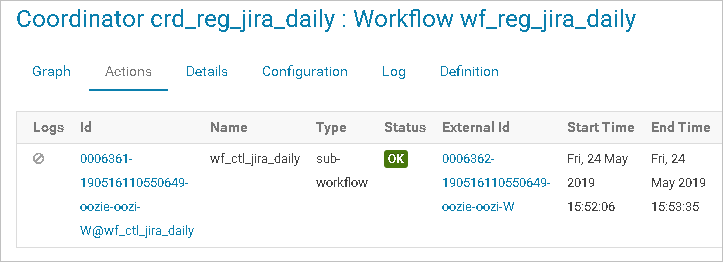
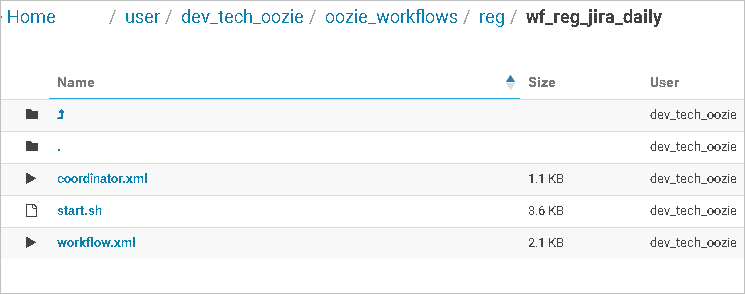
**Состав oozie-потоков загрузки из jira**

1. **reg-поток** [**wf\_reg\_jira\_daily**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/reg/wf_reg_jira_daily)





[**coordinator.xml**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/reg/wf_reg_jira_daily/coordinator.xml):

<coordinator-app name="crd\_reg\_jira\_daily"

frequency="${coord:days(1)}"

start="${coord\_start\_time}"

end="${coord\_end\_time}"

timezone="GMT+03:00"

xmlns="uri:oozie:coordinator:0.4">

<action>

<workflow>

<app-path>${workflow\_application\_path}</app-path>

<configuration>

<property>

<name>LOAD\_DATE\_FROM</name>

<value>${coord:formatTime(coord:dateOffset(coord:nominalTime(), -7, 'DAY'), 'yyyyMMdd')}</value>

</property>

<property>

<name>LOAD\_DATE\_FROM\_3d</name>

<value>${coord:formatTime(coord:dateOffset(coord:nominalTime(), -3, 'DAY'), 'yyyyMMdd')}</value>

</property>

<property>

<name>LOAD\_DATE\_TO</name>

<value>${coord:formatTime(coord:dateOffset(coord:nominalTime(), 1, 'DAY'), 'yyyyMMdd')}</value>

</property>

<property>

<name>current\_day</name>

<value>${coord:formatTime(coord:nominalTime(), 'dd.MM.yyyy')}</value>

</property>

</configuration>

</workflow>

</action>

</coordinator-app>

[**workflow.xml**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/reg/wf_reg_jira_daily/workflow.xml):

<workflow-app name="wf\_reg\_jira\_daily" xmlns="uri:oozie:workflow:0.5">

<parameters>

<property>

<name>wf\_reg\_name</name>

<value>wf\_reg\_jira\_daily</value>

</property>

</parameters>

<global>

<configuration>

<property>

<name>oozie.launcher.mapred.job.queue.name</name>

<value>tech\_oozie\_launchers</value>

</property>

<property>

<name>oozie.launcher.mapreduce.map.memory.mb</name>

<value>4096</value>

</property>

<property>

<name>oozie.launcher.mapreduce.map.java.opts</name>

<value>-Xmx4096m</value>

</property>

<property>

<name>oozie.launcher.yarn.app.mapreduce.am.resource.mb</name>

<value>4096</value>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>2048</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>4096</value>

</property>

<property>

<name>mapreduce.map.java.opts</name>

<value>-Xmx6144m</value>

</property>

<property>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx12288m</value>

</property>

</configuration>

</global>

<start to="wf\_ctl\_jira\_daily"/>

<kill name="Kill">

<message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>

</kill>

<action name="wf\_ctl\_jira\_daily">

<sub-workflow>

<app-path>${nameNode}${wf\_ctl\_path}/wf\_ctl\_jira\_daily</app-path>

<propagate-configuration/>

</sub-workflow>

<ok to="End"/>

<error to="Kill"/>

</action>

<end name="End"/>

</workflow-app>

[**start.sh**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/reg/wf_reg_jira_daily/start.sh)

#!/usr/bin/env bash

#VERSION1

err=0

err\_check() {

if [[ err -eq 1 ]];

then

read -p "WARNING: Errors detected! Continue? (Y/n): " answer

[[ ! ${answer} =~ ^("y"|"Y") ]] && exit 1

fi

}

import\_user\_properties() {

if [[ -f /home/${user\_name}/properties/oozie\_params.sh ]]

then

source /home/${user\_name}/properties/oozie\_params.sh

else

echo "ERROR: There is no oozie\_params.sh in /home/${user\_name}/properties/"

err=1

fi

}

arg\_parse() {

if [[ $# > 0 ]]

then

for arg in "$@"; do

[[ ${arg} =~ "custom" ]] && custom\_mode=1

done

fi

}

custom\_mode() {

## TODO convert to list

if [[ ${custom\_mode} -eq 1 ]]; then

hive2\_server\_principal=${custom\_hive2\_server\_principal}

hive2\_jdbc\_url=${custom\_hive2\_jdbc\_url}

oracle\_jdbc\_url=${custom\_oracle\_jdbc\_url}

oracle\_jdbc\_user=${custom\_oracle\_jdbc\_user}

oracle\_jdbc\_password=${custom\_oracle\_jdbc\_password}

coord\_start\_time=${custom\_coord\_start\_time}

coord\_end\_time=${custom\_coord\_end\_time}

DATABASE=${custom\_DATABASE}

TMP\_DATABASE=${custom\_TMP\_DATABASE}

PATH\_TO\_TMP=${custom\_PATH\_TO\_TMP}

TMP\_TABLE\_TYPE=${custom\_TMP\_TABLE\_TYPE}

echo RUNNING IN CUSTOM CONFIGURATION!

fi

}

user\_name=$(whoami) #DO NOT REMOVE!

# Params to Configure

#------------------------------------------------------------------

# Cluster settings moved to oozie\_params.sh (/home/<USERNAME>/properties/oozie\_params.sh)

# Workflow Path Parameters moved to oozie\_params.sh (/home/<USERNAME>/properties/oozie\_params.sh)

#------------------------------------------------------------------

# Hive settings moved to oozie\_params.sh (/home/<USERNAME>/properties/oozie\_params.sh)

# ADD YOUR JOB PROPERTIES HERE:

KERBEROS\_PRINCIPAL="dev\_tech\_oozie@BDA.MOSCOW.ALFAINTRA.NET"

KERBEROS\_KEYTAB="/user/dev\_tech\_oozie/config/dev\_tech\_oozie.keytab"

JIRA\_USER="debitsvzn\_tech\_user"

JIRA\_PASSWORD="6yfn^THC"

INITIAL\_LOAD="n"

#Not Used start-end by scripts but must be set

coord\_start\_time=2019-03-26T10:00+0300

coord\_end\_time=2019-04-20T11:00+0300

DATABASE=d\_sourcedata

TMP\_DATABASE=d\_sourcedata

#------------------------------------------------------------------

### Injection for running jobs inluding personal parameters

arg\_parse $@

import\_user\_properties

custom\_mode

err\_check

# Workflow to Run ##TODO needed to be automatic path

workflow\_application\_path=${wf\_reg\_path}/wf\_reg\_jira\_daily

echo "Run oozie job.."

# Run Oozie Job

oozie job -auth KERBEROS \

-D nameNode="${nameNode}" \

-D wf\_reg\_path="${wf\_reg\_path}" \

-D wf\_ctl\_path="${wf\_ctl\_path}" \

-D wf\_atom\_path="${wf\_atom\_path}" \

-D workflow\_application\_path="${workflow\_application\_path}" \

-D coord\_start\_time="${coord\_start\_time}" \

-D coord\_end\_time="${coord\_end\_time}" \

-D ORA\_USER="${ORA\_USER}" \

-D ORA\_PASSWORD="${ORA\_PASSWORD}" \

-D DATABASE="${DATABASE}" \

-D TMP\_DATABASE="${TMP\_DATABASE}" \

-D PATH\_TO\_TMP="${PATH\_TO\_TMP}" \

-D TMP\_TABLE\_TYPE="${TMP\_TABLE\_TYPE}" \

-D INITIAL\_LOAD="${INITIAL\_LOAD}" \

-D KERBEROS\_KEYTAB="${KERBEROS\_KEYTAB}" \

-D KERBEROS\_PRINCIPAL="${KERBEROS\_PRINCIPAL}" \

-D jobTracker="${jobTracker}" \

-D hive2\_server\_principal="${hive2\_server\_principal}" \

-D hive2\_jdbc\_url="${hive2\_jdbc\_url}" \

-D hcat\_metastore\_uri="${hcat\_metastore\_uri}" \

-D oracle\_jdbc\_url="${oracle\_jdbc\_url}" \

-D oracle\_jdbc\_user="${oracle\_jdbc\_user}" \

-D oracle\_jdbc\_password="${oracle\_jdbc\_password}" \

-D user\_name="${user\_name}" \

-D connection\_string="${connection\_string}" \

-D JIRA\_USER="${JIRA\_USER}" \

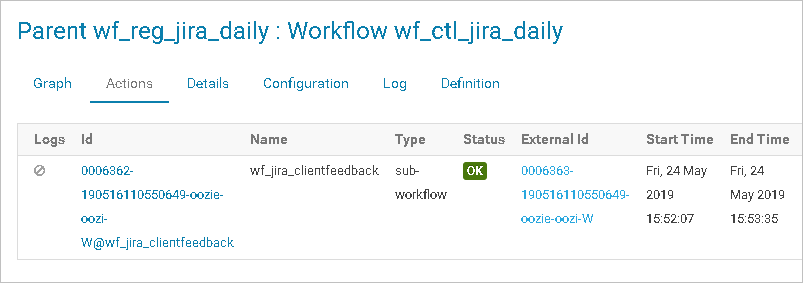
-D JIRA\_PASSWORD="${JIRA\_PASSWORD}" \

-D "oozie.coord.application.path"="${workflow\_application\_path}" \

-D "oozie.use.system.libpath"=true \

-run

1. ctl-ник [**wf\_ctl\_jira\_daily**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/ctl/wf_ctl_jira_daily)





<workflow-app name="wf\_ctl\_jira\_daily" xmlns="uri:oozie:workflow:0.5">

<parameters>

<property>

<name>wf\_ctl\_name</name>

<value>wf\_ctl\_jira\_daily</value>

</property>

</parameters>

<start to="wf\_jira\_clientfeedback"/>

<kill name="Kill">

<message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>

</kill>

<action name="wf\_jira\_clientfeedback"><!--Stream Count: 1-->

<sub-workflow>

<app-path>${nameNode}${wf\_atom\_path}/wf\_jira\_clientfeedback</app-path>

<propagate-configuration/>

</sub-workflow>

<ok to="End"/>

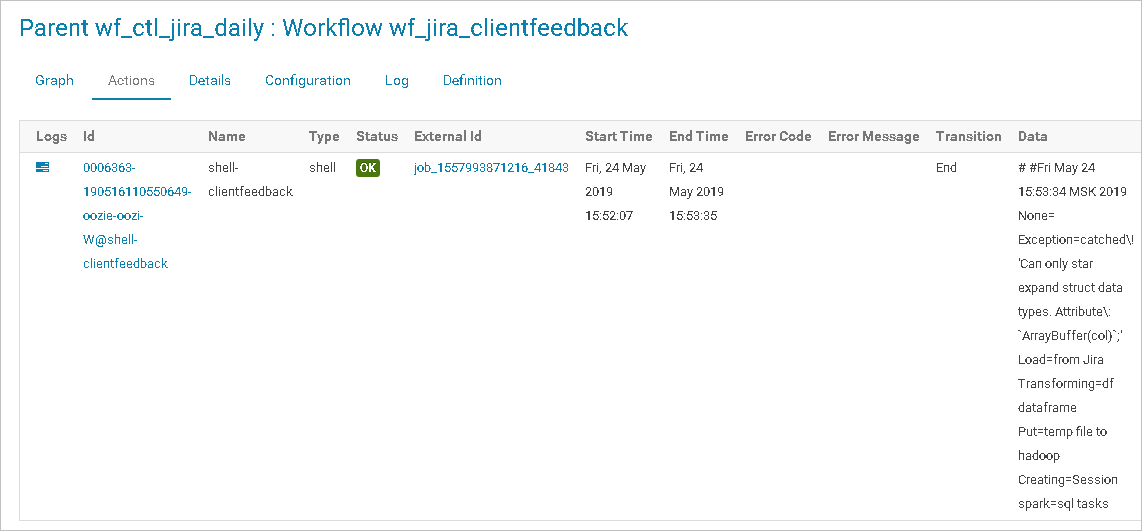
<error to="Kill"/>

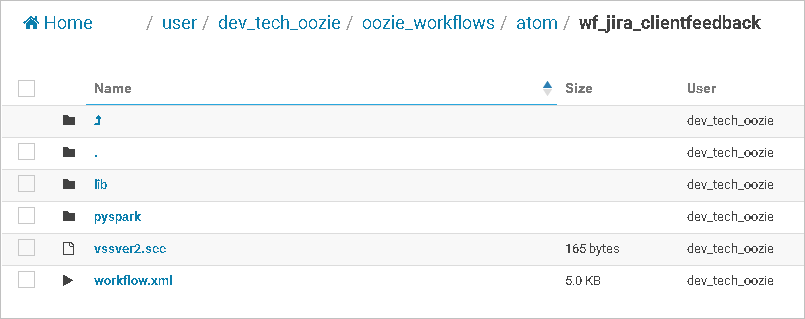
</action>

<end name="End"/>

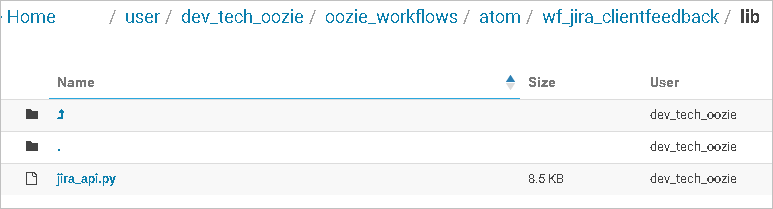
</workflow-app>

3. Атом **wf\_jira\_clientfeedback**





Каталог **lib:**



[**jira\_api.py**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/atom/wf_jira_clientfeedback/lib/jira_api.py):

#!/usr/bin/env python

import json

import sys

import requests

# This module uses Api Reference from Atlassian Jira

# located at

# https://docs.atlassian.com/software/jira/docs/api/REST/7.12.0

def jira\_login(username, password, jira\_endpoint='http://jiraft.moscow.alfaintra.net/rest/auth/1/session'):

data = {

"username": username,

"password": password

}

headers = {

'Content-Type': 'application/json'

}

try:

response = requests.post(jira\_endpoint, headers=headers, json=data)

response.raise\_for\_status()

except requests.exceptions.HTTPError as err:

print('Http Error occured')

print(f"Response is {err.response.content}")

print(f"HTTP Code is {response.status\_code}")

return response.json()

def get\_issues\_by\_project(project, jira\_endpoint='http://jiraft.moscow.alfaintra.net/rest/api/2/',

user=None, password=None, startpoint=0, expand='', session=None,

fields=['\*all'], datefrom=None):

jira\_endpoint += 'search'

results\_cnt = 50

# jql=project=ABC&maxResults=1000

if datefrom:

params = {

"jql": "project = {0} AND (updated >= {1} OR created >= {1}) ORDER BY due".format(project, datefrom),

"maxResults": results\_cnt,

"startAt": '{}'.format(int(startpoint)),

"fields": fields

}

else:

params = {

"jql": "project = {0} ORDER BY due".format(project),

"maxResults": results\_cnt,

"startAt": '{}'.format(startpoint),

"fields": fields

}

headers = {

'Content-Type': 'application/json'

}

if session:

headers['cookie'] = f"{session['session']['name']}={session['session']['value']}"

try:

if session:

response = requests.get(jira\_endpoint, params=params, headers=headers)

else:

response = requests.get(jira\_endpoint, params=params, headers=headers, auth=(user, password))

response.raise\_for\_status()

except requests.exceptions.HTTPError as err:

print(f'Http Error occured. status code is {response.status\_code}')

print(f"Response is {err.response.content}")

return response.json()

def get\_issues\_by\_type(type, jira\_endpoint='http://jiraft.moscow.alfaintra.net/rest/api/2/',

user=None, password=None, startpoint=0, expand='', session=None,

fields=['\*all'], datefrom=None):

jira\_endpoint += 'search'

results\_cnt = 50

# jql=project=ABC&maxResults=1000

if datefrom:

params = {

"jql": "type = {0} AND (updated >= {1} OR created >= {1}) ORDER BY due".format(type, datefrom),

"maxResults": results\_cnt,

"startAt": '{}'.format(int(startpoint)),

"fields": fields

}

else:

params = {

"jql": "type = {0} ORDER BY due".format(type),

"maxResults": results\_cnt,

"startAt": '{}'.format(startpoint),

"fields": fields

}

headers = {

'Content-Type': 'application/json'

}

if session:

headers['cookie'] = f"{session['session']['name']}={session['session']['value']}"

try:

if session:

response = requests.get(jira\_endpoint, params=params, headers=headers)

else:

response = requests.get(jira\_endpoint, params=params, headers=headers, auth=(user, password))

response.raise\_for\_status()

except requests.exceptions.HTTPError as err:

print(f'Http Error occured. status code is {response.status\_code}')

print(f"Response is {err.response.content}")

return response.json()

def paging\_for\_issues(filter\_value, filter\_criteria='project',

jira\_endpoint='http://jiraft.moscow.alfaintra.net/rest/api/2/',

user=None, password=None, startpoint=0, expand='', session=None,

fields=['\*all'], datefrom=None):

""" Use this method if you want PAGINATION in get\_issu

es\_by\_project() """

responses\_list = []

print(session)

if filter\_criteria == 'project':

response = get\_issues\_by\_project(filter\_value, jira\_endpoint=jira\_endpoint, user=user,

password=password, startpoint=startpoint, expand=expand,

session=session, fields=fields, datefrom=datefrom)

if filter\_criteria == 'type':

response = get\_issues\_by\_type(filter\_value, jira\_endpoint=jira\_endpoint, user=user,

password=password, startpoint=startpoint, expand=expand,

session=session, fields=fields, datefrom=datefrom)

# responses = dict(response)

responses\_list.append(response)

while dict(response)['total'] > (startpoint + dict(response)['maxResults']):

startpoint += dict(response)['maxResults']

print(f"====BEGIN====")

print(f"startpoint {startpoint}")

print(f"total {dict(response)['total']}")

print("+=====END=====+")

if filter\_criteria == 'project':

response = get\_issues\_by\_project(filter\_value, jira\_endpoint=jira\_endpoint, user=user,

password=password, startpoint=startpoint, expand=expand,

session=session, fields=fields, datefrom=datefrom)

if filter\_criteria == 'type':

response = get\_issues\_by\_type(filter\_value, jira\_endpoint=jira\_endpoint, user=user,

password=password, startpoint=startpoint, expand=expand,

session=session, fields=fields, datefrom=datefrom)

responses\_list.append(response)

return responses\_list

def get\_all\_projects(jira\_endpoint='http://jiraft.moscow.alfaintra.net/rest/api/2/',

user=None, password=None, startpoint=0, expand='', session=None):

jira\_endpoint += 'project'

params = {

'expand': '{}'.format(expand),

"maxResults": 50,

"startAt": '{}'.format(int(startpoint)),

# "fields": [

# # "summary",

# # "status",

# # "assignee"

# "\*all"

# ]

}

headers = {

'Content-Type': 'application/json'

}

if session:

headers['cookie'] = f"{session['session']['name']}={session['session']['value']}"

try:

if session:

response = requests.get(jira\_endpoint, params=params, headers=headers)

else:

response = requests.get(jira\_endpoint, params=params, headers=headers, auth=(user, password))

response.raise\_for\_status()

except requests.exceptions.HTTPError as err:

print(f'Http Error occured. status code is {response.status\_code}')

print(f"Response is {err.response.content}")

return response.json()

if \_\_name\_\_ == '\_\_main\_\_':

if len(sys.argv) == 4:

user = sys.argv[1]

password = sys.argv[2]

jira\_endpoint = sys.argv[3]

elif len(sys.argv) == 3:

user = sys.argv[1]

password = sys.argv[2]

else:

print("Unkown number of arguments, something gone wrong")

user = "any"

password = "any"

session = jira\_login(user, password, jira\_endpoint='http://jira.moscow.alfaintra.net/rest/auth/1/session')

fields = ['project',

'description',

'summary',

'subtasks',

'status',

'creator',

'assignee',

'duedate',

'issuetype',

'comment',

'created',

'resolutiondate',

'updated']

a = paging\_for\_issues("Datalake",

expand='operations,versionedRepresentations,editmeta,changelog,renderedFields',

user=user, password=password, fields=fields,

jira\_endpoint='http://jira.moscow.alfaintra.net/rest/api/2/',

)

# a = paging\_for\_issues("Dat

alake",

# expand='operations,versionedRepresentations,editmeta,changelog,renderedFields',

# session=session, jira\_endpoint='http://jira.moscow.alfaintra.net/rest/api/2/')

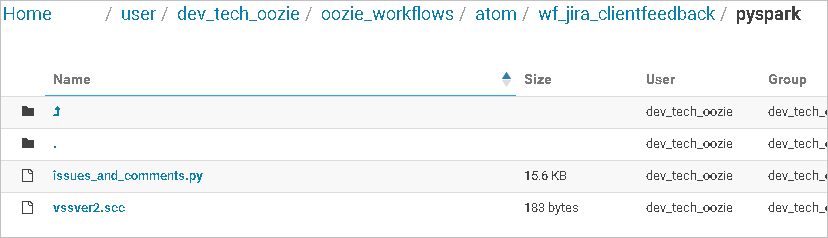
result = json.dumps(a, indent=4, sort\_keys=True)

with open('debug.json', 'w') as debug\_file:

# print(json.dumps(a, indent=4, sort\_keys=True, default=lambda x: x.\_\_dict\_\_), file=debug\_file)

print(result, file=debug\_file)

**Каталог pyspark:**



[**issues\_and\_comments.py**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/atom/wf_jira_clientfeedback/pyspark/issues_and_comments.py):

#!/opt/anaconda/bin/python3

import argparse

import sys

import os

parser = argparse.ArgumentParser(description="Arguments")

parser.add\_argument('--DATABASE')

parser.add\_argument('--TMP\_DATABASE')

parser.add\_argument('--JIRA\_USER')

parser.add\_argument('--JIRA\_PASSWORD')

parser.add\_argument('--INITIAL\_LOAD')

parser.add\_argument('--KERBEROS\_PRINCIPAL')

parser.add\_argument('--HIVE\_PRINCIPAL')

parser.add\_argument('--HIVE\_URI')

args = parser.parse\_args()

# Workaround for spark

if os.environ.get('SPARK\_HOME'):

PYSPARK\_PYTHON = os.environ['SPARK\_HOME']+'/python'

sys.path.append(PYSPARK\_PYTHON)

sys.path.append(PYSPARK\_PYTHON+'/lib/py4j-0.10.4-src.zip')

# HDFS OOZIE Workaround for local modules

sys.path.append(os.getcwd())

sys.path.append(os.getcwd()+'/lib')

import dateutil.parser

from subprocess import call

from jira\_api import \*

import pyspark.sql

from pyspark.sql.functions import \*

from pyspark.sql import SparkSession

from pyspark.sql.column import Column, \_to\_java\_column

## Because of Spark 2.2!!!

def explode\_outer\_(col):

\_explode\_outer = spark.sparkContext.\_jvm.org.apache.spark.sql.functions.explode\_outer

return Column(\_explode\_outer(\_to\_java\_column(col)))

# uncomment for local

# spark = SparkSession.builder\

# .appName('testapp') \

# .master("local") \

# .getOrCreate()

#

# try:

# call(f'kinit {args.KERBEROS\_PRINCIPAL} -kt KERBEROS\_KEYTAB', shell=True)

# os.system(f'kinit {args.KERBEROS\_PRINCIPAL} -kt KERBEROS\_KEYTAB')

# call('klist', shell=True)

# call('id', shell=True)

# call('hadoop fs -ls /user/oozie/share/lib/', shell=True)

# except Exception as e:

# print("Terminating program because of Kerberos: ", e)

# exit(1)

try:

print('Creating Session') # TODO Remove on production

spark = SparkSession.builder\

.appName('jiraEtl') \

.master('yarn-client') \

.config('spark.yarn.principal', args.KERBEROS\_PRINCIPAL) \

.config('spark.yarn.keytab', 'KERBEROS\_KEYTAB') \

.config('spark.driver.memory', '4096m') \

.config('spark.yarn.executor.memoryOverhead', '16192m') \

.config('hive.metastore.sasl.enabled', 'true') \

.config('hive.security.authorization.enabled', 'false') \

.config('hive.metastore.kerberos.principal', args.HIVE\_PRINCIPAL) \

.config('hive.metastore.uris', args.HIVE\_URI) \

.config('hive.metastore.execute.setugi', 'true') \

.enableHiveSupport() \

.getOrCreate()

except Exception as e:

print("Cannot create session, because of: ", e)

exit(1)

# .config("spark.yarn.security.tokens.hive.enabled", "false") \

# .config('spark.sql.warehouse.dir', '/user/u\_m10k4/db/') \

issues\_temp\_view = 'jira\_issues\_temp\_view'

comments\_temp\_view = 'jira\_comments\_temp\_view'

db\_schema = args.DATABASE

db\_tmp\_schema = args.TMP\_DATABASE

initial\_load = args.INITIAL\_LOAD

user = args.JIRA\_USER

password = args.JIRA\_PASSWORD

session = jira\_login(user, password, jira\_endpoint='http://jira.moscow.alfaintra.net/rest/auth/1/session')

fields = ['project',

'description',

'summary',

'subtasks',

'status',

'creator',

'assignee',

'duedate',

'issuetype',

'comment',

'created',

'resolutiondate',

'updated']

try:

print("Load from Jira") # TODO remove on production

if initial\_load == 'y':

response = paging\_for\_issues("19101", "type",

expand='operations,versionedRepresentations,editmeta,changelog,renderedFields',

user=user, password=password, fields=fields, jira\_endpoint='http://jira.moscow.alfaintra.net/rest/api/2/',)

else:

last\_date = spark.sql(f"select max(updated) from {db\_schema}.jira\_clientfeedback\_issues").head()[0]

start\_from = dateutil.parser.parse(last\_date).strftime("%Y-%m-%d")

response = paging\_for\_issues("19101", "type",

expand='operations,versionedRepresentations,editmeta,changelog,renderedF

ields',

user=user, password=password, fields=fields, jira\_endpoint='http://jira.moscow.alfaintra.net/rest/api/2/',

datefrom=start\_from)

except Exception as e:

print("Something gone wrong with JIRA API: ", e)

exit(1)

result = json.dumps(response, indent=4, sort\_keys=True)

with open('temp\_file', 'w') as temp\_file:

print(result, file=temp\_file)

try:

print("Put temp file to hadoop") # TODO remove on production

call('hadoop fs -put -f temp\_file temp\_file', shell=True)

except Exception as e:

print("Error while copying files to HDFS", e)

exit(1)

df = spark.read.json('temp\_file', multiLine=True)

try:

print('Transforming df dataframe') # TODO remove on production

df = df.select('issues') \

.withColumn('issues\_flat', explode(col('issues'))) \

.drop('issues')

df = df.select('issues\_flat.\*') \

.drop('issues\_flat')

df = df.select('id', 'key', 'self', 'fields.\*') \

.drop('fields')

df = df.select('id', 'key', 'self',

'assignee', 'comment', 'created',

'creator', 'description', 'duedate',

'issuetype', 'project', 'resolutiondate', 'status',

'summary', 'updated', 'subtasks') \

.withColumn('subtasks\_l', explode\_outer\_(col('subtasks'))) \

.drop('subtasks')

except pyspark.sql.utils.AnalysisException as e:

print('something gone wrong on base transformations: ', e)

exit(1)

try:

df = df.select('id', 'key', 'self',

df['assignee.displayName'].alias('assignee'), 'comment', 'created',

df['creator.displayName'].alias('creator'), 'description', 'duedate',

df['issuetype.name'].alias('issuetype'), df['project.key'].alias('project'),

'resolutiondate', df['status.name'].alias('status'),

'summary', 'updated', df['subtasks\_l.key'].alias('subtasks\_key')) \

.drop('subtasks\_l')

except pyspark.sql.utils.AnalysisException as e:

# If 'subtasks\_l' field is empty it is not unwrapped structure

if str(e).find('need struct type but got string') != -1: # if string will be founded

print('Exception catched!', e) # TODO remove on production

df = df.select('id', 'key', 'self',

df['assignee.displayName'].alias('assignee'), 'comment', 'created',

df['creator.displayName'].alias('creator'), 'description', 'duedate',

df['issuetype.name'].alias('issuetype'), df['project.key'].alias('project'),

'resolutiondate', df['status.name'].alias('status'),

'summary', 'updated', df['subtasks\_l'].alias('subtasks\_key')) \

.drop('subtasks\_l')

else:

print("something gone wrong with df dataframe:", e)

exit(1)

comments = df.select('comment', 'key')

comments\_are\_not\_empty = True

df = df.drop('comment')

try:

comments = comments.select('key', explode(col('comment.comments'))) \

.drop('comment') \

.select('key', 'col.\*') \

.drop('col')

except Exception as e:

if str(e).find('Can only star expand struct data types') != -1:

print('Exception catched!', e)

comments\_are\_not\_empty = False

else:

print("something gone wrong with comments dataframe", e)

else:

comments = comments.select('key', comments['author.displayName'].alias('author'),

'body', 'created', 'id', 'self',

comments['updateAuthor.displayName'].alias('updateAuthor'),

'updated')

df.createOrReplaceTempView(issues\_temp\_view)

if comments\_are\_not\_empty:

comments.createOrReplaceTempView(comments\_temp\_view)

create\_issues\_src\_table = f"""CREATE TABLE IF NOT EXISTS {db\_tmp\_schema}.jira\_clientfeedback\_issues\_src

ROW FORMAT SERDE

"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"

STORED AS INPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat"

OUTPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat"

TBLPROPERTIES ("kite.compression.type"="snappy")

AS SELECT

id, key, self, assignee, created, creator,

description, duedate, issuetype, project, resolutiondate, status, updated, summary, subtasks\_key FROM {issues\_temp\_view}

"""

create\_issues\_tmp\_table = f"""CREATE TABLE IF NOT EXISTS {db\_tmp\_schema}.jira\_clientfeedback\_issues\_tmp

ROW FORMAT SERDE

"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"

STORED AS INPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat"

OUTPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat"

TBLPROPERTIES ("kite.compression.type"="snappy")

AS SELECT

id, key, self, assignee, created, creator,

description, duedate, issuetype, project, resolutiondate, status, updated, summary, subtasks\_key FROM {issues\_temp\_view}

"""

create\_issues\_prod\_table = f"""CREATE TABLE IF NOT EXISTS {db\_schema}.jira\_clientfeedback\_issues

ROW FORMAT SERDE

"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"

STORED AS INPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat"

OUTPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat"

TBLPROPERTIES ("kite.compression.type"="snappy")

AS SELECT

id, key, self, assignee, created, creator,

description, duedate, issuetype, project, resolutiondate, status, updated, summary, subtasks\_key FROM {issues\_temp\_view}

"""

overwrite\_issues\_src\_table = f"""INSERT OVERWRITE TABLE {db\_tmp\_schema}.jira\_clientfeedback\_issues\_src

SELECT

id, key, self, assignee, created, creator,

description, duedate, issuetype, project, resolutiondate, status, updated, summary, subtasks\_key FROM {issues\_temp\_view}

"""

get\_old\_issues\_without\_increment = f"""INSERT OVERWRITE TABLE {db\_tmp\_schema}.jira\_clientfeedback\_issues\_tmp

SELECT b.id as id, b.key as key, b.self as self, b.assignee as assignee,

b.created as created, b.creator as creator, b.description as description, b.duedate as duedate,

b.issuetype as issuetype, b.project as project, b.resolutiondate as resolutiondate,

b.status as status, b.updated as updated, b.summary as summary, b.subtasks\_key as subtasks\_key

FROM {db\_schema}.jira\_clientfeedback\_issues b LEFT JOIN {db\_tmp\_schema}.jira\_clientfeedback\_issues\_src a

ON b.id = a.id WHERE a.id is null

"""

load\_increment\_issues = f"""INSERT INTO {db\_tmp\_schema}.jira\_clientfeedback\_issues\_tmp

SELECT

id, key, self, assignee, created, creator,

description, duedate, issuetype, project, resolutiondate, status, updated, summary, subtasks\_key

FROM {db\_tmp\_schema}.jira\_clientfeedback\_issues\_src

"""

reload\_tmp\_production\_issues = f"""INSERT OVERWRITE TABLE {db\_schema}.jira\_clientfeedback\_issues

SELECT

id, key, self, assignee, created, creator,

description, duedate, issuetype, project, resolutiondate, status, updated, summary, subtasks\_key

FROM {db\_tmp\_schema}.jira\_clientfeedback\_issues\_tmp

"""

create\_comments\_src\_table = f"""CREATE TABLE IF NOT EXISTS {db\_tmp\_schema}.jira\_clientfeedback\_comments\_src

ROW FORMAT SERDE

"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"

STORED AS INPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat"

OUTPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat"

TBLPROPERTIES ("kite.compression.type"="snappy")

AS SELECT

id, key, author, body, created, self, updateAuthor, updated FROM {comments\_temp\_view}

"""

create\_comments\_tmp\_table = f"""CREATE TABLE IF NOT EXISTS {db\_tmp\_schema}.jira\_clientfeedback\_comments\_tmp

ROW FORMAT SERDE

"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"

STORED AS INPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat"

OUTPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat"

TBLPROPERTIES ("kite.compression.type"="snappy")

AS SELECT

id, key, autho

r, body, created, self, updateAuthor, updated FROM {comments\_temp\_view}

"""

create\_comments\_prod\_table = f"""CREATE TABLE IF NOT EXISTS {db\_schema}.jira\_clientfeedback\_comments

ROW FORMAT SERDE

"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"

STORED AS INPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat"

OUTPUTFORMAT

"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat"

TBLPROPERTIES ("kite.compression.type"="snappy")

AS SELECT

id, key, author, body, created, self, updateAuthor, updated FROM {comments\_temp\_view}

"""

overwrite\_comments\_src\_table = f"""INSERT OVERWRITE TABLE {db\_tmp\_schema}.jira\_clientfeedback\_comments\_src

SELECT

id, key, author, body, created, self, updateAuthor, updated FROM {comments\_temp\_view}

"""

get\_old\_comments\_without\_increment = f"""INSERT OVERWRITE TABLE {db\_tmp\_schema}.jira\_clientfeedback\_comments\_tmp

SELECT b.id as id, b.key as key, b.author as author, b.body as body, b.created as created, b.self as self,

b.updateAuthor as updateAuthor, b.updated as updated

FROM {db\_schema}.jira\_clientfeedback\_comments b LEFT JOIN {db\_tmp\_schema}.jira\_clientfeedback\_comments\_src a

ON b.id = a.id WHERE a.id is null

"""

load\_increment\_comments = f"""INSERT INTO {db\_tmp\_schema}.jira\_clientfeedback\_comments\_tmp

SELECT

id, key, author, body, created, self, updateAuthor, updated

FROM {db\_tmp\_schema}.jira\_clientfeedback\_comments\_src

"""

reload\_tmp\_production\_comments = f"""INSERT OVERWRITE TABLE {db\_schema}.jira\_clientfeedback\_comments

SELECT

id, key, author, body, created, self, updateAuthor, updated

FROM {db\_tmp\_schema}.jira\_clientfeedback\_comments\_tmp

"""

try:

print('spark sql tasks') # TODO remove on production

if initial\_load == 'y':

spark.sql(create\_issues\_src\_table)

spark.sql(create\_issues\_tmp\_table)

spark.sql(create\_issues\_prod\_table)

spark.sql(create\_comments\_src\_table)

spark.sql(create\_comments\_tmp\_table)

spark.sql(create\_comments\_prod\_table)

spark.sql(overwrite\_issues\_src\_table)

spark.sql(get\_old\_issues\_without\_increment)

spark.sql(load\_increment\_issues)

if int(spark.sql(f'select count(1) from {db\_tmp\_schema}.jira\_clientfeedback\_issues\_tmp').head()[0]) > 0:

spark.sql(reload\_tmp\_production\_issues)

else:

print("Something with temp table gone wrong!")

exit(1)

if comments\_are\_not\_empty:

print('spark sql tasks for comments dataframe') # TODO remove on production

spark.sql(overwrite\_comments\_src\_table)

spark.sql(get\_old\_comments\_without\_increment)

spark.sql(load\_increment\_comments)

if int(spark.sql(f'select count(1) from {db\_tmp\_schema}.jira\_clientfeedback\_comments\_tmp').head()[0]) > 0:

spark.sql(reload\_tmp\_production\_comments)

else:

print("Something with temp table gone wrong!")

exit(1)

except Exception as e:

print("Exception while executing spark.sql:", e)

exit(1)

## For Debug reasons only!!:

# for table in ['jira\_clientfeedback\_comments', 'jira\_clientfeedback\_comments\_src',

# 'jira\_clientfeedback\_comments\_tmp', 'jira\_clientfeedback\_issues',

# 'jira\_clientfeedback\_issues\_src', 'jira\_clientfeedback\_issues\_tmp']:

# print(table)

# spark.sql(f'select count(1) from u\_m10k4.{table}').show()

#

# drop table d\_sourcedata.jira\_clientfeedback\_comments;

# drop table d\_sourcedata.jira\_clientfeedback\_comments\_src;

# drop table d\_sourcedata.jira\_clientfeedback\_comments\_tmp;

# drop table d\_sourcedata.jira\_clientfeedback\_issues;

# drop table d\_sourcedata.jira\_clientfeedback\_issues\_src;

# drop table d\_sourcedata.jira\_clientfeedback\_issues\_tmp;

[**workflow.xml**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/atom/wf_jira_clientfeedback/workflow.xml):

<workflow-app name="wf\_jira\_clientfeedback" xmlns="uri:oozie:workflow:0.5">

<parameters>

<property>

<name>wf\_reg\_name</name>

</property>

<property>

<name>wf\_ctl\_name</name>

</property>

</parameters>

<credentials>

<credential name="hive2" type="hive2">

<property>

<name>hive2.jdbc.url</name>

<value>${hive2\_jdbc\_url}</value>

</property>

<property>

<name>hive2.server.principal</name>

<value>${hive2\_server\_principal}</value>

</property>

</credential>

<credential name="hcat" type="hcat">

<property>

<name>hcat.metastore.uri</name>

<value>${hcat\_metastore\_uri}</value>

</property>

<property>

<name>hcat.metastore.principal</name>

<value>${hive2\_server\_principal}</value>

</property>

</credential>

</credentials>

<start to="shell-clientfeedback"/>

<kill name="Kill">

<message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>

</kill>

<!--Run Python Script-->

<action name="shell-clientfeedback" cred="hcat">

<shell xmlns="uri:oozie:shell-action:0.1">

<job-tracker>${jobTracker}</job-tracker>

<name-node>${nameNode}</name-node>

<exec>issues\_and\_comments.py</exec>

<argument>--DATABASE</argument>

<argument>${DATABASE}</argument>

<argument>--TMP\_DATABASE</argument>

<argument>${TMP\_DATABASE}</argument>

<argument>--JIRA\_USER</argument>

<argument>${JIRA\_USER}</argument>

<argument>--JIRA\_PASSWORD</argument>

<argument>${JIRA\_PASSWORD}</argument>

<argument>--KERBEROS\_PRINCIPAL</argument>

<argument>${KERBEROS\_PRINCIPAL}</argument>

<argument>--INITIAL\_LOAD</argument>

<argument>${INITIAL\_LOAD}</argument>

<argument>--HIVE\_PRINCIPAL</argument>

<argument>${hive2\_server\_principal}</argument>

<argument>--HIVE\_URI</argument>

<argument>${hcat\_metastore\_uri}</argument>

<env-var>JAVA\_HOME=/usr/java/default</env-var>

<env-var>SPARK\_HOME=/opt/cloudera/parcels/SPARK2/lib/spark2</env-var>

<env-var>PYSPARK\_PYTHON=/opt/anaconda/bin/python3</env-var>

<file>pyspark/issues\_and\_comments.py#issues\_and\_comments.py</file>

<file>${KERBEROS\_KEYTAB}#KERBEROS\_KEYTAB</file>

<capture-output/>

</shell>

<ok to="End"/>

<error to="Kill"/>

</action>

<!--<action name="pyspark-clientfeedback">-->

<!--<spark xmlns="uri:oozie:spark-action:0.2">-->

<!--<job-tracker>${jobTracker}</job-tracker>-->

<!--<name-node>${nameNode}</name-node>-->

<!--<configuration>-->

<!--<property>-->

<!--<name>oozie.launcher.mapred.child.env</name>-->

<!--<value>PYSPARK\_PYTHON=/opt/anaconda/bin/python3</value>-->

<!--</property>-->

<!--<property>-->

<!--<name>oozie.launcher.mapred.child.env</name>-->

<!--<value>PYTHONPATH=PYTHONPATH:/opt/cloudera/parcels/SPARK2/lib/spark2/python:/opt/cloudera/parcels/SPARK2/lib/spark2/python/lib/py4j-0.10.4-src.zip</value>-->

<!--</property>-->

<!--</configuration>-->

<!--<master>yarn-client</master>-->

<!--<name>ClientFeedback Jira task</name>-->

<!--<jar>issues\_and\_comments.py</jar>-->

<!--<spark-opts>&#45;&#45;conf spark.yarn.appMasterEnv.PYSPARK\_PYTHON=/opt/anaconda/bin/python3-->

<!--&#45;&#45;conf spark.yarn.appMasterEnv.PYSPARK\_DRIVER\_PYTHON=/opt/anaconda/bin/python3-->

<!--&#45;&#45;conf spark.executorEnv.PYSPARK\_PYTHON=/opt/anaconda/bin/python3-->

<!--&#45;&#45;conf spark.executorEnv.PYSPARK\_DRIVER\_PYTHON=/op

t/anaconda/bin/python3-->

<!--&#45;&#45;conf spark.executorEnv.PYTHONPATH=/opt/anaconda/bin/python3</spark-opts>-->

<!--<arg>&#45;&#45;HDFS\_SCRIPTS\_PATH</arg>-->

<!--<arg>/user/${user\_name}/scripts/</arg>-->

<!--<arg>&#45;&#45;DATABASE</arg>-->

<!--<arg>${DATABASE}</arg>-->

<!--<arg>&#45;&#45;TMP\_DATABASE</arg>-->

<!--<arg>${TMP\_DATABASE}</arg>-->

<!--<arg>&#45;&#45;JIRA\_USER</arg>-->

<!--<arg>${JIRA\_USER}</arg>-->

<!--<arg>&#45;&#45;JIRA\_PASSWORD</arg>-->

<!--<arg>${JIRA\_PASSWORD}</arg>-->

<!--<arg>&#45;&#45;INITIAL\_LOAD</arg>-->

<!--<arg>${INITIAL\_LOAD}</arg>-->

<!--<file>pyspark/issues\_and\_comments.py#issues\_and\_comments.py</file>-->

<!--</spark>-->

<!--<ok to="End"/>-->

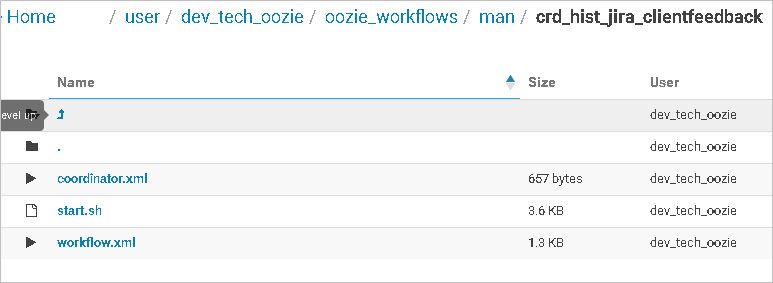
<!--<error to="Kill"/>-->

<!--</action>-->

<end name="End"/>

</workflow-app>

1. man-поток [**crd\_hist\_jira\_clientfeedback**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/man/crd_hist_jira_clientfeedback)



[**coordinator.xml**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/man/crd_hist_jira_clientfeedback/coordinator.xml):

<coordinator-app name="crd\_hist\_jira\_clientfeedback"

frequency="${coord:days(7)}"

start="${coord\_start\_time}"

end="${coord\_end\_time}"

timezone="GMT+03:00"

xmlns="uri:oozie:coordinator:0.4">

<action>

<workflow>

<app-path>${workflow\_application\_path}</app-path>

<configuration>

<property>

<name>wf\_reg\_name</name>

<value>man\_hist\_jira\_clientfeedback</value>

</property>

<property>

<name>wf\_ctl\_name</name>

<value>manual</value>

</property>

</configuration>

</workflow>

</action>

</coordinator-app>

[**workflow.xml**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/man/crd_hist_jira_clientfeedback/workflow.xml):

<workflow-app name="wf\_hist\_jira\_clientfeedback" xmlns="uri:oozie:workflow:0.5">

<global>

<configuration>

<property>

<name>oozie.launcher.mapred.job.queue.name</name>

<value>tech\_oozie\_launchers</value>

</property>

<property>

<name>oozie.launcher.mapreduce.map.memory.mb</name>

<value>4096</value>

</property>

<property>

<name>oozie.launcher.mapreduce.map.java.opts</name>

<value>-Xmx4096m</value>

</property>

<property>

<name>oozie.launcher.yarn.app.mapreduce.am.resource.mb</name>

<value>4096</value>

</property>

</configuration>

</global>

<start to="wf\_jira\_clientfeedback"/>

<kill name="Kill">

<message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>

</kill>

<action name="wf\_jira\_clientfeedback">

<sub-workflow>

<app-path>${nameNode}${wf\_atom\_path}/wf\_jira\_clientfeedback</app-path>

<propagate-configuration/>

</sub-workflow>

<ok to="End"/>

<error to="Kill"/>

</action>

<end name="End"/>

</workflow-app>

[**start.sh**](http://bda31node01.moscow.alfaintra.net:8899/filebrowser/view=/user/dev_tech_oozie/oozie_workflows/man/crd_hist_jira_clientfeedback/start.sh)

#!/usr/bin/env bash

#VERSION1

err=0

err\_check() {

if [[ err -eq 1 ]];

then

read -p "WARNING: Errors detected! Continue? (Y/n): " answer

[[ ! ${answer} =~ ^("y"|"Y") ]] && exit 1

fi

}

import\_user\_properties() {

if [[ -f /home/${user\_name}/properties/oozie\_params.sh ]]

then

source /home/${user\_name}/properties/oozie\_params.sh

else

echo "ERROR: There is no oozie\_params.sh in /home/${user\_name}/properties/"

err=1

fi

}

arg\_parse() {

if [[ $# > 0 ]]

then

for arg in "$@"; do

[[ ${arg} =~ "custom" ]] && custom\_mode=1

done

fi

}

custom\_mode() {

## TODO convert to list

if [[ ${custom\_mode} -eq 1 ]]; then

hive2\_server\_principal=${custom\_hive2\_server\_principal}

hive2\_jdbc\_url=${custom\_hive2\_jdbc\_url}

oracle\_jdbc\_url=${custom\_oracle\_jdbc\_url}

oracle\_jdbc\_user=${custom\_oracle\_jdbc\_user}

oracle\_jdbc\_password=${custom\_oracle\_jdbc\_password}

coord\_start\_time=${custom\_coord\_start\_time}

coord\_end\_time=${custom\_coord\_end\_time}

DATABASE=${custom\_DATABASE}

TMP\_DATABASE=${custom\_TMP\_DATABASE}

PATH\_TO\_TMP=${custom\_PATH\_TO\_TMP}

TMP\_TABLE\_TYPE=${custom\_TMP\_TABLE\_TYPE}

echo RUNNING IN CUSTOM CONFIGURATION!

fi

}

user\_name=$(whoami) #DO NOT REMOVE!

# Params to Configure

#------------------------------------------------------------------

# Cluster settings moved to oozie\_params.sh (/home/<USERNAME>/properties/oozie\_params.sh)

# Workflow Path Parameters moved to oozie\_params.sh (/home/<USERNAME>/properties/oozie\_params.sh)

#------------------------------------------------------------------

# Hive settings moved to oozie\_params.sh (/home/<USERNAME>/properties/oozie\_params.sh)

# ADD YOUR JOB PROPERTIES HERE:

KERBEROS\_PRINCIPAL="dev\_tech\_oozie@BDA.MOSCOW.ALFAINTRA.NET"

KERBEROS\_KEYTAB="/user/dev\_tech\_oozie/config/dev\_tech\_oozie.keytab"

JIRA\_USER="debitsvzn\_tech\_user"

JIRA\_PASSWORD="6yfn^THC"

INITIAL\_LOAD="y"

#Not Used start-end by scripts but must be set

coord\_start\_time=2019-04-02T10:00+0300

coord\_end\_time=2019-04-02T11:00+0300

DATABASE=d\_sourcedata

TMP\_DATABASE=d\_sourcedata

#------------------------------------------------------------------

### Injection for running jobs inluding personal parameters

arg\_parse $@

import\_user\_properties

custom\_mode

err\_check

# Workflow to Run ##TODO needed to be automatic path

workflow\_application\_path=${wf\_man\_path}/crd\_hist\_jira\_clientfeedback

echo "Run oozie job.."

# Run Oozie Job

oozie job -auth KERBEROS \

-D nameNode="${nameNode}" \

-D wf\_reg\_path="${wf\_reg\_path}" \

-D wf\_ctl\_path="${wf\_ctl\_path}" \

-D wf\_atom\_path="${wf\_atom\_path}" \

-D workflow\_application\_path="${workflow\_application\_path}" \

-D coord\_start\_time="${coord\_start\_time}" \

-D coord\_end\_time="${coord\_end\_time}" \

-D ORA\_USER="${ORA\_USER}" \

-D ORA\_PASSWORD="${ORA\_PASSWORD}" \

-D DATABASE="${DATABASE}" \

-D TMP\_DATABASE="${TMP\_DATABASE}" \

-D PATH\_TO\_TMP="${PATH\_TO\_TMP}" \

-D TMP\_TABLE\_TYPE="${TMP\_TABLE\_TYPE}" \

-D INITIAL\_LOAD="${INITIAL\_LOAD}" \

-D KERBEROS\_KEYTAB="${KERBEROS\_KEYTAB}" \

-D KERBEROS\_PRINCIPAL="${KERBEROS\_PRINCIPAL}" \

-D jobTracker="${jobTracker}" \

-D hive2\_server\_principal="${hive2\_server\_principal}" \

-D hive2\_jdbc\_url="${hive2\_jdbc\_url}" \

-D hcat\_metastore\_uri="${hcat\_metastore\_uri}" \

-D oracle\_jdbc\_url="${oracle\_jdbc\_url}" \

-D oracle\_jdbc\_user="${oracle\_jdbc\_user}" \

-D oracle\_jdbc\_password="${oracle\_jdbc\_password}" \

-D user\_name="${user\_name}" \

-D connection\_string="${connection\_string}" \

-D JIRA\_USER="${JIRA\_USER}" \

-D JIRA\_PASSWORD="${JIRA\_PASSWORD}" \

-D "oozie.coord.application.path"="${workflow\_application\_path}" \

-D "oozie.use.system.libpath"=true \

-run