stonebranch

Universal Task Documentation

Universal Automation Center support for HDFS

ut-hdfs-linux

Associated Activities:

Date: 28 November 2019

Author: Nils Buer

Revision: 00

CONFIDENTIALITY INFORMATION

Distribution list: Stonebranch Customer

Revision	Date	Author	Changes
00	20191129	Nils Buer	Initial Document (WIP)

Abstract:

The here described Universal Tasks allow to Transfer and retrieve files from the Hadoop file System (HDFS) to a local file system. As a result, you can integrate any HDFS file transfer into you existing or new scheduling workflows.

Contents

1 Disclaimer

No support and no warranty are provided by Stonebranch GmbH for this document and the related Universal Task. The use of this document and the related Universal Task is on your own risk.

Before using this task in a production system, please perform extensive testing.

Stonebranch GmbH assumes no liability for damage caused by the performance of the Universal Tasks

2 Scope

This document provides a documentation how to install and use the Universal Tasks for HDFS Storage File Transfers. If more Task will be created in the future this document will be updated accordingly.

3 Introduction

Storing data in a Hadoop file system becomes an integral part of most modern bigdata IT landscapes. With Universal Automation Center you can securely automate your AWS, Azure, Google or HDFS File Transfers and integrate them into your existing scheduling flows.

The here described Series of Universal Tasks focuses on HDFS file transfer. A similar solution as HDFS is also available for Amazon S3, Google GCS or Azure Blob storage.

Some details about the universal tasks for HDFS:

- The Universal Task uses the Python hdfs module, which calls the Hadoop WebHDFS REST API
- The Universal Task supports both Universal Agent for Linux/Unix and Windows however it has been currently only tested against a Linux Agent
- You can select different log-levels e.g. Info and debug
- All Passwords are encrypted using Controller Credentials
- Currently only the InsecureClient (the default) is implemented the TokenClient can be implemented on request

The following Universal Tasks for Google Blob Storage have been implemented:

Command	UT Name	Description
delete file	ut-hdfs-delete-file	Delete file on HDFS
download file	ut-hdfs-download-file	Copy file from HDFS to local
upload file	ut-hdfs-upload-file	Copy file to HDFS
file monitor	ut-hdfs-file-monitor	Monitor file on HDFS
list directory	ut-hdfs-list-directory	List directory on HDFS

4 Installation

4.1 Software Requirements for Linux Agent

Universal Task name: *ut-hdfs-linux*

Related UAC XML Files for template and task: *Github repository*

Requirements:

- Python 3.6
- For Python the following modules are required:
  - *os, for linux command execution*
  - *datetime and time to compare file timestamps*
  - *hdfs to perform the hdfs file operations*
  - *logging, for python loglevel support*

  Note: Only the module hdfs needs to be added via python installer

  ⇨ *pip install hdfs*

- Universal Controller V6.4.7.0 or higher
- Universal Agent V6.5.0.0 or higher installed on a Linux Server

4.2 Installation Steps

The following describes the installation steps:

1. **Check the current Python Version**

   python -V (Note: Captial "V")

   If your Version is Python 3.6 or later all is fine. If a no python or a lower Version has been installed upgrade your python Version or install the Universal Agent with the Python binding option (--python yes). This option will install python 3.6. along with your universal agent.

   e.g.

   sudo sh ./unvinst --network_provider oms --oms_servers 7878@192.168.88.12 --oms_port 7878 --oms_autostart no --ac_netname OPSAUTOCONF --opscli yes --python yes

2. **Add the required python modules**

   In a command shell run as sudo or root:

   - For Python the following modules are required:
     - *pip install hdfs*
       or in case of universal Agent with python binding:
       /opt/universal/python3.6/bin/python3 -m pip install hdfs

     Only run these if not available already:
     - *pip install datetime*
     - *pip install logging*
     - *pip install os*

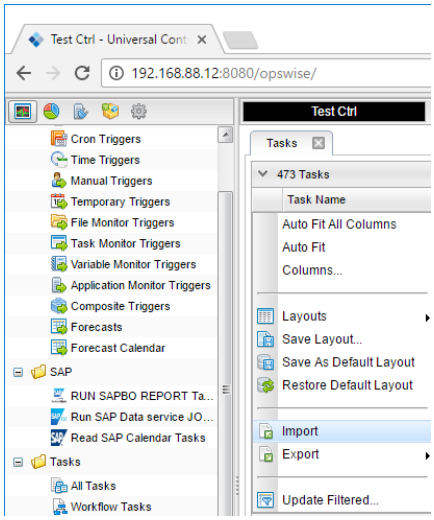
Note:

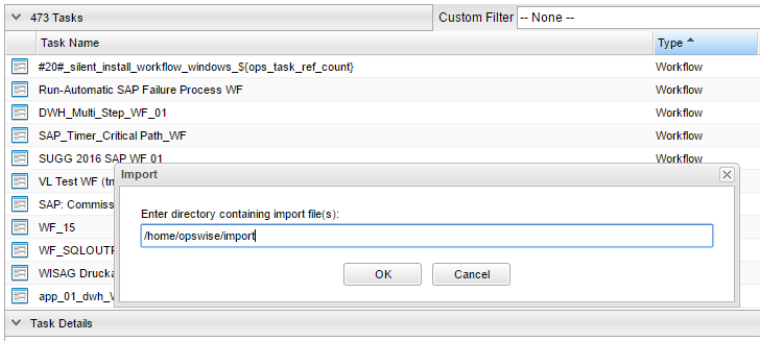
It is assumed that the modules logging, sys, datetime, os are already available. If not install them via pip. Only the module hdfs is usually not part of your installation.

https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads#bookmark-vs2013

3. Import each hdfs Universal Task including the Universal Template to your Controller

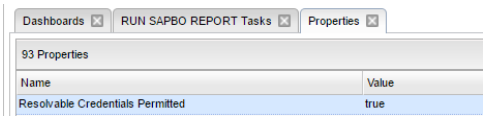
Go to “All Tasks” and load via the Import functionality the Universal Tasks configuration into the Controller.



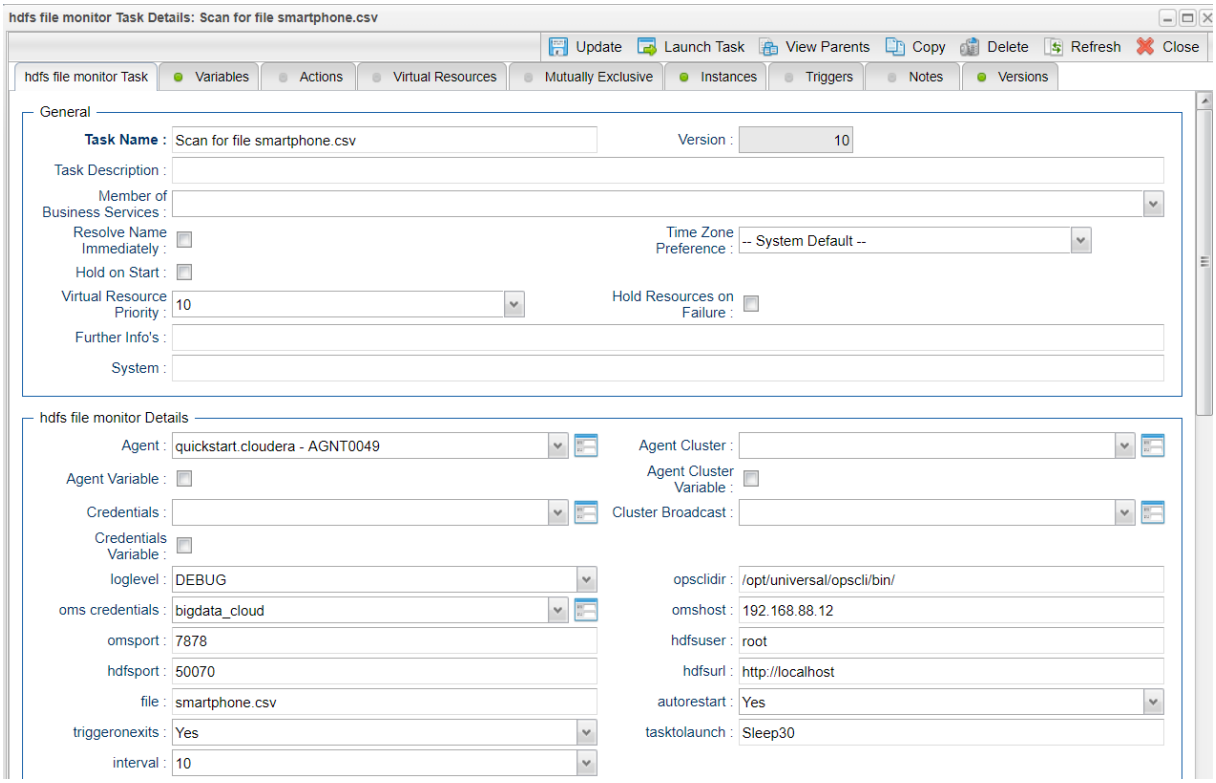


5 Universal Task Configuration

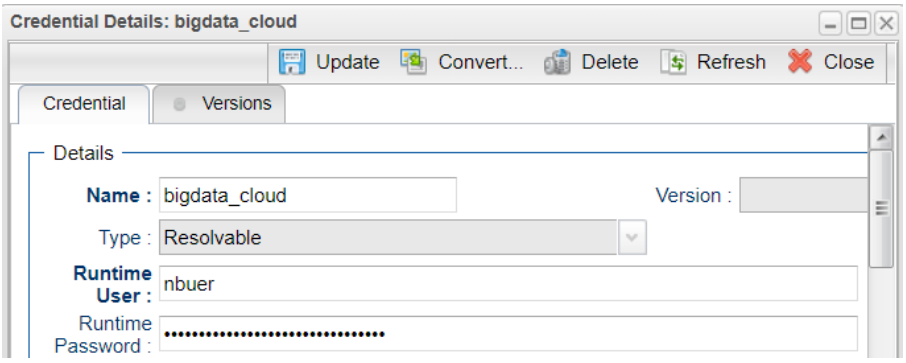
1. Activate: Resolvable Credentials in Universal Automation Center properties:



2. Fill Out each hdfs Universal Task e.g. Universal Task for hdfs file monitoring:



Credential for Universal Task:



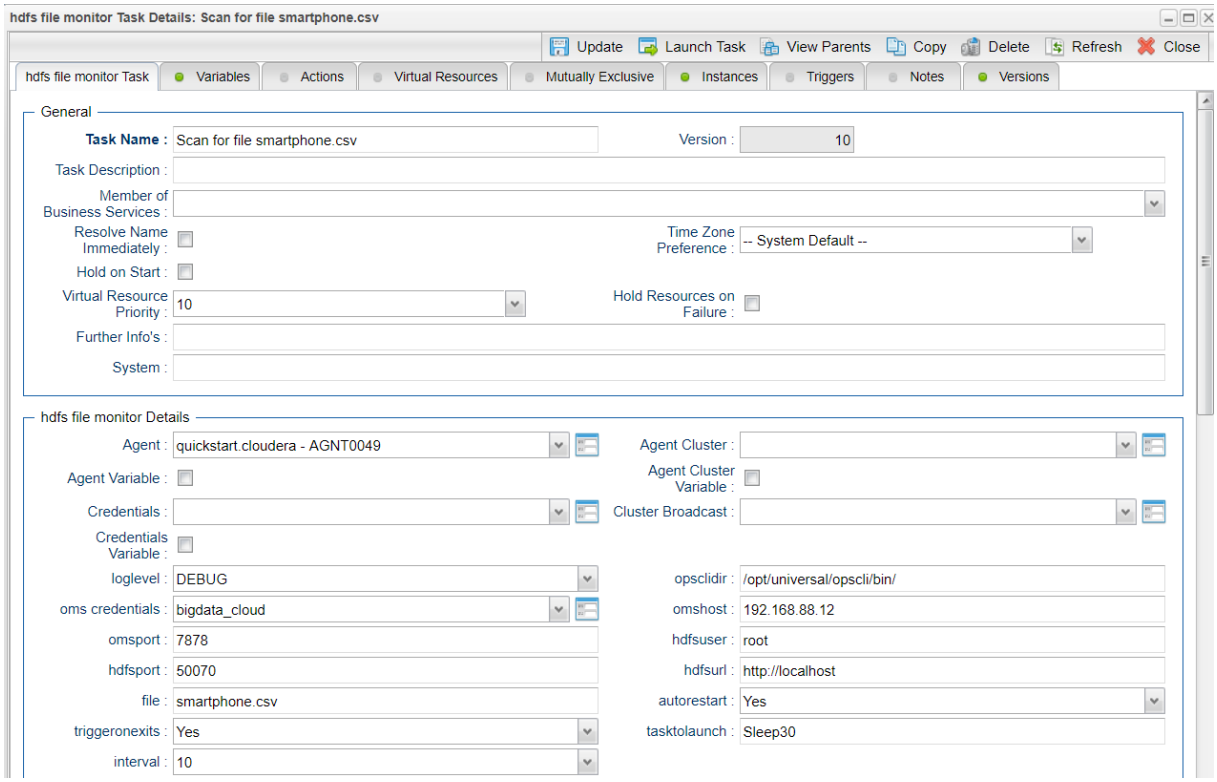
6 Universal Tasks for HDFS file system

The following chapter describes the provided HDFS Storage Universal Tasks.

6.1 Hdfs file monitor

Command	UT Name	Description
file monitor	ut-hdfs-file-monitor	Monitor file on HDFS

Task Screenshot:



Field Description:

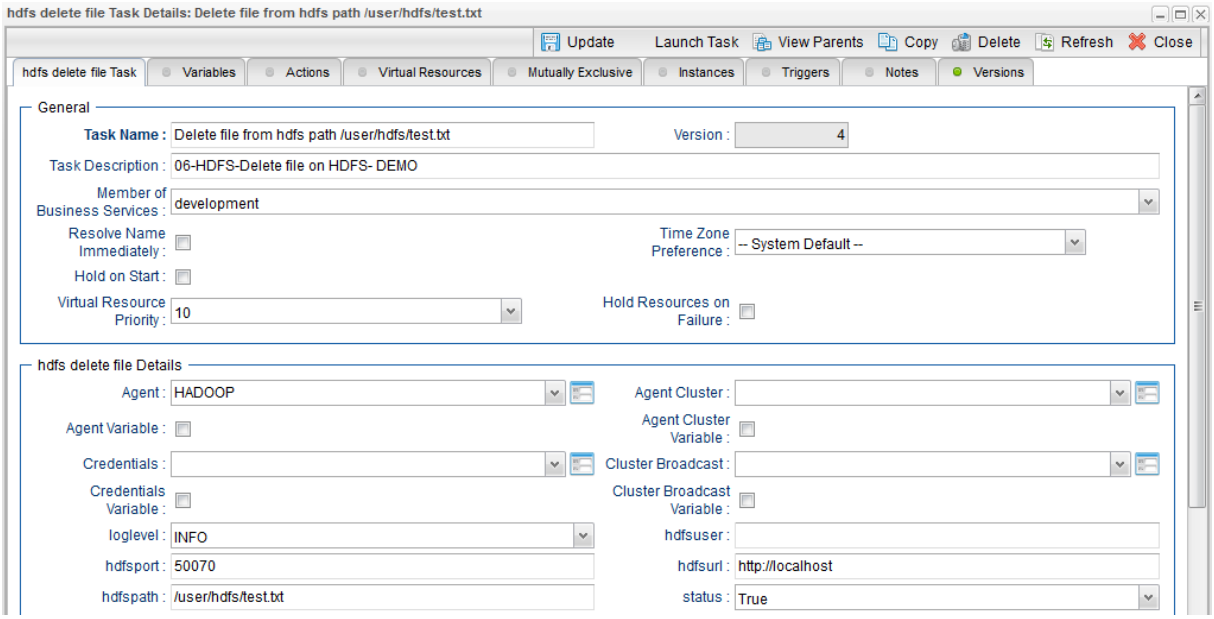
Field	Description
Agent	The Agent that runs the Python script assigned to the Universal Task
Loglevel	log level: DEBUG, INFO, WARNING, ERROR, CRITICAL
opsclidir	Directory where agent CLI executables are located default is: /opt/universal/opscli/bin/
omscred	*Credentials of the Universal Controller*
omshost	*Host of the OMS*

omsport	*Port of the OMS – default 7878*
hdfsuser	*user – User default. Defaults to the current user's (as determined by whoami). will be prefixed to all HDFS paths passed to the client*
hdfsport	*WebHDFS port on namenode default is 50070*
hdfsurl	*url – Hostname or IP address of HDFS namenode, prefixed with protocol, followed by WebHDFS port on namenode*
file	*File to monitor*
autorestart	*Defines if the monitor re-starts itself after going to success. Do not use this functionality in a Workflow Task.*
triggeronexits	*Defines if the monitor triggers also for files older than the start date of the hdfs filemonitor*
tasktolaunch	*Name of the task to launch. Note: also, wildcards are supported. E.g. sleep3* starts all task starting with sleep3 e.g. slepp30, sleep3, ..*
interval	*Define in which interval the hdfs file system is scanned.*

6.2 Hdfs file delete file

Command	UT Name	Description
delete file	ut-hdfs-delete-file	Delete file on HDFS

Task Screenshot:



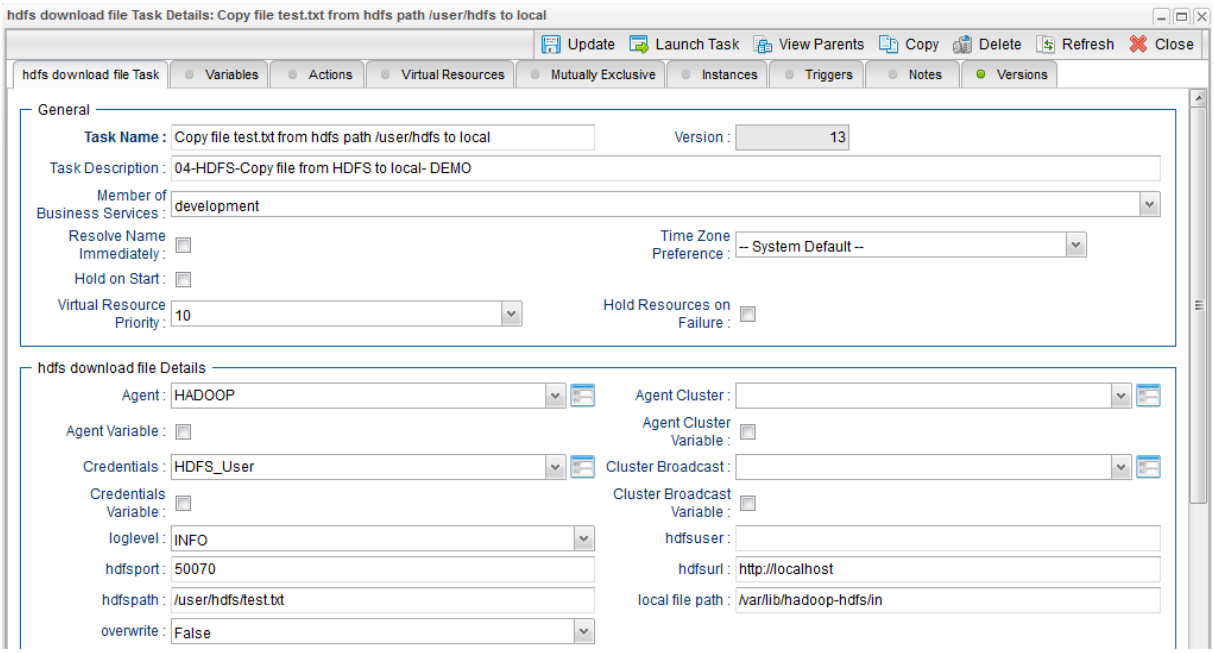
Field Description:

Field	Description
Agent	The Agent that runs the Python script assigned to the Universal Task
Loglevel	log level: DEBUG, INFO, WARNING, ERROR, CRITICAL
opsclidir	Directory where agent CLI executables are located default is: /opt/universal/opscli/bin/
omscred	*Credentials of the Universal Controller*
omshost	*Host of the OMS*
omsport	*Port of the OMS – default 7878*
hdfsuser	*user – User default. Defaults to the current user's (as determined by whoami). will be prefixed to all HDFS paths passed to the client*
hdfsport	*WebHDFS port on namenode default is 50070*
hdfsurl	*url – Hostname or IP address of HDFS namenode, prefixed with protocol, followed by WebHDFS port on namenode*
hdfspath	*Path to the file to delete on HDFS*
Status	*TRUE/FALSE* *FALSE: Only files are listed, like shell "ls" command* *TRUE: List details of the file like shell "ls -la" command*

6.3 Hdfs download file

Command	UT Name	Description
download file	ut-hdfs-download-file	Copy file from HDFS to local

Task Screenshot:



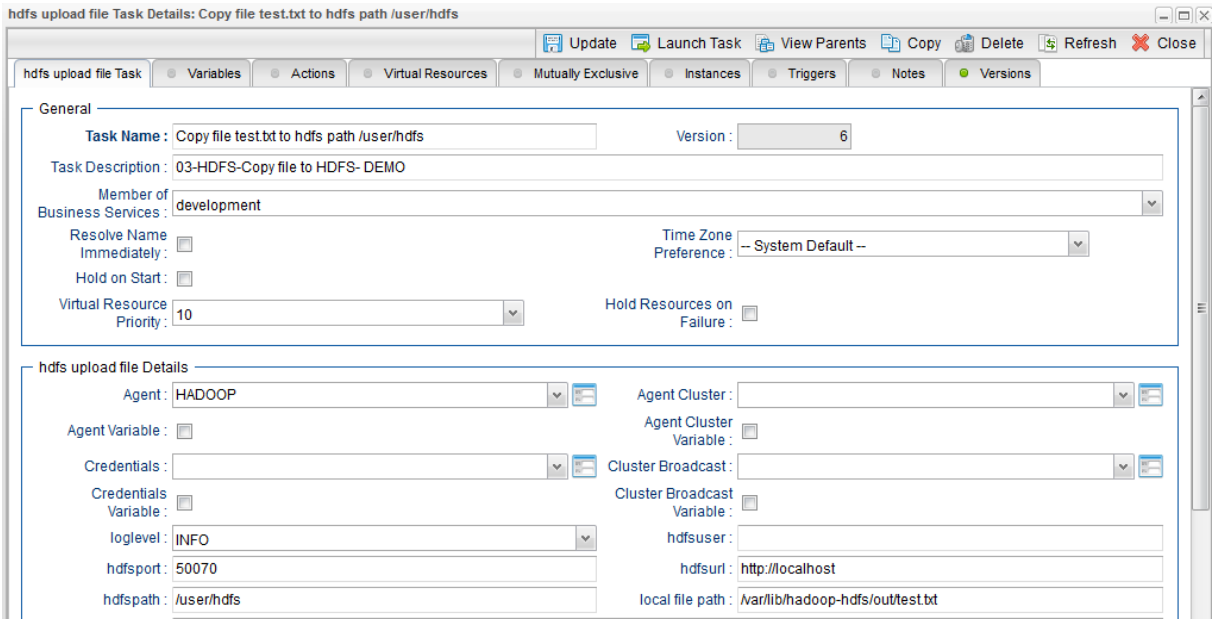
Field Description:

Field	Description
Agent	The Agent that runs the Python script assigned to the Universal Task
Loglevel	log level: DEBUG, INFO, WARNING, ERROR, CRITICAL
opsclidir	Directory where agent CLI executables are located default is: /opt/universal/opscli/bin/
omscred	*Credentials of the Universal Controller*
omshost	*Host of the OMS*
omsport	*Port of the OMS – default 7878*
hdfsuser	*user – User default. Defaults to the current user's (as determined by whoami). will be prefixed to all HDFS paths passed to the client*
hdfsport	*WebHDFS port on namenode default is 50070*
hdfsurl	*url – Hostname or IP address of HDFS namenode, prefixed with protocol, followed by WebHDFS port on namenode*
hdfspath	*Source file on HDFS – this file will be copied from hdfs to the local file path.*
Local file path	*Target path – to this directory the file will be copied to*
overwrite	*FALSE/TRUE* *Overwrite local file if exists*

6.4 Hdfs upload file

Command	UT Name	Description
upload file	ut-hdfs-upload-file	Copy file to HDFS

Task Screenshot:



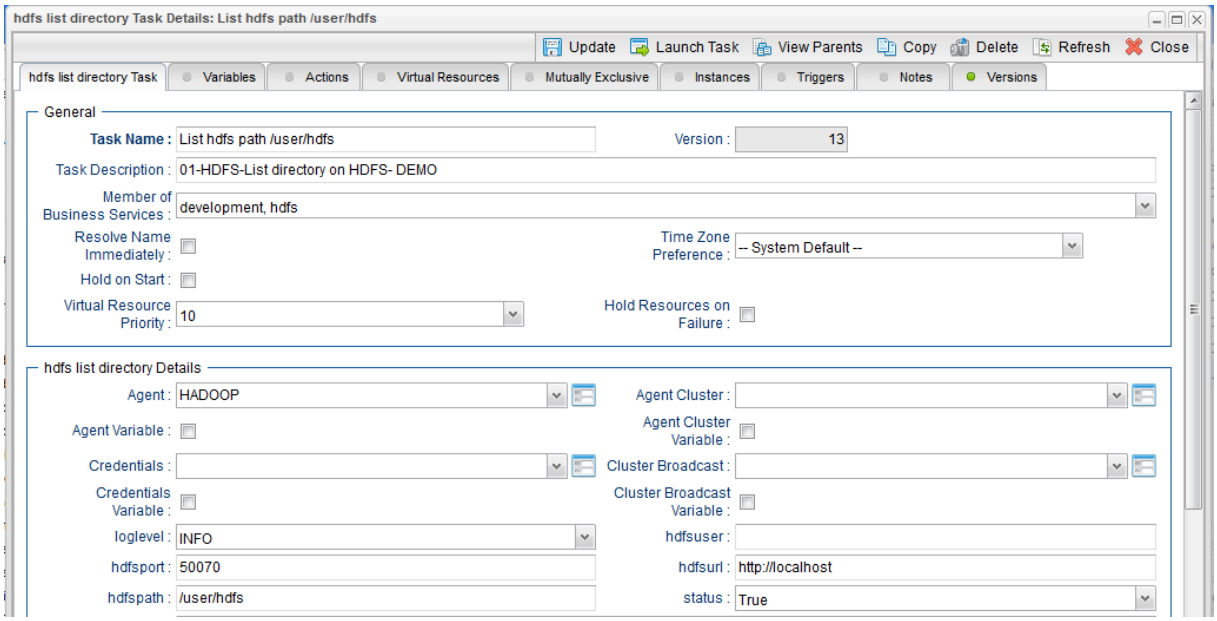
Field Description:

Field	Description
Agent	The Agent that runs the Python script assigned to the Universal Task
Loglevel	log level: DEBUG, INFO, WARNING, ERROR, CRITICAL
opsclidir	Directory where agent CLI executables are located default is: /opt/universal/opscli/bin/
omscred	*Credentials of the Universal Controller*
omshost	*Host of the OMS*
omsport	*Port of the OMS – default 7878*
hdfsuser	*user – User default. Defaults to the current user's (as determined by whoami). will be prefixed to all HDFS paths passed to the client*
hdfsport	*WebHDFS port on namenode default is 50070*
hdfsurl	*url – Hostname or IP address of HDFS namenode, prefixed with protocol, followed by WebHDFS port on namenode*
hdfspath	*target HDFS directory – the local file will be copied to this hdfs directory*
Local file path	*Source file and path – this local file will be copied to the hdfs target path*

6.5 Hdfs list directory

Command	UT Name	Description
list directory	ut-hdfs-list-directory	List directory on HDFS

Task Screenshot:



Field Description:

Field	Description
Agent	The Agent that runs the Python script assigned to the Universal Task
Loglevel	log level: DEBUG, INFO, WARNING, ERROR, CRITICAL
opsclidir	Directory where agent CLI executables are located default is: /opt/universal/opscli/bin/
omscred	*Credentials of the Universal Controller*
omshost	*Host of the OMS*
omsport	*Port of the OMS – default 7878*
hdfsuser	*user – User default. Defaults to the current user's (as determined by whoami). will be prefixed to all HDFS paths passed to the client*
hdfsport	*WebHDFS port on namenode default is 50070*
hdfsurl	*url – Hostname or IP address of HDFS namenode, prefixed with protocol, followed by WebHDFS port on namenode*
hdfspath	*Directory to list on HDFS*
Status	*TRUE/FALSE* *FALSE: Only files are listed, like shell "ls" command* *TRUE: List details of the file like shell "ls -la" command*

7 Test Cases

The following basic test cases has been performed:

Case#	Assumed behavior	Result
1) Monitor Started 2) File Copied to hdfs **Parameter** • Trigger on exists = “No” • Autorestart = “No” • Tasktolaunch = “Sleep30”	DEBUG - Starting new HTTP connection (1): localhost:5007 DEBUG http://localhost:50070 "GET /webhdfs/v1/?user.name=root&op=GETHOMEDIRECTORY HTTP/1.1" 200 None 2018-10-15 10:35:15,529 - DEBUG - Updated root to u'/user/root'. 2018-10-15 10:35:15,529 - DEBUG - Resolved path 'smartphone.csv' to u'/user/root/smartphone.csv'. 2018-10-15 10:35:15,533 - DEBUG - http://localhost:50070 "GET /webhdfs/v1/user/root/smartphone.csv?user.name=root&op=GETFILESTATUS HTTP/1.1" 200 None 2018-10-15 10:35:15,534 - DEBUG - modificationTime:1539624915175 2018-10-15 10:35:15,534 - DEBUG - current Time :1539624895468 2018-10-15 10:35:15,534 - DEBUG - Success, file:smartphone.csv was created after start of monitor 2018-10-15 10:35:15,974 - INFO - Task launched: Sleep30	Correct
1) File Copied to hdfs 2) Monitor Started **Parameter** • Trigger on exists = “Yes” • Autorestart = “No” • Tasktolaunch = “Sleep30”	DEBUG - file was created before start of monitor DEBUG - Success file:smartphone.csv was created before start of monitor but triggeronexits was set INFO - Task launched: Sleep30	Correct
1) Monitor Started 2) File Copied to hdfs **Parameter** • Trigger on exists = “No”	DEBUG - Success, file:smartphone.csv was created after start of monitor INFO - Task launched: Sleep30 INFO - Task launched: Scan for file smartphone.csv	Correct

• Autorestart = “Yes” • Tasktolaunch = “Sleep30”		
1) File Copied to hdfs 2) Monitor Started 3) File Copied to hdfs **Parameter** • Trigger on exists = “Yes” • Autorestart = “Yes” • Tasktolaunch = “Sleep30”	INFO - Task launched: Sleep30 INFO - Task Scan for file smartphone.csv re-started due to autorestart set to Yes INFO - Task launched: Sleep30 INFO - Task Scan for file smartphone.csv re-started due to autorestart set to Yes ..	Correct
1) File Copied to hdfs 2) Monitor Started **Parameter** • Trigger on exists = “Yes” • Autorestart = “Yes” • Tasktolaunch = “”	2018-10-15 13:41:08,696 - DEBUG - Resolved path 'smartphone.csv' to '/user/root/smartphone.csv'. 2018-10-15 13:41:08,701 - DEBUG - http://localhost:50070 "GET /webhdfs/v1/user/root/smartphone.csv?user.name=root&op=GETFILESTATUS HTTP/1.1" 200 None 2018-10-15 13:41:08,702 - DEBUG - modificationTime:1539635266169 2018-10-15 13:41:08,702 - DEBUG - current Time :1539636068644 2018-10-15 13:41:08,702 - DEBUG - file was created before start of monitor 2018-10-15 13:41:08,702 - DEBUG - Success file:smartphone.csv was created before start of monitor but triggeronexits was set	Correct
Wrong opsclidir	2018-10-15 13:44:24,384 - ERROR - Directory not found: /opt/universal/opscli/binx/	
Wrong omscred	success CLI requires valid user credentials. opscmd-complete	Error handling needs to be added
Wrong omshost	failed Network error: ops.agent.OPSCMDP.6E6B329247CC458B83A8E83DBD4A9FD1, ConnectNetwork, connect, 113, No route to host (192.168.88.11:7878)	correct

	Network error: ops.agent.OPSCMDP.6E6B329247CC458B83A8E83DBD4A9FD1, WmsgOmsConnect, WmsgOmsConnect, 10, Failed creating OMSConnection thread: time out Error sending command request. Failed creating OMSConnection thread: time out 2018-10-15 14:17:29,432 - ERROR - ##ERROR# Error Code: 2	
Wrong omsport	failed Network error: ops.agent.OPSCMDP.6D7475DC8FA84827BCE36B1772160525, ConnectNetwork, connect, 111, Connection refused (192.168.88.12:7879) Network error: ops.agent.OPSCMDP.6D7475DC8FA84827BCE36B1772160525, WmsgOmsConnect, WmsgOmsConnect, 10, Failed creating OMSConnection thread: time out Error sending command request. Failed creating OMSConnection thread: time out 2018-10-15 14:18:36,483 - ERROR - ##ERROR# Error Code: 2	correct
Wrong hdfsport	Failed: requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=50071): Max retries exceeded with url: /webhdfs/v1/?user.name=root&op=GETHOMEDIRECTORY (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fafd3182588>: Failed to establish a new connection: [Errno 111] Connection refused',))	correct
Wrong hdfsurl	Failed: requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhosts', port=50070): Max retries exceeded with url: /webhdfs/v1/?user.name=root&op=GETHOMEDIRECTORY (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f326e5ca588>: Failed to establish a new connection: [Errno -2] Name or service not known',))	correct

8 Document References

This document references the following documents:

Ref#	Description
[1] XML extract of Universal Task	UAC XML extract of the Universal Template and Task • all-hdfs-templates-28-11-2019.zip