Fabric 6.4 iidFinder Release Notes

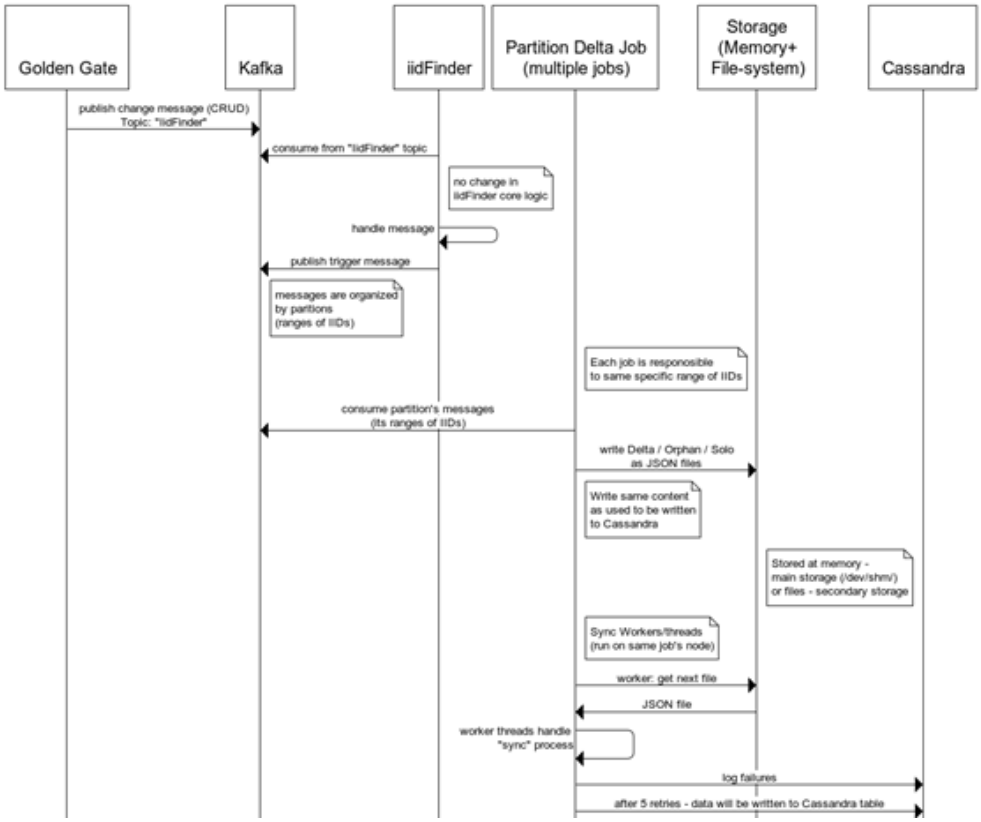
PARTITIONED DELTA MECHANISM

The Partitioned mechanism has been added in Fabric 6.4 to enhance Delta handling performance in the iidFinder. The following methodologies have adopted to support this:

- Storage: Delta content is written to memory, where possible, or to the file system and not into Cassandra.
- Partitioning: the core process outcome's data is handled by multiple jobs where each job owns a range of IIDs.
- An IID is always handled by the same job as long as the job is alive.
- Sync process: each job is also responsible for multiple worker threads that handle the sync process and that run on same node as the job where the data is stored.
- Priority is managed by a dedicated Kafka topic and thus it is faster.

Note that this new mechanism is optional and must be explicitly configured, both *Cassandra* and *Kafka* options are also supported.

The following diagram illustrates how data is handled by the new mechanism.



Click here to download the diagram.

- Each instance's data is usually contained in a single JSON file.
- Each job activates several sync workers managed by sqlite ordered.db.
  - Each entry in the sqlite contains the iid, partition, offset, update date and status.
  - The status of an sqlite entry can be:
    - Updating, while updates arrive.
    - Syncing, when a worker starts a sync. New updates that arrive during a sync are written into a new entry.
  - When the sync ends the entry is deleted.
- After five failed sync attempts, the entry is written to Cassandra and signed as handled by Kafka. This enables to continue managing messages even when a job crashed and avoid redundant sync actions to be executed again.
- Partitioned jobs are displayed in k2jobs.

Configuration Changes

iifConfig.ini

- DELTA_STORAGE – decide which mechanism to use, either of: CASSANDRA_ONLY (default) / Kafka / Partitioned (new)

config.ini

A new section named partitioned_delta has been added. The following are its main parameters:

Variable	Description	Default value	Notes
DELTA_MAIN_STORAGE	**Path to primary storage.**	**/dev/shm/delta**	**Default primary storage is in memory.**
MAIN_STORAGE_MAX_SIZE_KB	**Maximum size of main storage.**	**2973348 (~ 2.9G)**	
DELTA_SECONDARY_STORAGE	**Path to secondary path, used when primary storage exceeds capacity.**	**${FABRIC_HOME}/storage/delta**	**Optional, can be empty.**
SECONDARY_STORAGE_MAX_SIZE_KB	**Maximum size of secondary storage.**	**2973348 (~ 2.9G)**	
BOOTSTRAP_SERVERS	**Kafka server endpoint.**	**localhost:9093**	**Probably requires updating. Can be applied on the same Kafka used for other components or the iidFinder. Other Kafka parameters are hidden.**
SSL_ENABLED	**Indicates whether Kafka interactions use SSL.**	**False**	**Probably set to True. Other Kafka security parameters are hidden.**
NUM_OF_DELTA_JOBS	**Number of partitioned Delta jobs in the cluster.**	**100**	
NUM_OF_PARTITIONS	**Number of partitions.**	**100**	
PARTITIONED_DELTA_JOB_AFFINITY	**Affinity to the Partitioned Delta Job.**	**No affinity**	

Other parameters in the config file may require the following changes:

- SYNC_JOB_BULK_SIZE, default = 500, defines the sync job's bulk reading size of iids.
- SYNC_WORKERS_COUNT, default = 10, defines the number of sync threads to execute in parallel.
- SYNC_WORKERS_POOL_MAX_SIZE, default = 100, defines the maximum number of threads that can be executed on a node to run a sync.

The following two priority-related configuration parameters are currently not exposed and can be exposed if needed:

- PRIORITY_NUM_OF_PARTITIONS
- PRIORITY_NUM_OF_DELTA_JOBS

The priority is managed by a different topic and dedicated job and therefore has its own definitions that are represented by these parameters.

Setup Considerations

The following should be considered during setup:

- Kafka retention, when using the Partitioned Delta mechanism, the data is not persistent in the DB. This means that if the system crashes, the data might get abandoned. To prevent this, the Kafka log.retention.hours retention period shall be extended to ensure that the data is handled.
- Storage, the Partitioned Delta mechanism is mostly effective when data is managed in memory (aka /dev/shm). Once it exceeds its limit, the data is stored in the file system and becomes less effective. Therefore, consider the available memory that shall be allocated for the mechanism.

Implementation Recommendations

getDeltas Function

When the Partitioned Delta mechanism is used, the implementer shall use the new getDeltasStream() function exposed in the User Code API, instead of using the IidFinderApi.getDeltas() iidFinder API.

While the original function still works, using the new function enables faster handling of the higher priority delta data.

System Behavior Changes

The following changes occur when using the Partitioned Delta mechanism:

- Sync failure handling, when a sync is executed by the Partitioned Delta jobs workers, it attempts five retries during an error. If the sync fails, an error is written to the new k_<LU 3 first letters>_partitioned_delta_fail table in Cassandra and the delta datais written to Cassandra delta table which is used for other delta handling methods.

- LUI retrieval (get):
  - Running a manual get/sync via the console may take longer, since the delta data could be on a different node which requires the current node to make a remote get call to the node.
  - To avoid a remote get during migration in the initial load phase, the new sync_local_only SET command shall be used: set it to true to avoid remote get and set to false after the initial load ends. Using the SET command enables running the Fabric server with DELTA_STORAGE already configured to PARTITIONED during the initial load to prevent Fabric from restarting in between.
  - Web Services response time is not affected when using get/sync calls, since a request filter routes the request to the node handling the requested instance.

Known Issues

- The number of jobs and partitions are the same for all LUs.
- To enhance performance, JSON files are currently not encrypted.
- In case a job is crashed and reinitiated on another node, the JSON files are recreated on the that node but not deleted from the original node.

GLOBALS AS A SOURCE SCHEMA

Fabric now supports Globals as a source schema's value in an LU population property under the iidFinder section. The Globals name must be written between @ characters. For example: @BILLING_SCHEMA@.

INSTALLATION AND UPGRADE NOTES

Run the upgrade scripts and procedures of all released versions simce your current deployed release version up to 6.4.0.

Partitioned Delta Mechanism

To use the Delta mechanism, update the iidFinder as follows:

1. Stop new messages arriving via Golden Gate and wait for old messages to be handled.
2. If initial migrate shall be executed, a first deploy shall be done to load the XMLs and then run the migrate.
3. Update the configurations to use Partitioned mechanism.
4. Restart Fabric and then make a second deploy to activate the Jobs.
5. Reactivate Golden Gate Messages.

Globals as a Source Schema

To make it effective for iidFinder, on each update of the global value the following shall be done:

1. Drop k2staging, if exists.
2. Run the init tables script.
3. Restart the iidFinder.

Note that instances can only be migrated (migrate, get) once the above steps have been implemented. If an instance is migrated beforehand, drop and redeploy the LU.