Load Test for File Purge

Execution Plan

Scope:

- Load test will be performed using Jmeter with the JMX scripts & Stats collected as a part of Feature A6.
- Load test will be done for /filestores endpoints & Purge background job is run during this time.

How to generate files in OALCN_FILE_STORE to purge?

For volume, Run the existing script in Jmeter to generate load on File Uploader endpoints. Mark them as past run data (use some logic to spread the files creation date over a month)

In which environments should we do Load test?

We are planning to do a load test in DT. As per discussion with Raju in Pre-Dev review, load test is not required in FT.

What are the scenarios we're going to cover as a part of the load test?

1. Baseline scenario (Metrics will be gathered without running file purge background job. Stats collected as a part of Load test for Feature A6)
2. Purge job runs to delete the 1x number of files when there is 1x load on /filestores happening. (All the 3 kinds purge criteria comes into picture)
3. Purge job runs to delete the 2x number of files when there is 1.5x load on /filestores happening (All the 3 kinds purge criteria comes into picture)

Load test Execution Plan

- We will be doing Ramp up of 5mins for /filestores endpoints
- Load test will be carried out for 50 mins.
- Make the BG run after 15-20 mins of start of load test

How to do load test for CRUD API endpoints(CRUD API's to manage File Purge Criteria)?

As per discussion with Raju in Pre-Dev review: As this is master data and we don't see huge changes in the OALCN_FLP_PURGE_POLICY table, load test for these endpoints is not needed.

Load Test - Prerequisites to Gathering Metrics

The below steps guide to setting up the OALCN_FILE_STORE table, Queries to prepare the load, making necessary changes to microservice, and Queries to gather metrics.

1. Loading OALCN_FILE_STORE with files (to match the purge criteria):

The approximate number of files that needs to be purged = 1000

File Type	Total no. of files to be populated in OALCN_FILE_STORE	File Extension	Purge policy - file Age	Purge policy - Size limit	Purge policy - Count limit	Remarks
GSI_API_DATA	1800	JSON	30 days	34940	25060	900 files among 1800 violate the purge criteria for File Age.
DRM_HIERARCHY	100	ZIP	30 days	400	140	50 files among the 100 will violate the purge criteria for File Age.
ACCOUNT_CODE	35	XLSX	30 days	10	10	25 files among 35 will violate the purge criteria for Count.
CPT_COMP_HIERARCHY	10	XLSX	30 days	10	10	size of each file = 1.7MB size of 10 files = 17MB 7 MB extra space => 7/1.7 ~ 4.12 hence, 5 files from the earliest creation date will be purged as they violate the purge criteria for size.

2. Query to mark the files as past run data:

```
update oalcn_file_store set creation_date = sysdate-31 where file_name='loadtest_cpt_comp_hierarchy.xlsx';
update oalcn_file_store set creation_date = sysdate-31 where file_name='loadtest_gsi_api_data.json';
```

3. Modifications to Jmeter Script

Update the Client Id, Client secret, Scope, and Username in OAL OAuth Authenticator.

4. Changes to BG Job table

To make sure that BG is not run during the Ramp-up phase, update the LAST_EXECUTION_DATE in OALCN_FLP_BG_JOB_CONTROL to sysdate + 15 (As MAX_RUNTIME_SECONDS is 1800). So, After 30 mins, bg will run.

TODO: Test if the below configuration change works in dev environment

```
update oalcn_flp_bg_job_control set last_execution_date=sysdate + 1/96;
```

5. Updating BG config values

Background job config change to make the fixed delay as 1hr.

Config Property	Value
oalcn-fileupload.bg.purge_file_records.fixed-delay	3600000

6. Query to calculate the time taken by BG job to publish Kafka events

TODO: Code is yet to be merged to cn-development-inco-6020. Add logs around BG execution to calculate the time taken to publish kafka events.

7. Redeploy the microservice.

8. Query to check if all the eligible records are purged or not.

```
SELECT

    SUM(A.C),
    a.file_type_code,
    a.tenant_code,
    'FILE_AGE' DESCRIPTION
FROM
    (
        SELECT
            COUNT(*)             c,
            SUM(length(file_content) / 1000 / 1000),
            file_type_code,
            tenant_code,
            trunc(creation_date) creation_date
        FROM
            oalcn.oalcn_file_store lfp
        WHERE
            1 = 1
        GROUP BY
            file_type_code,
            tenant_code,
            trunc(creation_date)
    )                            a,
    oalcn.oalcn_flp_purge_policy b
WHERE
        a.creation_date < sysdate - b.file_age
    AND a.file_type_code = b.file_type_code
    AND a.tenant_code = b.tenant_code
    GROUP BY
    a.file_type_code,
    a.tenant_code
    union
```

```
SELECT
    SUM(a.s - b.size_limit),
    a.file_type_code,
    a.tenant_code,
    'FILE_SIZE' DESCRIPTION
FROM
    (
        SELECT
            SUM(length(file_content) / 1000 / 1000) total_file_size,
            file_type_code,
            tenant_code
        FROM
            oalcn.oalcn_file_store lfp
        WHERE
            1 = 1
        GROUP BY
            file_type_code,
            tenant_code
    )                           a,
    oalcn.oalcn_flp_purge_policy b
WHERE
        a.total_file_size>b.size_limit
    AND a.file_type_code = b.file_type_code
    AND a.tenant_code = b.tenant_code
    GROUP BY
    a.file_type_code,
    a.tenant_code
union
SELECT
    SUM(a.t - b.count_limit),
    a.file_type_code,
    a.tenant_code,
    'FILE_COUNT' DESCRIPTION
FROM
    (
        SELECT
            COUNT(*) T,
            file_type_code,
            tenant_code
        FROM
            oalcn.oalcn_file_store lfp
        WHERE
            1 = 1
        GROUP BY
            file_type_code,
            tenant_code
    )                           a,
    oalcn.oalcn_flp_purge_policy b
WHERE
        a.T>b.COUNT_LIMIT
    AND a.file_type_code = b.file_type_code
    AND a.tenant_code = b.tenant_code
    GROUP BY
    a.file_type_code,
    a.tenant_code;
```