horizontal line

### 

# LiveMigrator API examples

[**LiveMigrator API examples**](#_b44zi72ik0hs) **1**

[Launching a Migration](#_zepw6080nrib) 1

[Scripting multiple migrations launch](#_1ggb0mtuctss) 2

[# bulk\_launch\_lm.sh](#_tlrmdwlouyyc) 2

[Using an Edge Point - example](#_udket6oa71mv) 3

[Pause/Abort/Delete/Resume migrations](#_1d9varuepjrw) 3

[Additional Scripts](#_kk9cm4he7ao) 4

[# get\_all\_migrations.py](#_jhpk20ak5lyg) 4

[# get\_running\_migrations.py](#_r9kw8bqe66yt) 5

[# get\_fs\_path\_and\_ids.py](#_us8pyxssqewa) 6

[# get\_migration.sh](#_ldw4oz117atz) 7

[# abort\_finishing.py](#_oi218i1vjxb9) 7

[# statecount.sh](#_l2uwozz009f0) 8

[Bulk Create Rules](#_ng0jz59e6ypn) 9

[# login\_fusion.sh](#_ltt5utclbuno) 9

[# bulk\_create.sh](#_v751ro6f4b4p) 9

[Delete a Rule](#_qjbw9v7rtmtc) 10

[Make Consistent](#_w6q7rgwsta5c) 10

[# makeConsistent <path id - without /WD-VIRTUAL> <source of truth Zone - ALWAYS Gen1>](#_m16tmqsjc1l1) 10

[# mcStatus.sh <path id - without /WD-VIRTUAL>](#_fhjvfndjktlr) 10

[Other Tools](#_69kko9e8xu3m) 11

[Monitoring transfer rates](#_6xl8uumvhwkc) 11

[Task Monitoring](#_t2iagnic7hcr) 11

## Launching a Migration

Basic concepts

* + To use Start in migrationEdge, parent rule has to exist.
  + You cannot create a new rule to a subdirectory if a rule to a parent exists.
  + If using a migration edge, directories are processed in alphabetical order.
  + API call

curl --insecure -X PUT "[https://localhost:8084/plugins/migrator/migrate?path=<virtualPath>&donor=<donosrZone>&beneficiary=<beneficiaryZone>&startPath=&fsType=ADL&overwritePolicy=<SKIP\_IF\_SIZE\_MATCH or CLOBBER>&iterationLimit=10000&pollTimer=1&dateAfterWhichToMigrate=0](https://localhost:8084/plugins/migrator/migrate?path=/WD-VIRTUAL-$i&donor=$DONOR&beneficiary=$TARGET&startPath=&fsType=ADL&overwritePolicy=SKIP_IF_SIZE_MATCH&iterationLimit=10000&pollTimer=1&dateAfterWhichToMigrate=0)”

* + When launching or restarting a migration options are:
    - path= Virtual path of the rule
    - donor= Source Zone (Gen1)
    - beneficiary= Destination Zone (Gen2)
    - startPath= Where replication will start based on the last file or directory it saw (migration edge or no value for top of the rule) - Starts from the end of the rule path and never starts with a "/".
      * For example Rule = /data/raw/Marketing/Campaign/FormationDailyMetrics/ver=01/
      * Desired Start directory (Edge): /data/raw/Marketing/Campaign/FormationDailyMetrics/ver=01/**LoadDate=2020-02-23/**
      * startPath=LoadDate=2020-02-23/
    - fsType=ADL File system (ADL for Azure ADLS)
    - overwritePolicy= SKIP\_IF\_SIZE\_MATCH or CLOBBER (overwrite)
    - iterationLimit=10000 Windows size, number of simultaneous paths to pull for processing
    - pollTimer=1 time in between windows for processing (usually between 1-10)
    - dateAfterWhichToMigrate=0 - if doing CLOBBER, you can specify that files are only overwritten if their timestamp is after a specified time. This time is set in UTC.
  + For an aborted rule you can find the migration Edge (<migrationEdge>) via:
    - # curl --insecure https://localhost:8084/plugins/migrator/getAll | xmllint -format - > .migrations\_list.xml
    - # curl --insecure https://localhost:8084/plugins/migrator/<migrationId> | xmllint -format
  + You need to ABORT the migration to restart it using the migration edge:
    - # curl --insecure -X PUT https://localhost:8084/plugins/migrator/<migrationId>/abort

## Scripting multiple migrations launch

Here is how to launch multiple migrations using a script:

#### # bulk\_launch\_lm.sh

#!/bin/bash

# THIS SCRIPT STARTS FROM THE ROOT OF THE RULE - use it to launch multiple rules using WD-VIRTUAL Ids

DONOR=zone-central

BEN=zoneG2-central

# CLOBBER or SKIP\_IF\_SIZE\_MATCH

METHOD=CLOBBER

for i in 0ca624a1-60c5-11ea-9187-000d3aa50bab 0a380a71-60c5-11ea-9187-000d3aa50bab 0bd3923b-60c5-11ea-9187-000d3aa50bab 0b1c01ec-60c5-11ea-9187-000d3aa50bab 5638383a-60c5-11ea-9187-000d3aa50bab 5868850c-60c5-11ea-9187-000d3aa50bab 57ade781-60c5-11ea-9187-000d3aa50bab

do

echo "Doing rule $i"

curl --insecure -X PUT "https://localhost:8084/plugins/migrator/migrate?path=/WD-VIRTUAL-$i&donor=$DONOR&beneficiary=$BEN&startPath=&fsType=ADL&overwritePolicy=$METHOD&iterationLimit=10000&pollTimer=1&dateAfterWhichToMigrate=0"

done

printf '\n======\n[EOL]\n'

### Using an Edge Point - example

How to launch a migration from an edge point, for example:

* /data/raw/Marketing/Campaign/FormationDailyMetrics/ver=01/LoadDate=2020-02-23/
* Important to notice, for the startPath, DO NOT INCLUDE A “/“ at the beginning. If you do it will fail.
* Change donor and beneficiary to the appropriate values.

# curl --insecure -X PUT "https://localhost:8084/plugins/migrator/migrate?path=/WD-VIRTUAL-038cd2bf-51c1-11ea-8633-000d3aa50bab&donor=zoneBgen1&beneficiary=zoneBgen2&startPath=LoadDate=2020-02-23/&fsType=ADL&overwritePolicy=SKIP\_IF\_SIZE\_MATCH&iterationLimit=10000&pollTimer=1&dateAfterWhichToMigrate=0”

## Pause/Abort/Delete/Resume migrations

Pause

# curl --insecure -X PUT<https://localhost:8084/plugins/migrator/><migrationid>/pause

Resume (**you cannot resume an aborted rule**)

# curl --insecure -X PUT<https://localhost:8084/plugins/migrator/><migrationid>/resume

Abort

# curl --insecure -X PUT<https://localhost:8084/plugins/migrator/><migrationid>/abort

Delete (**requires it to be finished or aborted**)

# curl --insecure -X DELETE<https://localhost:8084/plugins/migrator/><migrationid>/delete

## Additional Scripts

#### # get\_all\_migrations.py

#!/usr/bin/python

import xml.etree.ElementTree as ET

import urllib2

import ssl

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

migrations = urllib2.urlopen("https://localhost:8084/plugins/migrator/getAll", context=ctx).read()

fs = urllib2.urlopen("https://localhost:8084/fusion/fs", context=ctx).read()

migroot = ET.fromstring(migrations)

fsroot = ET.fromstring(fs)

sum = 0

for tt in migroot.findall(".//migration"):

migrationPath = tt.find("./migrationPath")

migrationId = tt.find("./migrationId")

migrationState = tt.find("./state")

migrationStart = tt.find("./migrationStartTime")

for rd in fsroot.findall(".//replicatedDirectory"):

path = rd.find("./path")

if path.text == migrationPath.text:

locations = rd.findall("./mappings//mapping[zoneId='zoneXgen1']/location")

writerNode = rd.find("./writer/nodeId")

print locations[0].text, migrationState.text, migrationId.text, migrationStart.text, migrationPath.text, writerNode.text

#### # get\_running\_migrations.py

#!/usr/bin/python

import xml.etree.ElementTree as ET

import urllib2

import ssl

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

migrations = urllib2.urlopen("https://localhost:8084/plugins/migrator/getAll", context=ctx).read()

fs = urllib2.urlopen("https://localhost:8084/fusion/fs", context=ctx).read()

migroot = ET.fromstring(migrations)

fsroot = ET.fromstring(fs)

sum = 0

for tt in migroot.findall(".//\*[state='RUNNING']"):

migrationPath = tt.find("./migrationPath")

migrationId = tt.find("./migrationId")

duration = tt.find("./progress/duration")

filesSeen = tt.find("./progress/filesSeen")

filesToMigrate = tt.find("./progress/filesToMigrate")

sizeOfMigration = tt.find("./progress/sizeOfMigration")

bytesToMigrate = tt.find("./progress/bytesToMigrate")

for rd in fsroot.findall(".//replicatedDirectory"):

path = rd.find("./path")

if path.text == migrationPath.text:

locations = rd.findall("./mappings//mapping[1]/location")

print ""

print migrationId.text, migrationPath.text, locations[0].text

print " - duration:", duration.text, " filesSeen:", filesSeen.text, " filesToMigrate:", filesToMigrate.text

print " - sizeOfMigration:", sizeOfMigration.text, " bytesToMigrate:", bytesToMigrate.text

writerNode = rd.find("./writer/nodeId")

print " - WriterNodeID:", writerNode.text

#### # get\_fs\_path\_and\_ids.py

#!/usr/bin/python

import xml.etree.ElementTree as ET

import urllib2

import ssl

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

fs = urllib2.urlopen("https://localhost:8084/fusion/fs", context=ctx).read()

node = urllib2.urlopen("https://localhost:8084/fusion/nodes", context=ctx).read()

root = ET.fromstring(fs)

noderoot = ET.fromstring(node)

sum = 0

for rd in root.findall(".//replicatedDirectory"):

path = rd.find("./path")

writerNode = rd.find("./writer/nodeId")

for nd in noderoot.findall(".//node"):

nodeId = nd.find("./nodeIdentity")

nodeName = nd.findall("./attributes//attribute[key='node.name']/value")

if writerNode.text == nodeId.text:

for location in rd.findall("./mappings//mapping[zoneId='zoneXgen1']/location"):

print location.text , path.text, writerNode.text, nodeName[0].text

#### # get\_migration.sh

#!/bin/bash

curl --insecure https://localhost:8084/plugins/migrator/$1 | xmllint --format -

#### # abort\_finishing.py

#!/usr/bin/python

import xml.etree.ElementTree as ET

import urllib2

import ssl

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

class MethodRequest(urllib2.Request):

def \_\_init\_\_(self, \*args, \*\*kwargs):

if 'method' in kwargs:

self.\_method = kwargs['method']

del kwargs['method']

else:

self.\_method = None

return urllib2.Request.\_\_init\_\_(self, \*args, \*\*kwargs)

def get\_method(self, \*args, \*\*kwargs):

if self.\_method is not None:

return self.\_method

return urllib2.Request.get\_method(self, \*args, \*\*kwargs)

migrations = urllib2.urlopen("https://localhost:8084/plugins/migrator/getAll", context=ctx).read()

root = ET.fromstring(migrations)

for tt in root.findall(".//\*[state='FINISHING']/migrationId"):

migration\_id = tt.text

print("Aborting " + migration\_id)

req = MethodRequest("https://localhost:8084/plugins/migrator/" + migration\_id + "/abort", method='PUT')

urllib2.urlopen(req, context=ctx).read()

#### # statecount.sh

#!/usr/bin/env bash

cp ./count\_new.xml ./count\_old.xml

curl --insecure https://localhost:8084/plugins/migrator/getAll | xmllint -format - > ./count\_new.xml

if [ -z $1 ]

then

printf '\nNo dated output file provided ==== \n'

else

cp count\_new.xml $1

fi

printf '\n ===== Previous =====\n'

printf 'SCHEDULED = ' ; grep SCHEDULED ./count\_old.xml | wc -l

printf 'WARMING UP = ' ; grep WARMING ./count\_old.xml | wc -l

printf 'RUNNING = ' ; grep RUNNING ./count\_old.xml | wc -l

printf 'FINISHING = ' ; grep FINISHING ./count\_old.xml | wc -l

printf 'FINISHED = ' ; grep FINISHED ./count\_old.xml | wc -l

printf 'ABORTED = ' ; grep ABORTED count\_old.xml | wc -l

printf 'PAUSED = ' ; grep PAUSED ./count\_old.xml | wc -l

printf '\n ===== Current =====\n'

printf 'SCHEDULED = ' ; grep SCHEDULED ./count\_new.xml | wc -l

printf 'WARMING UP = ' ; grep WARMING ./count\_new.xml | wc -l

printf 'RUNNING = ' ; grep RUNNING ./count\_new.xml | wc -l

printf 'FINISHING = ' ; grep FINISHING ./count\_new.xml | wc -l

printf 'FINISHED = ' ; grep FINISHED ./count\_new.xml | wc -l

printf 'ABORTED = ' ; grep ABORTED count\_new.xml | wc -l

printf 'PAUSED = ' ; grep PAUSED ./count\_new.xml | wc -l

printf '\n====\n[EOF]\n'

### Bulk Create Rules

This is a two-step process. First you need to get a login token from the Fusion server and then create the rules.

#### # login\_fusion.sh

#!/bin/sh

# How to use it --> # login\_fusion

curl -v \

--header "Accept: application/json, text/plain, \*/\*" \

--header "Content-Type: application/json;charset=UTF-8" \

--data '{"username":"admin","password":"**<Insert password here>**"}' \

--insecure \

https://localhost:8443/api/authenticate/login

printf '\n=====[EOL]\n

#### # bulk\_create.sh

#!/bin/bash

# Insert cookie from Login

SRCZONE="zone-central"

TGTZONE="zoneG2-central"

COOKIE="Cookie: JSESSIONID=**1hji1fvb85zx36gbn8hwoh041**"

TOKEN="authn-token: **ec907607-2e64-44e4-94ed-a3679be4e832**"

**# Single rule example - for multiple rules use for loops, for Source/Destination paths remapping change RULEPATH variables.**

RULENAME="W3c - /data/published/Customer/Customer360/Customer360ActivityRestricted/ver=01/"

RULEPATH="/data/published/Customer/Customer360/Customer360ActivityRestricted/ver=01/"

echo "Creating rule called: $RULENAME"

echo curl -v \

--header "Accept: application/json, text/plain, \*/\*" \

--header "Content-Type: application/json;charset=UTF-8" \

--header "$COOKIE" \

--header "$TOKEN" \

--data "{\"replicatedFolder\":{\"zones\":[{\"local\":false,\"id\":\"$TGTZONE\",\"name\":\"$TGTZONE\"},{\"local\":true,\"id\":\"$SRCZONE\",\"name\":\"$SRCZONE\"}],\"priorityZone\":\"$SRCZONE\",\"pathMap\":{\"$TGTZONE\":\"$RULEPATH\",\"$SRCZONE\":\"$RULEPATH\"},\"name\":\"$RULENAME\",\"excluded\":[\"/\*\*/.Trash/\*\*\",\"/\*\*/.hive-staging\*\*\",\"/\*\*/.snapshot/\",\"/\*\*/.snapshot\",\"/\*\*/.Trash\"],\"scheduleChecks\":false}}" \

--insecure \

<https://localhost:8443/api/replicatedFolders>

### 

### Delete a Rule

Use the actual path for the rule:

# curl -v --insecure -X DELETE https://localhost:8084/fusion/fs?path=*/sbx001/poctest1*

### Make Consistent

#### # makeConsistent <path id - without /WD-VIRTUAL> <source of truth Zone **- ALWAYS Gen1**>

#!/bin/bash

# Starts a Make Consistent process - execute from the rule's Gen2 writer

WHERE=$1

ZONE=$2

curl --insecure -v -X PUT "https://localhost:8084/fusion/fs/repair?path=/WD-VIRTUAL-$WHERE&src=$ZONE&replace=true&preserve=false&type=cc&nonBlocking=true"

printf '\n=====\n[EOL]\n'

#### # mcStatus.sh <path id - without /WD-VIRTUAL>

#!/bin/bash

# Check the status of a Make Consistent task - execute on the rule's Gen2 writer

WHERE=$1

curl --insecure -v "https://localhost:8084/fusion/fs/repairs?path=/WD-VIRTUAL-$WHERE" | xmllint --format -

printf '\n=====\n[EOL]\n'

## Other Tools

### Monitoring transfer rates

The easiest way to estimate transfer rates by number of files is to do the following from the Donor/Beneficiary zones:

$ tail -F /var/log/fusion/server/fusion-migration-audit.log | grep Window

It returns the start/completion of processing a Window of files as set on the migration launch (i.e. 10,000 files). Depending if on Donor or Beneficiary it is getting or uploading the files.

For example, taking a reading from the Beneficiary Zone:

2020-03-19T15:**55:40**,532 INFO - d96bbfd9-6a1b-11ea-86f2-000d3a90dcbf: Iteration Window 1584658445764 Complete.

2020-03-19T15:**56:23**,918 INFO - d96bbfd9-6a1b-11ea-86f2-000d3a90dcbf: Iteration Window 1584658469948 Complete.

2020-03-19T15:**56:49**,006 INFO - d96bbfd9-6a1b-11ea-86f2-000d3a90dcbf: Iteration Window 1584658513359 Complete.

The time difference between each window is how much it takes to process 10,000 files into the storage. This number is affected by file size, directory structure, network conditions, etc. Note that while one side is working, the other is on hold waiting for the other to complete its Window.

### Task Monitoring

The output of all API calls include the task ID under the Content-Location line, fagor example:

Content-Location: <https://localhost:8084/fusion/task/bc64c47a-7070-11ea-afc9-000d3a41edf8>

You can display the status of the task by using:

# curl --insecure -v <https://localhost:8084/fusion/task/bc64c47a-7070-11ea-afc9-000d3a41edf8> | xmllint --format -

This will bring all tasks been executed in the server:

# curl --insecure -v [https://localhost:8084/fusion/task](https://localhost:8084/fusion/task/bc64c47a-7070-11ea-afc9-000d3a41edf8)s