents may have missing attributes, this may not always be the case. To handle this scenario, `pyairmettle` provides a MultiStreamReader to read multiple object segments with potentially different schemas.

```
tables = []
options = flight.FlightCallOptions(headers=[])
for endpoint in flight_info.endpoints:
  action = flight.Action("", endpoint.ticket.ticket)
  it = flight_client.do_action(action, options)
  while True:
```

(continues on next page)

(continued from previous page)

```
    try:
        result = next(it)

        input_stream = pa.input_stream(result.body)
        rdr = pam.ipc.MultiStreamReader(input_stream)
        tables.extend(rdr.read_all())

    except StopIteration:
        break
```

- Merge returned tables, promoting the schemas of any with missing fields.

```
merged_table = None

for table in tables:

    if merged_table is None:
        merged_table = table
    else:
        merged_table = pa.concat_tables([merged_table, table], True)
```

4.3 SQL Reference

4.3.1 SELECT command

SELECT list

The SELECT list names the attributes, functions, and expressions that you want the query to return. The list represents the output of the query.

For tabular data an *attribute* refers to a column in a table (e.g. CSV) or a name/value pair (e.g. JSON). An *attribute* in a multi-dimensional data set means a single tensor in a set of multiple tensors sharing one or more dimensions, such as a *variable* in NetCDF.

```
SELECT *
SELECT *, *, *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

The first form of SELECT with the * (asterisk) returns every row that passed the WHERE clause, as-is. The second form of SELECT creates a row with user-defined output scalar expressions projection1 and projection2 for each column.

Unlike S3 Select, both forms can be mixed with each other and the * may be used multiple times.

FROM clause

Like S3 Select, AirMettle supports the following forms of the FROM clause:

```
FROM relation
FROM relation relation_alias
FROM relation AS relation_alias
```

In each form of the FROM clause, relation is the Object or an expression based on it that's being queried. Users coming from traditional relational databases can think of this as a database schema that contains multiple views over a table or set of tensors.

Following standard SQL, the FROM clause creates rows or tensors that are filtered in the WHERE clause and projected in the SELECT list.

For JSON objects that are stored in Amazon S3 Select, you can also use the following forms of the FROM clause:

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

Using this form of the FROM clause, you can select from arrays or objects within a JSON object. You can specify path by using one of the following forms:

- By name (in an object): `.name` or `['name']`
- By index (in an array): `[index]`
- By wildcard character (in an object): `.*`
- By wildcard character (in an array): `[*]`

Note:

- This form of the FROM clause works only with JSON objects.

- Wildcard characters always emit at least one record. If no record matches, then Amazon S3 Select emits the value MISSING. During output serialization (after the query finishes running), Amazon S3 Select replaces MISSING values with empty records.
- Aggregate functions (AVG, COUNT, MAX, MIN, and SUM) skip MISSING values.
- If you don't provide an alias when using a wildcard character, you can refer to the row by using the last element in the path. For example, you could select all prices from a list of books by using the query SELECT price FROM S3Object[*].books[*].price. If the path ends in a wildcard character instead of a name, then you can use the value _1 to refer to the row. For example, instead of SELECT price FROM S3Object[*].books[*].price, you could use the query SELECT _1.price FROM S3Object[*].books[*].
- AirMettle queries always treat a JSON document as an array of root-level values. Thus, even if the JSON object that you are querying has only one root element, the FROM clause must begin with S3Object[*]. However, for compatibility reasons, Amazon S3 Select allows you to omit the wildcard character if you don't include a path. Thus, the complete clause FROM S3Object is equivalent to FROM S3Object[*] as S3Object. If you include a path, you must also use the wildcard character. So, FROM S3Object and FROM S3Object[*].path are both valid clauses, but FROM S3Object.path is not.

Examples:

Example #1

This example shows results when using the following dataset and query:

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
{}
{"id":"2"}
{}
```

Amazon S3 Select produces each result for the following reasons:

- {"id":"id-1"} – S3Object[0].Rules[0].id produced a match.
- {} – S3Object[0].Rules[1].id did not match a record, so Amazon S3 Select emitted MISSING, which was then changed to an empty record during output serialization and returned.
- {"id":"id-2"} – S3Object[0].Rules[2].id produced a match.
- {} – S3Object[1] did not match on Rules, so Amazon S3 Select emitted MISSING, which was then changed to an empty record during output serialization and returned.

If you don't want Amazon S3 Select to return empty records when it doesn't find a match, you can test for the value MISSING. The following query returns the same results as the previous query, but with the empty values omitted:

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

Example #2

This example shows results when using the following dataset and queries:

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." }, {
↪"name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon S3" }
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." }, { "name
↪": ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},{"name":
↪"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},{"name":
↪"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE clause

The WHERE clause follows this syntax:

```
WHERE condition
```

The WHERE clause filters data based on the condition. A condition is an expression that has a Boolean result. Only data for which the condition evaluates to TRUE are returned in the result.

Just like the the result of an evaluated tabular filter is applied to entire rows, an evaluated multi-dimensional filter is applied to entire tensor sets. For example, the condition `a < 5` applied to the following tabular data results in the exclusion of not only the value 7 in column *a*, but the entire last row.

Table filtering

Input 3 columns, 3 rows

a	b	c
1	2	3
4	5	6
7	8	9

Output Filtered with `a < 5`

a	b	c
1	2	3
4	5	6

Similarly, filtering the following multi-dimensional data set with the same `a < 5` condition excludes not only the cells with the value `5` in the *a* tensor, but the cells at the same coordinates in the *b* tensor.

Tensor filtering

Input 2 dimensions, 2 tensors/variables

Table 1: **a**

	0	1
0	2	3
1	4	5

Table 2: **b**

	0	1
0	12	13
1	14	15

Output Filtered with `a < 5`

Table 3: **a**

	0	1
0	2	3
1	4	

Table 4: **b**

	0	1
0	12	13
1	14	

LIMIT clause

Important: The LIMIT clause is not yet implemented.

The LIMIT clause follows this syntax:

```
LIMIT number
```

The LIMIT clause limits the number of records that you want the query to return based on number.

Attribute access

The SELECT and WHERE clauses can refer to record data by using one of the methods in the following sections, depending on whether the file that is being queried is in CSV or JSON format.

CSV

- **Column Numbers** – You can refer to the Nth column of a row with the column name _N, where N is the column position. The position count starts at 1. For example, the first column is named _1 and the second column is named _2.

  You can refer to a column as _N or alias._N. For example, _2 and myAlias._2 are both valid ways to refer to a column in the SELECT list and WHERE clause.

- **Column Headers** – For objects in CSV format that have a header row, the headers are available to the SELECT list and WHERE clause. In particular, as in traditional SQL, within SELECT and WHERE clause expressions, you can refer to the columns by alias.column_name or column_name.

JSON

- **Document** - You can access JSON document fields as alias.name. You can also access nested fields, for example, alias.name1.name2.name3.
- **List** - You can access elements in a JSON list by using zero-based indexes with the [] operator. For example, you can access the second element of a list as alias[1]. You can combine accessing list elements with fields, for example, alias.name1.name2[1].name3.
- **Examples**: Consider this JSON object as a sample dataset:

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

  Example #1

  The following query returns these results:

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

  Example #2

  The following query returns these results:

```
Select s.projects[0].project_name from S3Object s
```

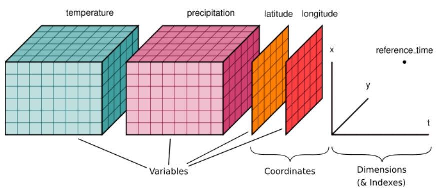
```
{"project_name":"project1"}
```

Parquet

Parquet fields are referred to by name.

NetCDF

NetCDF variables (including coordinate variables) as illustrated below are referred to by name only.



HDF5

Important: HDF5 is not yet implemented.

HDF5 variables are referred to by name.

Case sensitivity of header and attribute names

Like S3 Select, with AirMettle you can use double quotation marks to indicate that column headers (for CSV objects) and attributes (for JSON objects) are case sensitive. Without double quotation marks, object headers and attributes are case insensitive. An error is thrown in cases of ambiguity.

The following examples are either 1) Amazon S3 objects in CSV format with the specified column headers, and with FileHeaderInfo set to "Use" for the query request; or 2) Amazon S3 objects in JSON format with the specified attributes.

Example #1: The object being queried has the header or attribute NAME.

- The following expression successfully returns values from the object. Because there are no quotation marks, the query is case insensitive.

```
SELECT s.name from S3Object s
```

- The following expression results in a 400 error MissingHeaderName. Because there are quotation marks, the query is case sensitive.

```
SELECT s."name" from S3Object s
```

Example #2: The Amazon S3 object being queried has one header or attribute with NAME and another header or attribute with name.

- The following expression results in a 400 error AmbiguousFieldName. Because there are no quotation marks, the query is case insensitive, but there are two matches, so the error is thrown.

```
SELECT s.name from S3Object s
```

- The following expression successfully returns values from the object. Because there are quotation marks, the query is case sensitive, so there is no ambiguity.

```
SELECT s."NAME" from S3Object s
```

Using reserved keywords as user-defined terms

AirMettle has a set of reserved keywords that are needed to run the SQL expressions used to query object content. Reserved keywords include function names, data types, operators, and so on. In some cases, user-defined terms, such as the column headers (for CSV files) or attributes (for JSON objects), might clash with a reserved keyword. When this happens, you must use double quotation marks to indicate that you are intentionally using a user-defined term that clashes with a reserved keyword. Otherwise a 400 parse error will result.

For the full list of reserved keywords, see *Reserved keywords*.

The following example is either 1) an AirMettle object in CSV format with the specified column headers, with File-HeaderInfo set to "Use" for the query request, or 2) an AirMettle object in JSON format with the specified attributes.

Example: The object being queried has a header or attribute named CAST, which is a reserved keyword.

- The following expression successfully returns values from the object. Because quotation marks are used in the query, S3 Select uses the user-defined header or attribute.

```
SELECT s."CAST" from S3Object s
```

- The following expression results in a 400 parse error. Because no quotation marks are used in the query, CAST clashes with a reserved keyword.

```
SELECT s.CAST from S3Object s
```

Note: While similar, the list of reserved words in AirMettle is larger than S3 Select due to a larger feature set.

4.3.2 Data types

Name	Description	Examples
bool	A Boolean value, either TRUE or FALSE.	FALSE
int, integer	An 8-byte signed integer in the range -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.	12345
string	A UTF8-encoded variable-length string. The default limit is 1 character. The maximum character limit is 2,147,483,647.	'abc'
float	An 8-byte floating point number.	CAST(0.456 AS FLOAT)
decimal, numeric	A base-10 number, with a maximum precision of 38 (that is, the maximum number of significant digits), and with a scale within the range of -231 to 231-1 (that is, the base-10 exponent).	123.456
timestamp	Timestamps represent a specific moment in time, always include a local offset, and are capable of arbitrary precision.	CAST('2007-04-05T14:30Z' AS TIMESTAMP)

Literals

Strings are things in single quotes Anything without a decimal point is an int Anything with a decimal is a decimal TRUE and FALSE are boolean literals NULL

4.3.3 Operators

Math Operators

- - Unary minus
- +
- -
- *
- /
- %

Comparison Operators

- <
- >
- <=
- >=
- =
- !=
- <>
- Between (Inclusive)
- In (Set Membership)

Logical Operators

- And
- Or
- Not

Pattern Matching Operators

- Like (_ and %)

4.3.4 Reserved keywords

The following is the list of reserved keywords for Amazon S3 Select. These keywords include function names, data types, operators, and so on, that are needed to run the SQL expressions that are used to query object content.

```
ABSOLUTE
ACTION
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BAG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BLOB
BOOL
BOOLEAN
BOTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHAR_LENGTH
CHARACTER
CHARACTER_LENGTH
CHECK
CLOB
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
```

(continues on next page)

(continued from previous page)

```
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATE
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END
END_EXEC
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE_
FETCH
FIRST
FLOAT
FOR
FOREIGN
FOUND
```

(continues on next page)

(continued from previous page)

```
FROM
FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IMMEDIATE
IN
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEADING
LEFT
LEVEL
LIKE
LIMIT
LIST
LOCAL
LOWER
MATCH
MAX_
MIN_
MINUTE
MISSING
MODULE
MONTH
NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NO
NOT
```

(continues on next page)

(continued from previous page)

```
NULL_
NULLIF
NUMERIC
OCTET_LENGTH
OF
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PAD
PARTIAL
PIVOT
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
READ
REAL
REFERENCES
RELATIVE
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SESSION
SESSION_USER
SET
SEXP
SIZE
SMALLINT
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLSTATE
```

(continues on next page)

(continued from previous page)

```
STDDEV_SAMP
STRING
STRUCT
SUBSTRING
SUM
SYMBOL
SYSTEM_USER
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIM
TRUE_
TUPLE
UNION
UNIQUE
UNKNOWN
UNPIVOT
UPDATE
UPPER
USAGE
USER
USING
VALUE
VALUES
VARCHAR
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE
DATE_ADD
DATE_DIFF
TO_STRING
TO_TIMESTAMP
UTCNOW
REGEXP_CONTAINS
```

4.3.5 Functions

Conversion functions

- cast

Conditional functions

Important: Conditional functions are not yet implemented.

Temporal functions

- to_string
- to_timestamp
- utcnow
- date_add
- date_diff
- extract

String functions

Important: String functions are not yet implemented.

Pattern matching functions

- regexp_contains todo: Regex standard

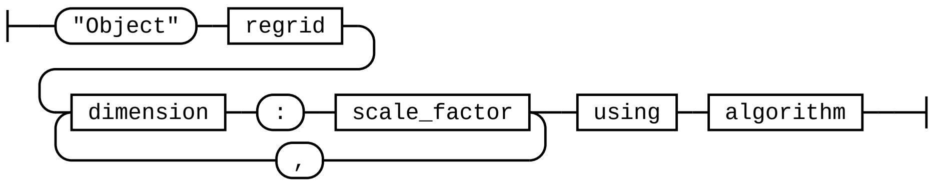
Aggregate functions

Function	Argument type	Return type	Description
AVG(express	INT, FLOAT, DECIMAL	DECIMAL for an INT argument, FLOAT for a floating-point argument; otherwise the same as the argument data type.	Average
COUNT	- or *	INT	Count (Returns 0 for 0 rows matched, while other aggregate functions return NULL)
MAX(expres	INT, DECIMAL	Same as the argument type.	Maximum
MIN(express	INT, DECIMAL	Same as the argument type.	Minimum
SUM(express	INT, FLOAT, DOUBLE, DECIMAL	INT for an INT argument, FLOAT for a floating-point argument; otherwise, the same as the argument data type.	Sum
STD-DEV_SAMP	INT, FLOAT, DOUBLE, DECIMAL	DECIMAL for an INT argument, FLOAT for a floating-point argument; otherwise the same as the argument data type.	Sample standard deviation

Regrid function

The regrid function is used to change the grid that underlies a data set. To do this input data is partitioned into blocks according to a given scale factor. A scale factor is specified for each dimension of the input data, indicating whether the data will be interpolated over a smaller or larger area.

```
SELECT * FROM
  Object regrid dimension1:scale_factor1,
                dimension2:scale_factor2[,...dimension3:scale_factor3]
    using algorithm
```



The scale factor is a positive number that determines the size of the data to generated in each dimension. A value > 1 expands values over a dimension, and a value < 1 aggregates values. Scale factors of 2 and 0.5 applied to a 10x10 2D input array, for example, will yield an output array of size 20x5.

An algorithm is specified to determine how the data in each block is aggregated or expanded. The algorithm may be an interpolation function intended to preserve the qualities of the original data or one yielding an entirely new data set.

Important: Regridding is implemented for multi-dimensional data (netCDF) only.

todo Detail the considerations between statistical and geospatial regridding (e.g. spherical vs. cartesian)

Algorithms

Sum

todo

Avg

todo

Bilinear

Bilinear interpolation computes the linear interpolation of the values in each block over 2 or 3 dimensions.

Patch

todo

Nearest source to destination

todo

Nearest destination to source

todo

First-order conservative

todo

Second-order conservative

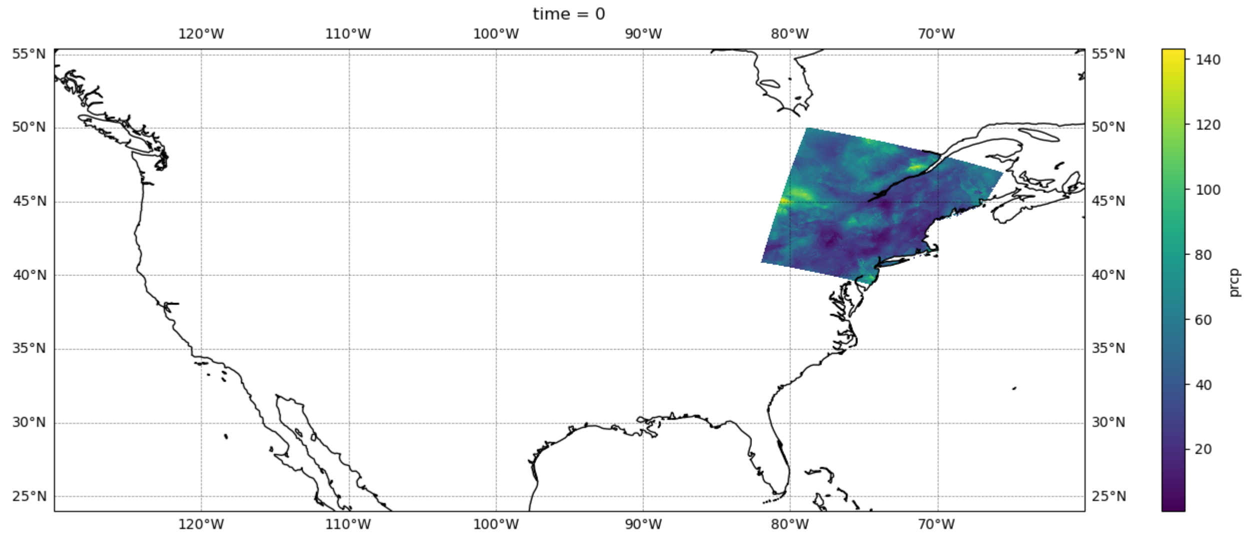
todo

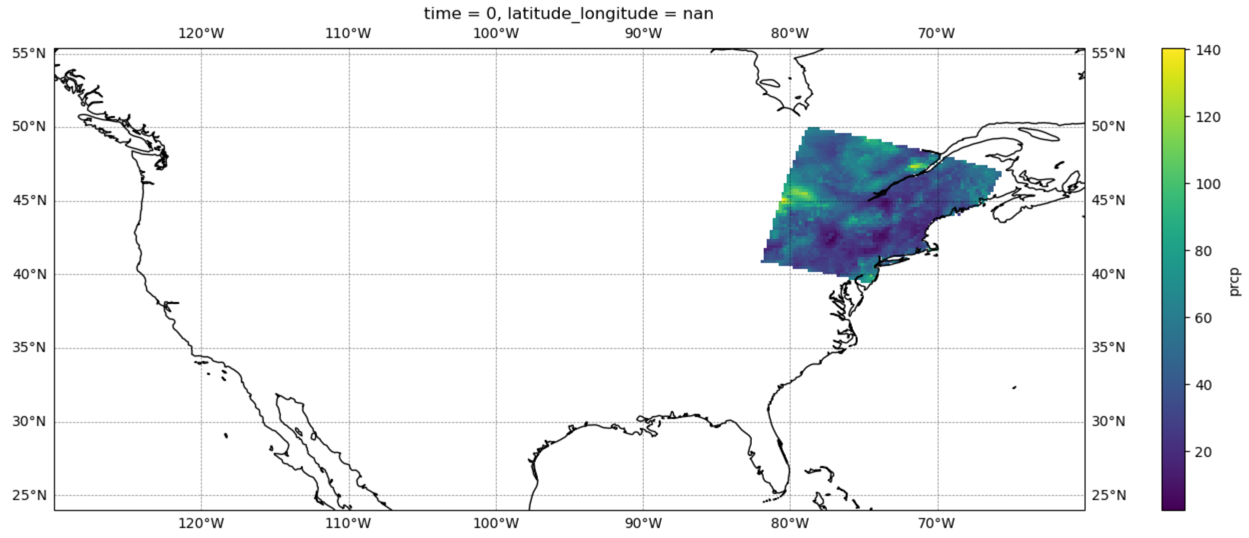
Example

Regrid precipitation data   This example demonstrates filtering a precipitation dataset by latitude and longitude, and regridding the data using the bilinear interpolation algorithm over blocks of size 100x45.

todo Detail mapping of lat/lon to x/y, use correct x,y and scale factors.

```
SELECT prcp
FROM Object REGRID x:100, y:45 USING Bilinear
WHERE
  y > 10 AND y < 20 AND
  x > 30 x < 40
```


time = 0
120°W
110°W
100°W
90°W
80°W
70°W
55°N
50°N
45°N
40°N
35°N
30°N
25°N
prcp
140
120
100
80
60
40
20


time = 0, latitude_longitude = nan
120°W
110°W
100°W
90°W
80°W
70°W
55°N
50°N
45°N
40°N
35°N
30°N
25°N
prcp
140
120
100
80
60
40
20

4.4 pyairmettle

`class pyairmettle.v1_0.flight.`**`ClientAuthHandler`**(*username*, *password*, *application*)

Implementation of Flight's ClientAuthHandler for authentication/authorization with a credential of 3 attributes.

`authenticate`(*self*, *outgoing*, *incoming*)

Conduct the handshake with the server.

Parameters

- **`outgoing`** (*`ClientAuthSender`*) – A channel to send messages to the server.
- **`incoming`** (*`ClientAuthReader`*) – A channel to read messages from the server.

`get_token`(*self*)

Get the auth token for a call.

`class pyairmettle.ipc.`**`MultiStreamReader`**(*input_stream*)

A reader that can read an Arrow multi stream (multiple concatenated individual Arrow IPC streams).

`read_all`()

Reads all the tables in the stream

Returns

`read_next`()

Reads the next table from the stream

Returns

CHAPTER

FIVE

DEPLOYMENT

5.1 Kubernetes

AirMettle is delivered as a docker container image tarball.

There are instructions for the helm chart are at https://airmettle.github.io/airmos-helm-charts/.

5.2 `airmettle` Binary

AirMettle is a single binary that may be started by running this command:

```
./airmettle -f airmettle.properties
```

`airmettle.properties` is a configuration file detailed below.

When running the `airmettle` binary on the command line, these are the options it supports:

Argument	Description
`-f <path>`	Path to `airmettle.properties` file
`-p <path>`	Path to file to store process id in
`-d`	Run `airmettle` in background
`-m`	`airmettle` runs as a monitor process that forks a child to run the server. Whenever the child process exits, a new one is created. This is useful if you want airmos to automatically restart if there are any crashes.
`-i <node_na`	configure the name of the node in the cluster instead of using `airmos.instance` from `airmettle.properties`
`-a`	Turn off any access control lists. This can be used if you accidentally set an IP ACL in the UI that blocks access.

5.2.1 airmettle.properties

The base `airmettle.properties` file contains these configuration parameters that can be modified.

```
# The instance acts as a node that can store files
node_server=true
# The instance acts as a node that can run admin commands
admin_server=true

# The instance can run administration jobs used to move and rebalance files
run_admin_jobs=true

# The http port the instance listens on
http_port=8080

# The ipc port used for internal communication the instance listens on
ipc_port=8082

# Sets the instance of the node to its IP.
# If this isn't set, gethostname is used.
airmos.instance=##NAME_OR_IP_OF_THIS_HOST##:8080
ipc_use_gethostname=false

# This optimization could also be enabled.  It sends the replicate commands over the IPC␣
↪channels
# use_ipc_for_internal = true

# Directory for web console UI pages
static_page_dir=/data/servers/airmettle/web

# Initial login for admin user
admin_user=admin
admin_password=amospass

# Directory to store database files
db_dir=/data/servers/airmettle/db

# IP:PORT of node with configuration database.  This may be comma separated list if the␣
↪database is replicated
config_hosts=##CHOOSE_HOST##:8082

# IP:PORT of node with metadata database.  This may be comma separated list if the␣
↪database is sharded over multiple hosts
meta_hosts=##CHOOSE_HOST##:8082

# Directory to store log files
log_directory=/data/servers/airmettle/logs

# Directory to store phone home crash and diagnostic files
phone_home_dir=/data/servers/airmettle/phone_home

# log level
log_level=DEBUG
```

(continues on next page)

(continued from previous page)

```
# ID and name of cluster.   It must be unique if connecting multiple clusters
cluster_name=demo
cluster_id=1

# Rocksdb configuration values to limit write ahead log size
rocksdb_wal_size_limit=4096
rocksdb_wal_ttl_seconds=86400

# Optional ssl cert and key files so server can support https
# ssl_server_cert_file=/path_to_server_cert
# ssl_server_key_file=/path_to_server_key
```

5.3 Node Differentiation

AirMettle instances can be run either symmetrically or specialized for front-end (user request handling) and back-end (storage and maintenance) operation.

todo: How

CHAPTER

SIX

UI CONSOLE

The UI console can be reached at the port `http_port` as set in the default `airmettle.properties`.

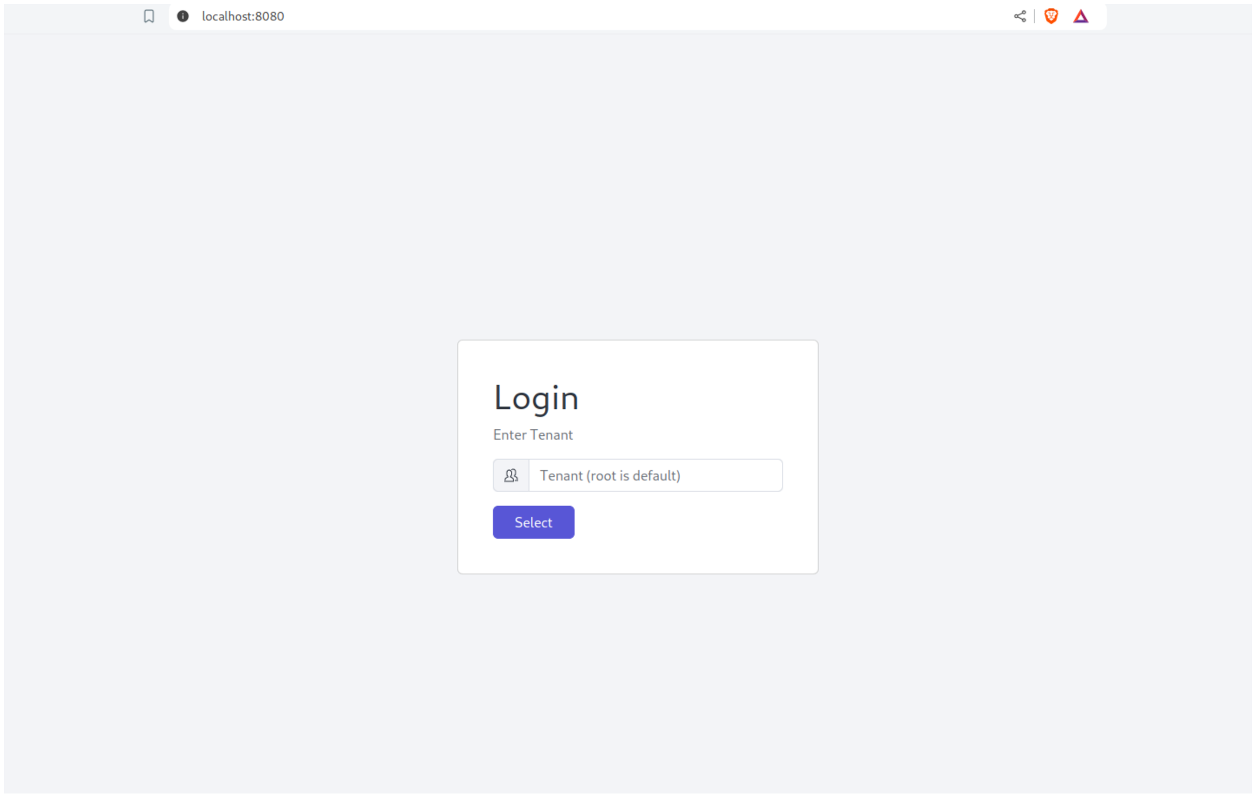
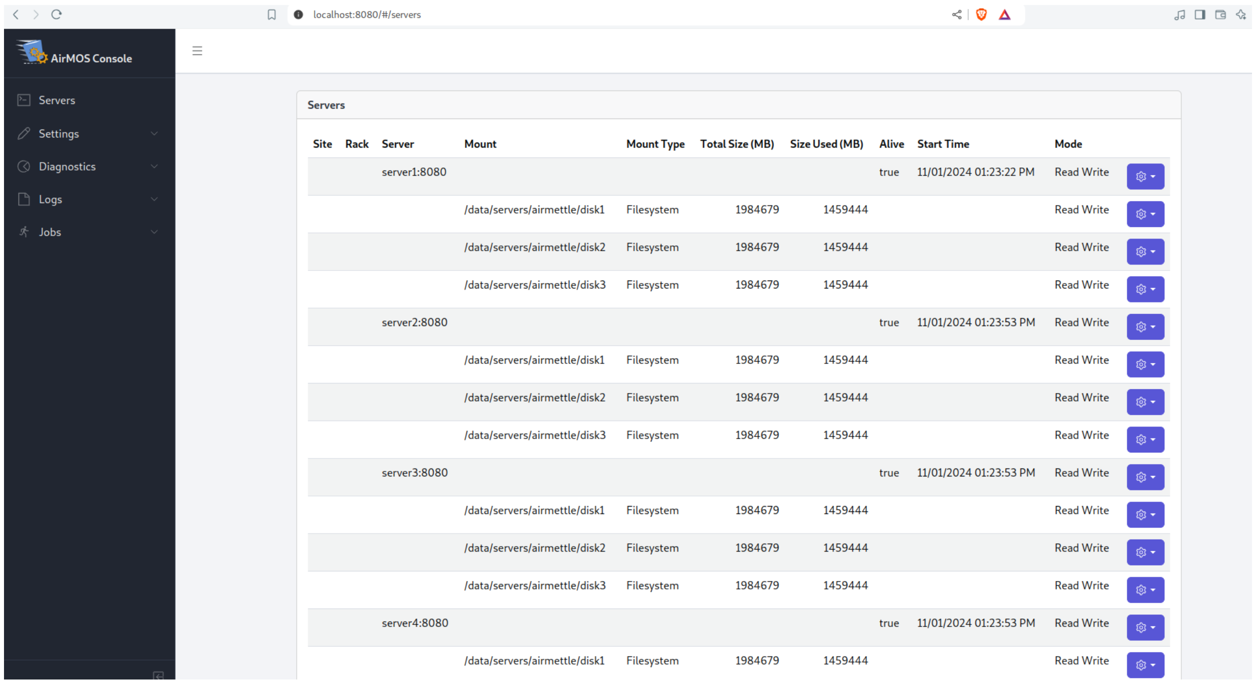


Fig. 1: Login Page

The default superadmin credentials are specified by `admin_user` and `admin_pass` in the properties file. The default landing page shows a menu on the left and the current status of servers and storage devices in the cluster.

The next chapters will describe the administrative features of AirMettle. All functions that are provided by the UI can also be performed via the `/admin` API interface.



Servers

Site	Rack	Server	Mount	Mount Type	Total Size (MB)	Size Used (MB)	Alive	Start Time	Mode
		server1:8080					true	11/01/2024 01:23:22 PM	Read Write
			/data/servers/airmettle/disk1	Filesystem	1984679	1459444			Read Write
			/data/servers/airmettle/disk2	Filesystem	1984679	1459444			Read Write
			/data/servers/airmettle/disk3	Filesystem	1984679	1459444			Read Write
		server2:8080					true	11/01/2024 01:23:53 PM	Read Write
			/data/servers/airmettle/disk1	Filesystem	1984679	1459444			Read Write
			/data/servers/airmettle/disk2	Filesystem	1984679	1459444			Read Write
			/data/servers/airmettle/disk3	Filesystem	1984679	1459444			Read Write
		server3:8080					true	11/01/2024 01:23:53 PM	Read Write
			/data/servers/airmettle/disk1	Filesystem	1984679	1459444			Read Write
			/data/servers/airmettle/disk2	Filesystem	1984679	1459444			Read Write
			/data/servers/airmettle/disk3	Filesystem	1984679	1459444			Read Write
		server4:8080					true	11/01/2024 01:23:53 PM	Read Write
			/data/servers/airmettle/disk1	Filesystem	1984679	1459444			Read Write

Fig. 2: Landing Page

CHAPTER

SEVEN

SERVERS

The servers section contains all the nodes in the cluster. It allows adding drives to the servers and marking which are available.

When a new node is started, it will automatically add itself to the list of servers. If a host needs to be explicitly added, use the API `/admin/addServer`. You can then go into the UI and add its drives or add drives via `/admin/addDrive`.

Servers and drives have four statuses they can be set to:

- Read/Write - Files can be uploaded and fetched
- Read Only - Existing files can only be fetched. No new files will be stored.
- Offline - Files cannot be uploaded or fetched.
- Maintenance - Files can only be fetched for maintenance operations such as migrating files.

They can be set by the corresponding APIs: `/admin/changeServerStatus` and `/admin/changeDriveStatus`.

If a node has no available drives, it will proxy new upload requests to other nodes in the cluster. This is equivalent to a frontend node configuration.

CHAPTER

EIGHT

SETTINGS

8.1 Tenants

The tenants section is used to create different tenants in the cluster and set the configuration for each. A tenant is also called an application. Configuration includes storage and replication policies. Use the API calls `/admin/addApplication` and `/admin/deleteApplication`.

The Authentication provider for each tenant can be set to AirMOS or OpenID Connect. When it is set to AirMOS, authentication credentials are stored in the local airmettle database. When it is set to OpenID Connect, you configure airmettle to use an external IDP to authenticate the user logging in. The OpenID Connect APIs include: `/admin/oidcStart`, `/admin/oidcAssumeRole`, `/admin/oidcCheckApplication`, and `/admin/oidcComplete`.

8.2 Accounts

The accounts section allows for the creation and management of user accounts for each tenant. This includes API keys used to run S3 APIs.

The corresponding account management APIs include: `/admin/addAccount`, `/admin/fetchAccounts`, `/admin/updateAccount`, and `/admin/deleteAccount`

8.3 Configuration

The configuration section contains variables that can be dynamically changed. They are stored in the configuration database. When a variable is changed, it is reloaded in all the nodes in the cluster so changes will be made in all nodes.

Configuration that cannot be dynamically changed needs to be placed in the `airmettle.properties` file and is loaded at startup. If dynamic variables are placed in `airmettle.properties`, the values read from the file override the global dynamic value.

APIs for managing configuration include: `/admin/addGlobalConfig`, `/admin/fetchConfig`.

8.4 IP Access Control

The IP access control section allows for the setting of different IP Access Control Lists. IP Addresses can be entered as an address or in CIDR notation (10.0.0.0/8) to denote a range.

The supported IP Access Lists IP addresses can be added to are:

Trust XFF Headers
This ACL is used to process the X-Forwared-For header in a request. It should contain proxy IP address that are trusted. If the list is empty, all IPs are trusted.

AirMOS Nodes
List of IPs for airmettle nodes used to restrict internal nodes for proxying and migrating files. If the list is empty, all IPs are allowed.

CDN IPs
If a tenant (application) is set with force cdn as true, this ACL is used to make sure fetch object requests are coming from the CDN

Admin UI Allowed
List of IPs allowed to access admin UI. If the list is empty, all IPs can access it.

Admin Trusted
List of IPs allowed to access admin API without a token. If the list is empty, all IPs can access it.

APIs for these functions include: `/admin/addIP`, `/admin/fetchAcls`, `/admin/deleteIP`.

8.5 File Integrity

The File Integrity section is used to configure the policy for checking if files have been corrupted. File Integrity checking consists of hashing the contents of the files to make sure they match the hash that was stored when the file was first uploaded. You can configure when the check is done and under what conditions it is done.

The method for a rule determines when the check is done:

- GET: On every file fetch
- Skulker: When background checker runs against a file
- Migrate: When file is migrated
- All: All above cases

The factor determines which files are done:

- Size over: when file is over a certain size
- Size under: when file is under a certain size
- Type contains: when content type of file matches one of values in list
- Type not contains: when content type of file matches no values in list
- Always: for every file regardless of size or content type

A file integrity policy can be set to be global to apply to all tenants or can be individually assigned to a tenant in the tenant configuration section.

The corresponding APIs include: `/admin/addIntegrityPolicy`, `/admin/addIntegrityPolicyRule`, `/admin/listIntegrityPolicies`, `/admin/getIntegrityPolicy`, `/admin/updateIntegrityPolicy`, `/admin/updateIntegrityPolicyRule`, `/admin/deleteIntegrityPolicy`, `/admin/deleteIntegrityPolicyRule`

8.6 Alerts

The Alerts section is used to configure alerts that should fire for different events within the cluster. You can configure the alert type such as drive errors and at what threshold to fire the alert. Alerts can be sent over SNMP or email. Email and SNMP destinations are configured with alerts_smtp and snmp configuration parameters.

The corresponding APIs include: `/admin/addAlertPolicy`, `/admin/listAlertPolicies`, `/admin/getAlertPolicy`, `/admin/updateAlertPolicy`, `/admin/deleteAlertPolicy`

8.7 DB Shard Map

The DB Shard Map section is used to configure the configuration and metadata database that runs with the cluster on the nodes. It allows for sharding the database and replicating to new nodes.

There are three types of databases:

Config
configuration data such as configuration variables, tenants, etc.

Metadata
metadata for raw file objects including data on where parts and replicas are stored. It does not contain the file contents.

File Group
bucket and object data including mapping to Metadata objects.

All database types support replication. Metadata and File Group data is sharded. Config data is not.

Adding replicas for the configuration database must be done by editing `airmettle.properties` and setting `config_hosts` to a comma separated list of all the hosts for the configuration database. After this is changed, all nodes must be restarted.

Adding replicas and shards for the Metadata and File Group database is done through the Admin UI (and corresponding APIs) and can be done dynamically without needing to restart nodes. The UI lets you add new shards and add new hosts to a shard. A shard consists of multiple hashes. Data is migrated between shards by moving the hashes between the shards using the UI. There is a default of 32768 hashes for each database type.

Replica and shard APIs include: `/admin/addShard`, `/admin/addShardHost`, `/admin/fetchShardMap`, `/admin/migrateShardMap`, `/admin/cancelMigrateShardMap`, `/admin/redistributeShardMap`, `/admin/updateShardHost`, `/admin/deleteShard`, `/admin/deleteShardHost`

8.8 Clusters

The cluster section is used to configure file replication between multiple airmettle clusters. Replication between clusters works by replicating storage objects. Each cluster has its own database and replication is not done at the database level.

To setup replication between clusters:

1. In the `airmettle.properties` file each cluster uses, make sure each has a unique `cluster_name` and `cluster_id`
2. In the Admin UI Cluster Section for each cluster, Add the id/name of the other cluster (`/admin/addCluster`)
3. For each cluster you add, one one or more hosts from that cluster in the Admin UI Cluster Section (`/admin/addClusterHost`). The replication stream will be distributed over those hosts.

The cluster management APIs include: `/admin/addCluster`, `/admin/addClusterHost`, `/admin/fetchClusters`, `/admin/fetchClusterStates`, `/admin/updateCluster`, `/admin/deleteCluster`, `/admin/deleteClusterHost`.

CHAPTER

NINE

DIAGNOSTICS

9.1 Overview

Graphs are displayed showing the CPU and Memory usage on the node. The data is only from the node the UI fetches the data from. It is not aggregated across the cluster.

9.2 Timings

Request times are shown for various operations. The data is only from the node the UI fetches the data from. It is not aggregated across the cluster. Airmettle also has the ability to use open telemetry to send timing data to an open telemetry collector.

9.3 Counters

Counters and errors are shown for various operations. The data is only from the node the UI fetches the data from. It is not aggregated across the cluster. Airmettle also has the ability to use open telemetry to send timing data to an open telemetry collector.

9.4 Connections

Incoming and outgoing HTTP and IPC connections are shown. IPC connections are internal connections between nodes. The data is only from the node the UI fetches the data from. It is not aggregated across the cluster.

9.5 Threads

The different thread pool and number of threads in use are shown. The data is only from the node the UI fetches the data from. It is not aggregated across the cluster. This does not include threads used by the S3 select query engine.

CHAPTER

TEN

JOBS

10.1 Running Jobs

This allows for the creation of migrations and population jobs on the cluster. Migration jobs move files off of a server and device. Population jobs populate a new drive with files from other drives. Jobs are divided into tasks so they can run in parallel on different nodes.

10.2 Job Reports

This allows for viewing jobs from “Running Jobs” that have completed. Failed tasks in jobs can be retried or the error can be viewed for why the job task failed.

10.3 Skulk Reports

This displays the output of skulk jobs that run in the background to fix files on the cluster. Skulking can be enabled by setting `background_file_check` to `true` in the configuration section. `max_background_skulks` and `skulk_delay` can be used to control how many files the skulk checking does in parallel. The output in the skulk reports shows any error fixing files during skulking. Skulking checks file hashes to make sure files are not corrupt. It also makes sure the right number of replicated copies exist. It will attempt to repair any issues it finds.

CHAPTER

ELEVEN

ADMINISTRATIVE WEB API

11.1 /admin/addAccount

Adds an account to an application (tenant).

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
apiKey	S3 API Key for Account	String
s3Secret	S3 Secret for Account	String
password	Password for Account	String
name	Username for account	String
superUser	Super User Type	String value: NONE, SUPERUSER, or SUPERUSER_READ_ONLY
ownerApplication	ID of application (tenant) account belongs to	Numeric
accountRole	Role in application (tenant)	String value: NONE, READ_ONLY, READ_WRITE, ADMIN

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.2 `/admin/addAlertPolicy`

Adds an alert policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Alert Type	String value: DRIVE_ERRORS, SKULK_ERRORS, SERVERS_DOWN, SERVERS_UP, WRITE_OBJECTS, READ_OBJECTS, LOGIN_FAILURES, REPLICATION_ERRORS, MD5_ERRORS, BYTES_IN_HIGH, BYTES_OUT_HIGH, SWIFT_S3_AUTH_ERRORS
`alertValue`	Threshold Value	Number
`alertValueTime`	Threshold Time Period (seconds)	Number
`destination`	Alert Destination	String value: EMAIL, SNMP

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.3 `/admin/addApplication`

Adds an application (tenant).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`name`	Tenant name	String
`s3HostnamePostfix`	S3 Hostname	String
`maxUploadSize`	Max Upload (MB)	Number
`fileSplitSize`	Split Size (MB)	Number
`expireTime`	Expire Time (sec)	Number
`integrityPolicyId`	ID of File Integrity Policy	Number
`totalCopies`	Total Copies	Number
`numSites`	Num Sites	Number
`forceCdn`	Only allow request to come from CDN ACL	Boolean (1 or 0)
`useCache`	Use Cache	Boolean (1 or 0)
`erasureCodingType`	Erasure Coding Type	String value: NONE, CAUCHY_GOOD
`erasureCodingK`	Erasure Coding K Value	Number

continues on next page

Table 1 – continued from previous page

Parameter	Description	Type
erasureCodingN	Erasure Coding N Value	Number
compressionType	Compression Type	String value: NONE, SNAPPY, GZIP
encryptionOn	Encryption is On	Boolean (1 or 0)
extractArchive	Extract files from archive file uploads	Boolean (1 or 0)
versionObjects	Version Objects	Boolean (1 or 0)
maxObjectVersions	Maximum number of versions to store	Number
encryptionNewKeys	Generate new encryption keys	Boolean (1 or 0)
expireFiles	Expire Files	Boolean (1 or 0)
masterOnNotFound	Use Master DB if entry not found for file	Boolean (1 or 0)
oidcClientId	OIDC (OpenID Connect) Client ID	String
oidcSecret	OIDC Client Secret	String
oidcDiscoveryUrl	OIDC Discovery URL	String
oidcScopes	OIDC Scopes (comma separated)	String
oidcAutoCreateAccounts	OIDC Auto Create Accounts on Login	Boolean (1 or 0)
oidcUsePkce	OIDC Use PKCE	Boolean (1 or 0)
oidcUsernameClaim	OIDC Username Claim in JWT	String
oidcAllowedDomains	OIDC Allowed Domains (comma separated)	String
authProvider	Authentication Provider Type	String value: LOCAL, OIDC
url	URL for information purposes	String
email	Email contact	String
comment	Comment for information purposes	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.4 /admin/addCluster

Adds a cluster. Clusters are used to have different AirMOS clusters replicate to each other.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
id	Unique id of cluster	Number
name	Name of cluster	String

Response Example:

```
{
    "AdminResponse": {
```

(continues on next page)

(continued from previous page)

```
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.5 /admin/addClusterHost

Adds a host endpoint to send the replication stream to another cluster.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
id	Unique id of cluster	Number
hostname	FQDN of host to connect to	String
httpPort	HTTP port of host	Number
ipcPort	IPC port of host	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.6 /admin/addGlobalConfig

Update configuration value

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
name	Name of variable	String
value	Value for variable	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
```

(continues on next page)

(continued from previous page)

```
        "statusText": "Success"
    }
}
```

11.7 /admin/addDrive

Adds a drive to a host.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
hostname	Hostname	String
mount	If type =1, mount path of drive If type =2, name of spdk blobstore device	String
type	Type of device	Number value: • 0; Filesystem • 1: Blobstore

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.8 /admin/addDriveAll

Use the same parameters to add drives to all hosts in the cluster.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
mount	If type=1, mount path of drive If type=2, name of spdk blobstore device	String
type	Type of device	Number value: • 0: Filesystem • 1: Blobstore

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.9 /admin/addIntegrityPolicy

Creates a new file integrity policy. The id of the created policy can then be used in an application (tenant).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyName`	Name for policy	String
`isGlobal`	Indicates if policy is default	Boolean (1 or 0)

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.10 /admin/addIntegrityPolicyRule

Add a rule to a file integrity policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`factor`	Indicates when an integrity check is done. It can be based on file size or content type.	String value: size over, size under, type contains, type not contains, always
`method`	Which operation the rule is applied to for when that operation should do an integrity check. • GET: on file get requests • skulker: when files are checked in background • migrate file: when files are migrated • all: all of the above	String value: GET, skulker, all, migrate file
`value`	Value for factor	String: bytes for `factor` = size over or size under. comma separated list of content types for type not contains or type contains
`policyId`	ID of policy to add rule to	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.11 `/admin/addIP`

Add an IP to an IP Access Control List.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`address`	IP Address	String. IP Address can be entered as an address or in CIDR notation (10.0.0.0/8) to denote a range.
`type`	IP Acl Type • xff: Trust XFF Headers' • internal: AirMOS nodes • cdn: CDN IPs • adminui: Admin UI Allowed • admintrusted: Admin Trusted • debugpage: Debug Page Allowed	String value: xff, internal, cdn, adminui, admintrusted, debugpage
`appNames`	Optional application (tenant) names, comma separated if the IP should only apply to certain applications (tenants)	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.12 /admin/addJob

Add a job to move files.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Type of job • moveDevice: migrate data off a failed drive • populateDevice: migrate data to a new drive	String value: moveDevice or populateDevice
`hostname`	Host to move data from or move data to	String
`mount`	Drive to move data from or move data to	String
`jobName`	Optional name for job	String
`size`	If job is populateDevice, the amount of bytes to move	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.13 /admin/addServer

Add a host to the cluster.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`hostname`	FQDN:http_port of server to add	String
`rack`	Optional rack name server is one	String
`site`	Optional site name server is at	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.14 /admin/addShard

Add an empty metadata shard to the database configuration. Once an empty shard is created, data can be assigned to it with `/admin/migrateShardMap`.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard type	String value: file_group or meta

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
```

(continues on next page)

(continued from previous page)

```
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.15 /admin/addShardHost

Adds a node to a shard for storing data for the shard. The node can serve as both replica and primary for the shard.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard type	String value: file_group or meta
`shardId`	ID of shard to add host to	String
`site`	Optional site of host	String
`rack`	Optional rack of host	String
`host`	Host FQDN	String
`port`	Host IPC Port	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.16 /admin/cancelMigrateShardMap

Cancel a request created from a call to `/admin/migrateShardMap`.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard type	String value: file_group or meta
`migrateId`	Migration ID to cancel	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
```

(continues on next page)

(continued from previous page)

```
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.17 /admin/changeDriveStatus

Change the status of a drive.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`hostname`	hostname	String
`mount`	Mount path of drive	String
`status`	Status to change drive to	String value: OFFLINE, RW, RO, MAINT
`submitJob`	If marking drive offline, submit job to move files off of it	Boolean (1 or 0)
`applyAll`	Apply change to all drives	Boolean (1 or 0)

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.18 /admin/changeServerStatus

Change the status of a server.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`hostname`	hostname	String
`status`	Status to change drive to	String value: OFFLINE, RW, RO, MAINT
`submitJob`	If marking server offline, submit job to move files off of it	Boolean (1 or 0)
`applyAll`	Apply change to all drives	Boolean (1 or 0)

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.19 /admin/checkSession

Check whether the session is logged in. The session cookie and bearer token in the Authorization header is checked.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "superUser": "SUPERUSER",
        "adminNode": true,
        "userName": "admin",
        "token": "3rjOPK..."
    }
}
```

11.20 /admin/clearAccessLog

Erase the access log file on disk.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
```

(continues on next page)

(continued from previous page)

```
        "statusText": "Success"
    }
}
```

11.21 /admin/clearAccessMemoryLog

Erase the access log in memory.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.22 /admin/clearErrorLog

Erase the error log file on disk.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.23 /admin/clearErrorMemoryLog

Erase the error log in memory.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.24 /admin/deleteAccount

Delete an account from an application (tenant).

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
id	Account ID	Number
name	Account username	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.25 /admin/deleteAlertPolicy

Delete an alert policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyId`	Policy ID	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.26 /admin/deleteApplication

Delete an application (tenant).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`id`	Application ID	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.27 /admin/deleteCluster

Delete a cluster configuration for a remote cluster.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
id	ID of cluster	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.28 /admin/deleteClusterHost

Delete a cluster host from a cluster configuration.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
id	ID of cluster	Number
hostname	hostname	String
ipcPort	IPC port of host	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.29 /admin/deleteDrive

Remove a drive from a host. This can only be done if all files have been migrated off of the drive.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`hostname`	hostname	String
`mount`	mount path	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.30 /admin/deleteDriveAll

Remove drives from all hosts. This can only be done if all files have been migrated off of the drives.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`mount`	mount path	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.31 /admin/deleteErrorAlerts

Clear error alerts shown in the UI under Logs / Config Errors.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
startId	Optional record id to start delete from	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.32 /admin/deleteIP

Delete an IP Address from an IP Access Control List.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
address	IP Address	String. IP Address can be entered as an address or in CIDR notation (10.0.0.0/8) to denote a range.
type	IP Acl Type • xff: Trust XFF Headers' • internal: AirMOS nodes • cdn: CDN IPs • adminui: Admin UI Allowed • admintrusted: Admin Trusted • debugpage: Debug Page Allowed	String value: xff, internal, cdn, adminui, admintrusted, debugpage

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.33 /admin/deleteIntegrityPolicy

Delete a file integrity policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyId`	ID of policy	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.34 /admin/deleteIntegrityPolicyRule

Delete a file integrity policy rule.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyId`	ID of policy	Number
`ruleId`	ID of rule	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.35 /admin/deleteJobReport

Delete the reports status that is generated by a migration job.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`jobName`	Name of job	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.36 /admin/deleteJobTask

Stop a task that is running as part of a migration job. A job consists of multiple tasks.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`name`	Name of job	String
`id`	ID of task	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.37 /admin/deleteJobTasks

Stop all tasks that are running as part of a migration job. A job consists of multiple tasks.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`name`	Name of job	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.38 /admin/deleteServer

Delete a server from the cluster. The server must have no drives attached to it to be deleted.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`hostname`	hostname	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.39 /admin/deleteShard

Delete a shard from the database config. The shard must have no data assigned to it.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
type	Shard type	String value: file_group or meta
shardId	ID of shard	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.40 /admin/deleteShardHost

Removes a node from a shard for storing data for the shard.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
type	Shard type	String value: file_group or meta
shardId	ID of shard	Number
host	Host FQDN	String
port	Host IPC Port	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.41 /admin/deleteSkulkReport

Removes report status that are collected as part of background skulk checking of files. Reports can be deleted either by passing `reportId` or by passing `date`, `server`, or `device`.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`reportId`	ID of report to delete	Number
`date`	Report date (seconds since unix epoch)	Number
`server`	Server ID report is for	Number
`device`	Device ID report is for	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.42 /admin/dumpStats

Creates a bundle of stats for the system and uploads to the phone home endpoint the system is configured to use.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.43 /admin/fetchAccessMemoryLog

Fetches access logs that are stored in memory.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
raw	Return data as an attachment	Boolean (0 or 1)
limit	Max rows to fetch	Number

Response Example:

```
{
    "LogsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "logs": [
            {
                "log": "- - 10.0.0.18 [30\/Dec\/2023:19:12:12 +0000] \"GET \/ HTTP\/1.1\
↪" 304 0 12 0\n"
            }
        ]
    }
}
```

11.44 /admin/fetchAccounts

Fetches accounts.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AccountsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "accounts": [
            {
                "id": "17",
                "name": "admin",
                "applicationOwnerId": "1",
                "applicationOwnerName": "root",
```

(continues on next page)

(continued from previous page)

```
                "accountRole": "ADMIN",
                "apiKey": "root-o4jg8V8Pl2MAkdk",
                "hasS3Secret": true,
                "hasPassword": true,
                "passwordLocation": "LOCAL",
                "uid": 0,
                "superUser": "SUPERUSER",
                "noChangeRole": true
            }
        ]
    }
}
```

11.45 /admin/fetchAcls

Fetches IP Access Control Lists.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml

Response Example:

```
{
    "AclsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "acls": [
            {
                "type": "writer",
                "address": "1.2.3.4",
                "applications": ""
            }
        ]
    }
}
```

11.46 /admin/fetchApplication

Fetch an application (tenant).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`name`	Application name	String

Response Example:

```
{
    "AppsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "apps": [
            {
                "id": 1,
                "name": "root",
                "maxUploadSize": 0,
                "fileSplitSize": 0,
                "expireTime": 0,
                "totalCopies": 3,
                "url": "",
                "email": "",
                "forceCdn": false,
                "comment": "",
                "redirectSize": 0,
                "useCache": false,
                "expireFiles": false,
                "masterOnNotFound": false,
                "numSites": 1,
                "integrityPolicyId": 0,
                "personalityCalypso": true,
                "personalitySwift": true,
                "personalityS3": true,
                "personalityFS": true,
                "erasureType": "NONE",
                "erasureK": 0,
                "erasureN": 0,
                "keystoneUrl": "",
                "s3HostnamePostfix": "",
                "compressionType": "NONE",
                "encryptionOn": false,
                "encryptionKeyDate": 0,
                "gid": 0,
                "extractArchive": false,
                "versionObjects": false,
                "maxObjectVersions": 0,
                "authProvider": "LOCAL",
                "oidcClientId": "",
                "oidcClientSecret": "",
                "oidcDiscoveryUrl": "",
                "oidcScopes": "",
                "oidcAutoCreateAccounts": false,
                "oidcUsernameClaim": "",
```

(continues on next page)

(continued from previous page)

```
                "oidcAllowedDomains": "",
                "oidcUsePkce": false
            }
        ]
    }
}
```

11.47 /admin/fetchApplications

Fetch all applications (tenants).

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AppsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "apps": [
            {
                "id": 1,
                "name": "root",
                "maxUploadSize": 0,
                "fileSplitSize": 0,
                "expireTime": 0,
                "totalCopies": 3,
                "url": "",
                "email": "",
                "forceCdn": false,
                "comment": "",
                "redirectSize": 0,
                "useCache": false,
                "expireFiles": false,
                "masterOnNotFound": false,
                "numSites": 1,
                "integrityPolicyId": 0,
                "personalityCalypso": true,
                "personalitySwift": true,
                "personalityS3": true,
                "personalityFS": true,
                "erasureType": "NONE",
                "erasureK": 0,
                "erasureN": 0,
                "keystoneUrl": "",
```

(continues on next page)

(continued from previous page)

```
                "s3HostnamePostfix": "",
                "compressionType": "NONE",
                "encryptionOn": false,
                "encryptionKeyDate": 0,
                "gid": 0,
                "extractArchive": false,
                "versionObjects": false,
                "maxObjectVersions": 0,
                "authProvider": "LOCAL",
                "oidcClientId": "",
                "oidcClientSecret": "",
                "oidcDiscoveryUrl": "",
                "oidcScopes": "",
                "oidcAutoCreateAccounts": false,
                "oidcUsernameClaim": "",
                "oidcAllowedDomains": "",
                "oidcUsePkce": false
            }
        ]
    }
}
```

11.48 /admin/fetchClusterStates

Fetch state of local and remote cluster synchronization for display in the Cluster section of Administration UI.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "ClustersResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "clusters": [
            {
                "id": 1,
                "name": "1",
                "local": true,
                "hosts": [],
                "segments": [
                    {
                        "id": 17,
                        "serverId": 17,
                        "serverName": "localhost:8010",
```

(continues on next page)

(continued from previous page)

```
                        "updateDate": 1701867620,
                        "lastVersion": "0",
                        "lastVersionTimestamp": 0,
                        "masterVersion": "0",
                        "masterVersionTimestamp": 0,
                        "repState": "NONE",
                        "dataType": "",
                        "objectType": "",
                        "destClusterId": 0
                    }
                ]
            }
        ]
    }
}
```

11.49 /admin/fetchClusters

Fetch local and remote cluster configuration for display in the Cluster section of Administration UI.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml

Response Example:

```
{
    "ClustersResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "clusters": [
            {
                "id": 1,
                "name": "1",
                "local": true,
                "hosts": [],
                "segments": [
                    {
                        "id": 17,
                        "serverId": 17,
                        "serverName": "localhost:8010",
                        "updateDate": 1701867620,
                        "lastVersion": "0",
                        "lastVersionTimestamp": 0,
                        "masterVersion": "0",
                        "masterVersionTimestamp": 0,
                        "repState": "NONE",
```

(continues on next page)

(continued from previous page)

```
                        "dataType": "",
                        "objectType": "",
                        "destClusterId": 0
                    }
                ]
            }
        ]
    }
}
```

11.50 /admin/fetchConfig

Fetch configuration variables.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`category`	Optional category to filter on	String

Response Example:

```
{
    "ConfigsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "categories": [
            {
                "name": "Background Processing",
                "configs": [
                    {
                        "name": "autoheal_migrate_min_time",
                        "value": "86400",
                        "description": "Seconds before automigrating files (0 = never)",
                        "type": 0,
                        "min": 0,
                        "max": 4294967295,
                        "pattern": ""
                    },
                    {
                        "name": "autoheal_migrate_startup_time",
                        "value": "300",
                        "description": "Seconds before starting automigrating checks␣
↪after startup",
                        "type": 0,
                        "min": 0,
                        "max": 4294967295,
                        "pattern": ""
```

(continues on next page)

(continued from previous page)

```
                    }
                ]
            }
        ]
    }
}
```

11.51 /admin/fetchDBStats

Fetch raw database stats from RocksDB.

Response Example:

```
**** config - default ****

** Compaction Stats [default] **
Level    Files   Size     Score Read(GB)  Rn(GB) Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-
↪Amp Rd(MB/s) Wr(MB/s) Comp(sec) CompMergeCPU(sec) Comp(cnt) Avg(sec) KeyIn KeyDrop␣
↪Rblob(GB) Wblob(GB)
-----------------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------------
↪----------------
 Sum      0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0      0.0       0.0  ␣
↪0.0      0.0      0.0      0.00              0.00         0    0.000       0      0   ␣
↪   0.0       0.0
 Int      0/0    0.00 KB   0.0      0.0     0.0      0.0       0.0      0.0       0.0  ␣
↪0.0      0.0      0.0      0.00              0.00         0    0.000       0      0   ␣
↪   0.0       0.0

** Compaction Stats [default] **
Priority    Files   Size     Score Read(GB)  Rn(GB) Rnp1(GB) Write(GB) Wnew(GB)␣
↪Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) CompMergeCPU(sec) Comp(cnt) Avg(sec) KeyIn␣
↪KeyDrop Rblob(GB) Wblob(GB)
-----------------------------------------------------------------------------------------
↪----------------------------------------------------------------------------------------
↪-------------------

Blob file count: 0, total size: 0.0 GB, garbage size: 0.0 GB, space amp: 0.0

Uptime(se
```

11.52 /admin/fetchErrorAlerts

Fetch error alerts shown in the UI under Logs / Config Errors.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "ErrorAlertsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "errors": []
    }
}
```

11.53 /admin/fetchErrorMemoryLog

Fetch the error log that is stored in memory.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
raw	Return data as an attachment	Boolean (0 or 1)
limit	Max rows to fetch	Number

Response Example:

```
{
    "LogsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "logs": [
            {
                "log": "2023-12-30 16:43:20,296862(139643264648960): ERROR: \/home\/
↪vagrant\/git\/AirMOS\/src\/pcos_cluster_replication.c: pcosclustrep_local_object_fetch_
↪last_version_cb(2666):  : cluster replicate start : status(ERROR - An error occurred)\n
↪"
            },
            {
                "log": "2023-12-30 16:43:20,296837(139643264648960): ERROR: \/home\/
↪vagrant\/git\/AirMOS\/src\/pcos_cluster_replication.c: pcosclustrep_local_group_fetch_
↪last_version_cb(2684):  : cluster replicate start : status(ERROR - An error occurred)\n
```

(continues on next page)

(continued from previous page)

```
↪"
                }
            ]
        }
}
```

11.54 /admin/fetchJobReports

Fetch the reports status that is generated by migration jobs.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`startId`	Start id to fetch reports from	Number
`limit`	Max rows to fetch	Number

Response Example:

```
{
    "JobReportsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "reports": []
    }
}
```

11.55 /admin/fetchJobTasks

Fetch tasks that are running as part of migration jobs. A job consists of multiple tasks.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml

Response Example:

```
{
    "JobTasksResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "jobs": []
    }
}
```

11.56 /admin/fetchServers

Fetch server information.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "ServersResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "isNode": true,
        "servers": [
            {
                "serverId": 17,
                "hostname": "localhost:8010",
                "internalHostname": "localhost",
                "ipcPort": 8082,
                "sslPort": 0,
                "rack": "",
                "site": "",
                "status": "RW",
                "pingUp": true,
                "ipcUp": true,
                "lastCrash": 0,
                "startTime": 1703954594,
                "version": "Development - De",
                "cpuInfo": "Intel(R) Core(TM) i9-8950HK CPU @ 2.90GHz (3 logical, 1␣
↪physical)",
                "ramTotal": 16014,
                "ramUsed": 963,
                "ramUsedServer": 71,
                "cpuSystem": 0.07,
                "cpuUser": 0.04,
                "cpuSystemServer": 0.33,
                "cpuUserServer": 0.39,
                "numThreads": 75,
                "numFds": 114,
                "numSockets": 4,
                "bytesInPerSec": 235,
                "bytesOutPerSec": 235,
                "devices": [
                    {
                        "deviceId": 17,
                        "mount": "\/home\/vagrant\/drive",
                        "type": 0,
                        "status": "RW",
```

(continues on next page)

(continued from previous page)

```
                        "sizeUsed": 78028,
                        "sizeTotal": 159538,
                        "ioReadSec": 0.00,
                        "ioWriteSec": 0.00,
                        "ioErrors": 0
                    }
                ]
            }
        ]
    }
}
```

11.57 /admin/fetchShardMap

Fetch shard map information.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard Type	String value: file_group or meta

Response Example:

```
{
    "ShardMapResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "maps": [
            {
                "dataType": "METADATA",
                "shards": [
                    {
                        "id": 1,
                        "hashes": "-
↪ffffffffffffffff04ffff00ff00ff00ff00ff00ff00ff00ff00ff00ff00ff00ff00ff000c00",
                        "numHashes": 8192,
                        "hosts": [
                            {
                                "conn": "localhost:8082",
                                "master": true,
                                "lastTransactionId": {
                                    "id": 8,
                                    "termId": 3,
                                    "timestamp": 1703537854663
                                },
                                "repState": "NONE",
                                "site": "",
```

(continues on next page)

(continued from previous page)

```
                                "rack": ""
                            }
                        ]
                    }
                ],
                "migrations": []
            }
        ]
    }
}
```

11.58 /admin/fetchSkulkReports

Fetch skulk reports.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
lastDate	Used for pagination to fetch after last entry	Number
lastServer	Used for pagination to fetch after last entry	Number
lastDevice	Used for pagination to fetch after last entry	Number
limit	Maximum rows to return	Number

Response Example:

```
{
    "SkulkReportsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "reports": []
    }
}
```

11.59 /admin/flushCache

Flushes caches of all servers in the cluster. The caches contain cached database objects.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.60 /admin/getAlertPolicy

Fetch an alert policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyId`	Id of policy	Number

Response Example:

```
{
    "AlertPoliciesResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "policies": [
            {
                "id": 17,
                "type": "DRIVE_ERRORS",
                "value": "3",
                "valueTime": 30,
                "destination": "SNMP"
            }
        ]
    }
}
```

11.61 /admin/getHttpConns

Get information about current open http connections on the server.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml

Response Example:

```
{
    "HttpResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "conns": [
            {
                "address": "10.0.0.18",
                "url": "GET \/admin\/getHttpConns?type=meta&f=json",
                "startTime": 1703976312,
                "inProgress": true,
                "bytesInPerSec": 637,
                "bytesOutPerSec": 0,
                "bytesInProgress": 637,
                "bytesOutProgress": 0,
                "bytesInTotal": 0,
                "bytesOutTotal": 0
            },
            {
                "address": "10.0.0.18",
                "url": " ",
                "startTime": 1703976312,
                "inProgress": false,
                "bytesInPerSec": 0,
                "bytesOutPerSec": 0,
                "bytesInProgress": 0,
                "bytesOutProgress": 0,
                "bytesInTotal": 0,
                "bytesOutTotal": 0
            }
        ],
        "client_conns": []
    }
}
```

11.62 /admin/getIntegrityPolicy

Fetch a file integrity policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyId`	Id of policy	Number

Response Example:

```
{
    "PolicyResponse": {
        "statusCode": 200,
```

(continues on next page)

(continued from previous page)

```
        "statusString": "OK",
        "statusText": "Success",
        "id": 17,
        "name": "test",
        "rules": [
            {
                "id": 1,
                "method": "GET",
                "factor": "size_over",
                "value": "200"
            }
        ]
    }
}
```

11.63 /admin/getIpcConns

Get information about current open inter-process http connections on the server.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml

Response Example:

```
{
    "IpcResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "connsIn": [
            {
                "connectTime": 1703954594,
                "address": "",
                "host": "localhost:8010",
                "bytesInPerSec": 186,
                "bytesOutPerSec": 201,
                "bytesInTotal": 3374138,
                "bytesOutTotal": 24804451,
                "loopback": true,
                "dataConn": false,
                "transfers": []
            }
        ],
        "connsOut": [
            {
                "connected": true,
                "connectTime": 1703954594,
```

(continues on next page)

(continued from previous page)

```
                "host": "localhost:8010",
                "bytesInPerSec": 201,
                "bytesOutPerSec": 186,
                "bytesInTotal": 24804451,
                "bytesOutTotal": 3374138,
                "loopback": true,
                "dataConn": false,
                "transfers": []
            }
        ]
    }
}
```

11.64 /admin/getStats

Returns stats shown in Timings and Counters diagnostics portions of UI.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`delta`	1 if only counts from last call to API with reset=1 should be returned	Boolean (1 or 0)
`reset`	1 if delta counts should be reset	Boolean (1 or 0)

Response Example:

```
{
    "StatsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "counters": [
            {
                "label": "http_client_200",
                "description": "HTTP client request with 200 status",
                "startTime": 1703537527,
                "endTime": 1703539360,
                "count": 0,
                "isAggregate": true,
                "parentLabel": ""
            }
        ],
        "events": [
            {
                "label": "system_thread_wait",
                "description": "system thread wait time",
                "startTime": 1703537527,
                "endTime": 1703539360,
                "min": 0,
```

(continues on next page)

(continued from previous page)

```
                "max": 0,
                "mean": 0.000000,
                "stddev": 0.000000,
                "numEvents": 31,
                "numErrors": 0,
                "isAggregate": false,
                "parentLabel": "THR"
            }
        ]
    }
}
```

11.65 /admin/getStatsOverview

Returns stats for memory and requests displayed in UI stats overview.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "StatsResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "startTime": 1703537521,
        "idCacheSize": 0,
        "memory": [
            [
                1703537520000,
                0.00
            ],
            [
                1703537580000,
                0.00
            ]
        ],
        "requests": [
            [
                1703537520000,
                2
            ],
            [
                1703537580000,
                23
            ]
```

(continues on next page)

(continued from previous page)

```
        ]
    }
}
```

11.66 /admin/getThreadPools

Returns stats for thread pools.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "ThreadResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "pools": [
            {
                "name": "config",
                "numThreads": 1,
                "numBusy": 0,
                "queueSize": 0,
                "queueSizeHi": 7
            }
        ]
    }
}
```

11.67 /admin/listAlertPolicies

Fetch all of the alert policies.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AlertPoliciesResponse": {
        "statusCode": 200,
```

(continues on next page)

(continued from previous page)

```
        "statusString": "OK",
        "statusText": "Success",
        "policies": [
            {
                "id": 17,
                "type": "DRIVE_ERRORS",
                "value": "3",
                "valueTime": 30,
                "destination": "SNMP"
            }
        ]
    }
}
```

11.68 /admin/listIntegrityPolicies

Fetch all of the file integrity policies.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "PoliciesResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "policies": [
            {
                "id": 17,
                "name": "test",
                "isGlobal": false,
                "rules": [
                    {
                        "id": 1,
                        "method": "GET",
                        "factor": "size_over",
                        "value": "200"
                    }
                ]
            }
        ]
    }
}
```

11.69 /admin/login

Login using local authentication. A token is returned that can then be used in the Authorization header as a bearer token for other API requests.

Tokens are stored internally in a distributed hash table, and will be lost if the system is bounced.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml
loginId	username	String
password	password	String
application	Application name (tenant)	String
source	Optional parameter that indicates if request is for UI or API	String value: ui, api

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "superUser": "SUPERUSER",
        "adminNode": true,
        "userName": "admin",
        "token": "dAFpbnk..."
    }
}
```

11.70 /admin/logout

Logout using the token passed in cookie or authorization header.

Parameters:

Parameter	Description	Type
f	Response format	String value: json or xml

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.71 /admin/migrateShardMap

Make a request to migrate data between database shards. Migration can either be done by specifying the number of hashes to migrate (`numHashes`) or the list of hashes (`hashes`).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard type	String value: file_group or meta
`srcShard`	ID of source shard	Number
`destShard`	ID of destination shard	Number
`numHashes`	Number of hashes to migrates	Number
`hashes`	Comma separated list of hashes to migrate	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.72 /admin/oidcAssumeRole

Used to get AirMettle S3 credentials given a JWT from an Identity Provider that an application (tenant) is configured to use as an OpenID Connect provider.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`applic`	Application name	String
`token`	JWT token from authentication provider. This can also be passed in the Authorization header as a Bearer token. The JWT token should be obtained from the authentication provider using their SDK or an openauth SDK.	String
`ttl_pa`	how long s3 credentials should be good for. This defaults to 86400 seconds	Number

Response Example:

```
{
    "OIDCAssumeRoleResponse": {
```

(continues on next page)

(continued from previous page)

```
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "s3KeyId": "test-AdcIiEaMEtEN3TB-1",
        "s3Secret": "ZUg6...",
        "ttl": 86400
    }
}
```

11.73 /admin/oidcCheckApplication

Used to check if an application (tenant) is configured to use OpenID Connect for login.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`application`	Application name	String

Response Example:

```
{
    "OIDCCheckApplicationResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "isOIDC": true
    }
}
```

11.74 /admin/oidcComplete

Used by UI to complete OpenID Connect login flow after receiving redirect callback from the Identity Provider. A token is returned that can then be used in the Authorization header as a bearer token for other API requests.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`application`	Application name	String
`redirectUri`	Redirect URI passed to oidcStart	String
`codeVerifier`	Code verifier for OIDC flow	String
`token`	Token received in OIDC flow	String

Response Example:

```
{
    "OIDCCompleteResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "superUser": "SUPERUSER",
        "adminNode": true,
        "userName": "admin",
        "token": "dAFpbnk..."
    }
}
```

11.75 /admin/oidcStart

Used by UI to start OpenID Connect flow. A redirect uri is returned that is used to redirect the browser to the Identity Provider.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`application`	Application name	String
`redirectUri`	URI the Identity Provider should redirect to when complete	String
`s256CodeChallenge`	PKCE code challenge to use	String
`state`	State to pass back in redirect	String

Response Example:

```
{
    "OIDCStartResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success",
        "redirectUri": "https://..."
    }
}
```

11.76 /admin/redistributeShardMap

Make a request to migrate data between database shards and have the hashes distributed evenly.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard type	String value: file_group or meta

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.77 /admin/resubmitJobTask

Rerun a task in a migration job that failed.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`id`	ID of job task	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.78 /admin/resubmitJobTasks

Rerun all failed tasks in a migration job.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`name`	Job name	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.79 /admin/server/version

Returns the version string of the server.

Response Example:

```
Developement
```

11.80 /admin/updateAccount

Update an account in an application (tenant).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`id`	ID of accout	Number
`apiKey`	S3 API Key for Account	String
`s3Secret`	S3 Secret for Account	String
`password`	Password for Account	String
`superUser`	Super User Type	String value: NONE, SUPERUSER, or SUPERUSER_READ_ONLY
`accountRole`	Role in application (tenant)	String value: NONE, READ_ONLY, READ_WRITE, ADMIN

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.81 /admin/updateAlertPolicy

Updates an alert policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policy`	ID of policy	Number
`type`	Alert Type	String value: DRIVE_ERRORS, SKULK_ERRORS, SERVERS_DOWN, SERVERS_UP, WRITE_OBJECTS, READ_OBJECTS, LOGIN_FAILURES, REPLICATION_ERRORS, MD5_ERRORS, BYTES_IN_HIGH, BYTES_OUT_HIGH, SWIFT_S3_AUTH_ERRORS
`alertV`	Threshold Value	Number
`alertV`	Threshold Time Period (seconds)	Number
`destin`	Alert Destination	String value: EMAIL, SNMP

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.82 /admin/updateApplication

Updates an application (tenant).

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`id`	ID of application (tenant)	Number
`s3HostnamePostfix`	S3 Hostname	String
`maxUploadSize`	Max Upload (MB)	Number
`fileSplitSize`	Split Size (MB)	Number
`expireTime`	Expire Time (sec)	Number
`integrityPolicyId`	ID of File Integrity Policy	Number
`totalCopies`	Total Copies	Number
`numSites`	Num Sites	Number
`forceCdn`	Only allow request to come from CDN ACL	Boolean (1 or 0)
`useCache`	Use Cache	Boolean (1 or 0)
`erasureCodingType`	Erasure Coding Type	String value: NONE, CAUCHY_GOOD
`erasureCodingK`	Erasure Coding K Value	Number

continues on next page

Table 2 – continued from previous page

Parameter	Description	Type
`erasureCodingN`	Erasure Coding N Value	Number
`compressionType`	Compression Type	String value: NONE, SNAPPY, GZIP
`encryptionOn`	Encryption is On	Boolean (1 or 0)
`extractArchive`	Extract files from archive file uploads	Boolean (1 or 0)
`versionObjects`	Version Objects	Boolean (1 or 0)
`maxObjectVersions`	Maximum number of versions to store	Number
`encryptionNewKeys`	Generate new encryption keys	Boolean (1 or 0)
`expireFiles`	Expire Files	Boolean (1 or 0)
`masterOnNotFound`	Use Master DB if entry not found for file	Boolean (1 or 0)
`oidcClientId`	OIDC (OpenID Connect) Client ID	String
`oidcSecret`	OIDC Client Secret	String
`oidcDiscoveryUrl`	OIDC Discovery URL	String
`oidcScopes`	OIDC Scopes (comma separated)	String
`oidcAutoCreateAccounts`	OIDC Auto Create Accounts on Login	Boolean (1 or 0)
`oidcUsePkce`	OIDC Use PKCE	Boolean (1 or 0)
`oidcUsernameClaim`	OIDC Username Claim in JWT	String
`oidcAllowedDomains`	OIDC Allowed Domains (comma separated)	String
`authProvider`	Authentication Provider Type	String value: LOCAL, OIDC
`url`	URL for information purposes	String
`email`	Email contact	String
`comment`	Comment for information purposes	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.83 `/admin/updateCluster`

Updates the name of a remote cluster.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`id`	ID of cluster	Number
`name`	Name of cluster	String

Response Example:

```
{
    "AdminResponse": {
```

(continues on next page)

(continued from previous page)

```
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.84 /admin/updateIntegrityPolicy

Updates a file integrity policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`policyId`	ID of policy	Number
`policyName`	Name for policy	String
`isGlobal`	Indicates if policy is default	Boolean (1 or 0)

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.85 /admin/updateIntegrityPolicyRule

Update a rule in a file integrity policy.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`factor`	Indicates when an integrity check is done. It can be based on file size or content type.	String value: size over, size under, type contains, type not contains, always
`method`	Which operation the rule is applied to for when that operation should do an integrity check. • GET: on file get requests • skulker: when files are checked in background • migrate file: when files are migrated • all: all of the above	String value: GET, skulker, all, migrate file
`value`	Value for factor	String: bytes for factor = size over or sizer under. comma separated list of content types for type “not contains” or type “contains”
`ruleId`	ID of rule	Number
`policyId`	ID of policy	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.86 `/admin/updateServer`

Add a host to the cluster.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`id`	ID of server	Number
`hostname`	FQDN:http_port of server to add	String
`rack`	Optional rack name server is one	String
`site`	Optional site name server is at	String

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

11.87 /admin/updateShardHost

Updates a node for a shard for storing data for the shard.

Parameters:

Parameter	Description	Type
`f`	Response format	String value: json or xml
`type`	Shard type	String value: file_group or meta
`shardId`	ID of shard to add host to	String
`site`	Optional site of host	String
`rack`	Optional rack of host	String
`host`	Host FQDN	String
`port`	Host IPC Port	Number

Response Example:

```
{
    "AdminResponse": {
        "statusCode": 200,
        "statusString": "OK",
        "statusText": "Success"
    }
}
```

PYTHON MODULE INDEX

INDEX