Classification of Errors

1. Retriable Errors : These are the errors for which we will retry. Examples of such errors are Hosts not reachable and other intermittent connectivity issues. After we exhaust the number of retries defined by the retry policy, these errors become non-retriable errors and need manual intervention.
2. Non-Retriable Errors : These errors need manual intervention.No retry.

Error Code Range	Description	Examples
10000 to 19999	Errors occurring during semantic analysis and compilation of the query.	Table Not Found Database is not source of replication
20000 to 39999	Retriable Errors	
40000 to 49999	Non-Retriable Errors	

Current Classification of Replication Errors :

```
REPL_EVENTS_MISSING_IN_METASTORE(20016, "Notification events are missing in the meta store."),
REPL_BOOTSTRAP_LOAD_PATH_NOT_VALID(20017, "Load path {0} not valid as target database is bootstrapped " +
    "from some other path : {1}."),
REPL_FILE_MISSING_FROM_SRC_AND_CM_PATH(20018, "File is missing from both source and cm path."),
REPL_LOAD_PATH_NOT_FOUND(20019, "Load path does not exist."),
REPL_DATABASE_IS_NOT_SOURCE_OF_REPLICATION(20020,
    "Source of replication (repl.source.for) is not set in the database properties."),
REPL_INVALID_DB_OR_TABLE_PATTERN(20021,
    "Invalid pattern for the DB or table name in the replication policy. "
        + "It should be a valid regex enclosed within single or double quotes."),
REPL_FILE_SYSTEM_OPERATION_RETRY(30045, "Replication file system operation retry expired."),
GENERIC_ERROR(40000, "Exception while processing"),
```

They are all marked as retryable errors, even source of replication not set error. We need to classify them appropriately.

For other errors like ranger not reachable or atlas not reachable, we currently just throw a generic error which is not a retriable error by definition. We need to introduce new error codes and map them accordingly.

Retry Policy

Approach 1: Retry at Instance Level

Current State

This is already present. But that needs to be cleaned up and made consistent across based on the retry policy defined below. The error codes also need to be made consistent.

Retry Interface and configurations to be implemented with the schedule

1. Configs at the policy level : If we keep the configurations at the policy level, we needn't worry about policies with higher frequencies exhausting the retries faster and vice versa. A common default for all policies mayn't make sense. So it's better to keep it at the policy level.

The below configurations will be honored at the every instance level.

Name	Description	Default
Initial Delay	First retry delay in seconds	60
Backoff Coefficient	Exponential Delay between retries. Previous Delay * Backoff Coefficient will determine the next retry interval	1.2
Jitter	A random jitter to be applied to avoid all retries happening at the same time.	0-30 seconds
Max Retry Delay	Maximum allowed retry delay in seconds after including exponential backoff. If this limit is reached, retry will continue with this retry duration.	60 mins
Total Retry Duration	Total allowed retry duration in seconds inclusive of all retries.	24hrs

2. Retry will stop once Total Retry Duration is reached.
3. The error code has to be the same for the retry to be exhausted. If a new error code is seen and the previous error is resolved, the whole retry cycle will be reset and start from the beginning.
4. Even after exhausting the retries, if the instance fails, mark the instance as non-recoverable and create a marker for non-recoverable error.

Approach 2 Retry across instances

Current State

Currently the policy is always scheduled irrespective of the previous state. Even if it fails with a non-recoverable error, it is scheduled to run as per schedule as it doesn't maintain any state of previous execution.

To implement retry across instances one of the following approaches can be implemented

- Count Based : Count based has the cons that if the frequency of policy is very less, it will exhaust all retries.
- Time Based : This is recommended. A max time(configurable) is defined and till that time is reached, the executions continue as per the policy schedule.
    a. Keep track of the previous failures in the staging directory.
    b. If the previous error codes are the same and it has exceeded the time configured for max retry across instances, then fail the policy with a non-recoverable error.
    c. Create a non-recoverable marker in the staging directory.

Enforcement of Non Recoverable Errors and Manual Intervention

Query Level

1. Further instances of the same policy will pick up the non recoverable error marker from the staging directory and not proceed with any more instances till the marker is removed.
2. Cons : Admin has to manually go and remove the non recoverable marker file.(stack trace)
3. Staging location marker file.Add error code in metrics saying FAILED ADMIN. in error

Replication Manager App Level

1. RM disables the policy if it fails with a non-recoverable error.
2. Admin has to manually enable the policy.

Scheduler Level

1. The scheduler checks the state of the previous policy instance.
2. If the previous instance has failed due to irrecoverable error code, it won't trigger new instances and disable the policy.
3. Admin has to manually enable the policy.

Task Break Up

Approach 1 : Consistent Retry within the instance

Name	Timelines(in days)
Classification of Errors	3
Implement Retry Interface	2
Use the same Retry Interface across replication	3
Add Non Recoverable marker and read the marker file in the next execution.	1

Approach 2 : Retry across instances

Name	Timelines(in days)
Classification of Errors	3
Retry Across Schedules(Time Based)	3
Add Non Recoverable marker (Read and Write)	1