**Multiple TarS Based ON Files Count**

1. **Objective**:

The objective of this new feature is for workflow to tar the files in a directory based on a configuration parameter that indicates the number of files that should be picked up for creating a tarball. This new feature will also have a tracking file to store information about Tar name and files in the tar.

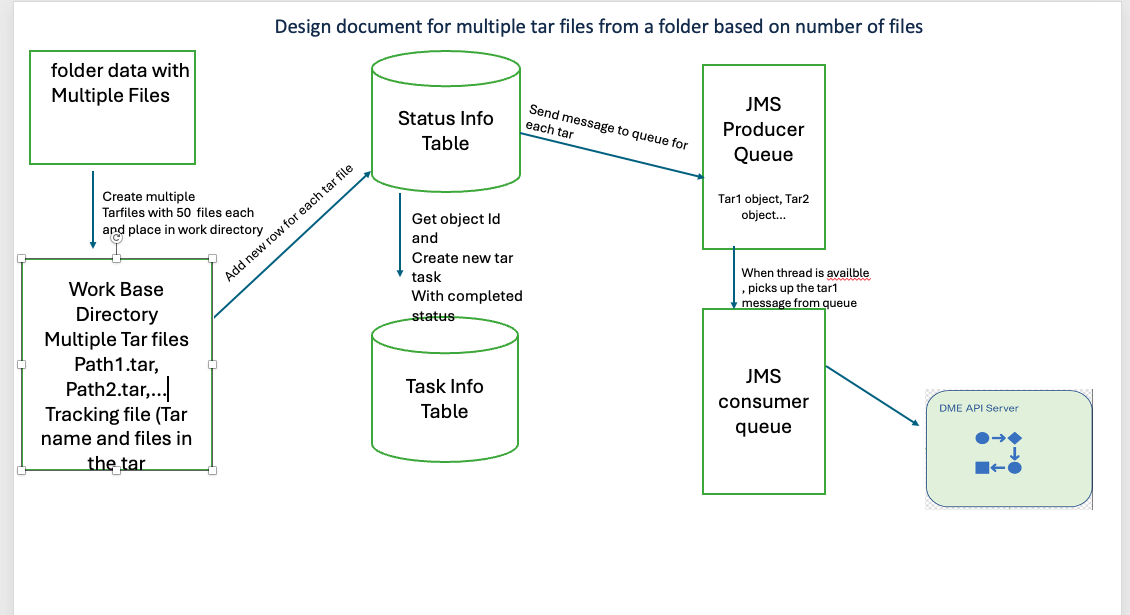
1. **Design Overview**:

The goal of this utility is to tar multiple files from a specified directory based on a given count, save them in a temporary work directory, insert row in status info table with the created tar and send a message over JMS after creating each tar archive for uploading to DME. Additionally, it provides functionality to add the tar file and its mapping information.

1. Functional requirements:
2. Movies: will create multiple tars. Tarring is based on the configuration parameter for number of files per each tar.
3. Images: Images have sub folders, so will upload whole images folder as one tar like metadata folder.
4. Metadata: currently uploading as one jar, no changes for this upload.
5. gain: Tar the folder.
6. Tracking text file: This file has the name of Tar and files in that tar.
7. Tar Contents count in Excel: After creating each tar, get the files count and save them in DB.
8. Check for no\_archive.json file: If file is present send the email notification, else check for archive.json file. If archive.json file is present continue else send the email notification.
9. Validations: Once all the Tars are created, there is one more validation which checks the total files count in movies directory with count of files in all the created tars.
10. Checks before creating Tar mainly used in Reruns: Two checks before creation of the tar mainly for rerun when there are issues.

* First: check if tarName already got uploaded to DME from statusInfo table.
* Second: If not, check if the tar is available in the temporary work directory and count of files matched. If both the cases are false, then only create the tar.

1. **Design Template**



1. Functional Requirements

4.1. Inputs

* Source directory: The directory containing the files to be tarred. dmesync.source.base.dir, dmesync.source.base.dir.folders
* Temporary work directory: The directory where the tar files will be saved temporarily. dmesync.work.base.dir= /mnt/dme\_scratch2/CSB
* Files per archive: The number of files per tar archive. dmesync.multiple.tars.files.count=50
* Folders in the directory where workflow has to perform multiple tars. dmesync.multiple.tars.dir.folders=movies
* Only exclude the archive.json , all other folders should be tarred. dmesync.exclude.pattern=\*\*/archive.json\*
* Preprocess depth to create the tar

dmesync.preprocess.depth=1, dmesync.jms.transactional=false

* dmesync.tar and dmesync.cleanup should be true
* properties for no\_archive.json changes

dmesync.file.noArchive.exist=”not\_archived.json” and dmesync.file.archive.exist=”archive.json” , dmesync.file.exist.under.basedir= true

4.2. Outputs

* Tar files: Tar archives containing the specified number of files each gets uploaded to DME.
* Mapping information notes file: store the paths of tar files or write them to a text file for reference.

1. Functionality:

* First, retrieve all the files in the directory and get work Directory location from config parameter for saving the tars temporarily.
* Check if directory as permission or not.
* Exclude any folders from the configuration parameter.
* Create the tar tracking info file in the work Directory and a Map to include Tar Name and list of files in it.
* Find the number of tars that should be created in the directory. This helps for loop count to create the tars.
* Start the loop and create the tar file based on the configuration number of files in each tar in work Directory.
* Before creating the tar, check if tar Name got uploaded or available in temp directory.
* Once tar file is created, insert the row in status\_info table with the source filename and path for newly created tar and number of files in tar.
* And mark tar task as completed for above object Id, won’t redo the tar task again.
* Send the objectId to JMS. If any consumers are available JMS will pick up the task right away and starts uploading to DME.
* Continue the next iteration, creation for second jar and so on until all the files are completed.
* Once all the tars are created, write the tracking info map to the note’s files.
* Check the count of files in directory that matched with the files in each tar. If not throw an exception.
* Update the status object with source details to the creates notes file, so current task will upload the tracking file to DME.

1. Design Considerations

* Error Handling: If exceptions occur after creating few tars, the workflow should upload already created tars and clear them from temp work directory. When we rerun again below scenario applies.
* Error Handling: what happens if all tars got successfully created and any of the next task fails for tar and didn’t get uploaded?? In this case would rerun the directory again, In the code we have logic before creating tar we check the DB row if tar got uploaded. If not, then additional check to see if the tar is available in temporary work directory. If available use the one in temporary work directory instead of creating new tar.
* Assumption: No New files gets added to the upload after successfully uploading the tar.
* Validation: After creating the tar, we are getting the count of files in the tar to storing in the Database to display in the excel report. After all the tars are created, we use the sum of counts of files in each tar to match with the files in the original directory

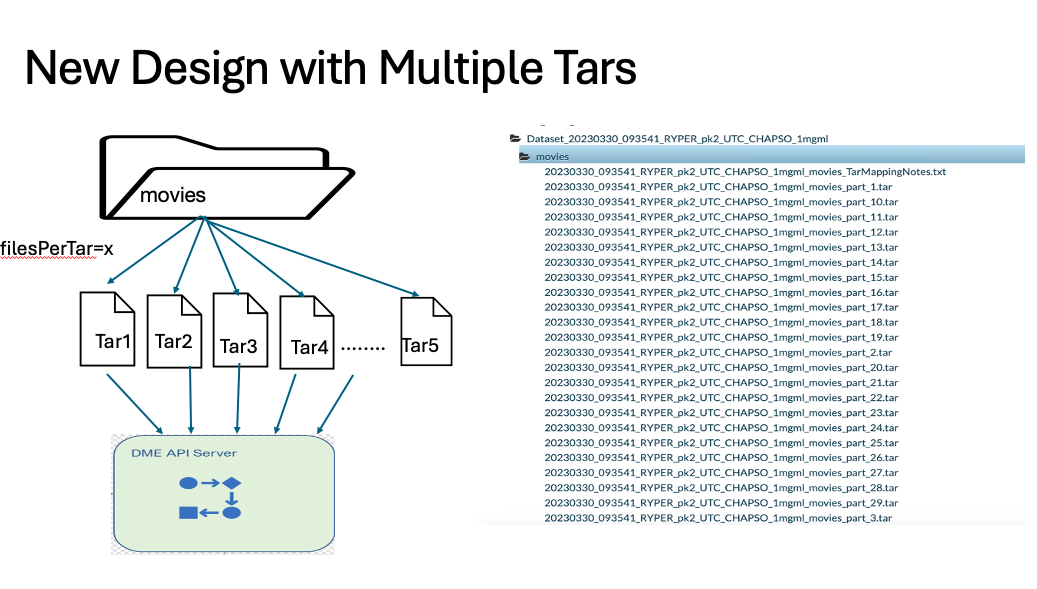
7. Testing Strategy

* Unit tests: Test individual components such as file listing, tar creation, JMS message sending, and error handling.
* Some border test scenarios:
  + Take a folder with x files. Set the tar file size configuration to be x-1 files. Then one tar should be created with all the files

Tested the scenario: folder has 9 files, configuration is set to 8, verified 2 tars are created , last tar has one file.

* + Take a folder with x + 2 files. Set the tar file configuration to be x/3 files. 4 tars should be created, with the last one having only 2 files.

Tested the scenario: X=9, folder has 12 files, configuration is set to 3, verified 4 tars are created and last one has only 2 files.

* + Test on empty folder
* Verified if the exception occurs in one of the created tars, already created tars get uploaded and cleared from the temp directory.
* Verified if the errors occur in one directory, other directories shouldn’t get effected.
* 

Statatistics Results

