



Course 11

Fabric Batch

Agenda

- What is a Fabric Batch?
- Batch command
- Managing Batch Processes
- Batch Monitoring Commands
- Batch behind the scenes
- Common tables dependencies
- Batch system_db tables
- Batch Actors
- Batch Config





What is Fabric Batch?

Fabric Batch is a built-in utility that executes `fabric_command` operations on a list of instances, leveraging multiple Fabric nodes for parallel execution.

Key Features

- **Distributed Execution**
Configure which nodes will participate in running the batch across instances.
- **Dynamic Load Balancing**
Control the number of threads per node for optimal resource usage during execution.
- **Failure Recovery**
Automatically handles unresponsive nodes to ensure smooth execution.
- **Pause & Resume**
Supports stopping and resuming of long-running migration processes.
- **Real-Time Monitoring**
Track batch progress and performance by cluster, data center, node, or IID (via CLI or the Batch Monitor Dashboard).
- **Detailed Tracking**
Monitor execution time, duration, responsible node, and failure diagnostics at the entity level.

Supported Use Cases

Instance Sync (Migration Process)

Runs sync operations across selected IIDs :

```
BATCH LU ('LUI','LUI2','LUI3','LUI4')  
FABRIC_COMMAND="sync_instance LU.?" with  
ASYNC='true';
```

Broadway Flows

Executes a Broadway flow across selected IIDs:

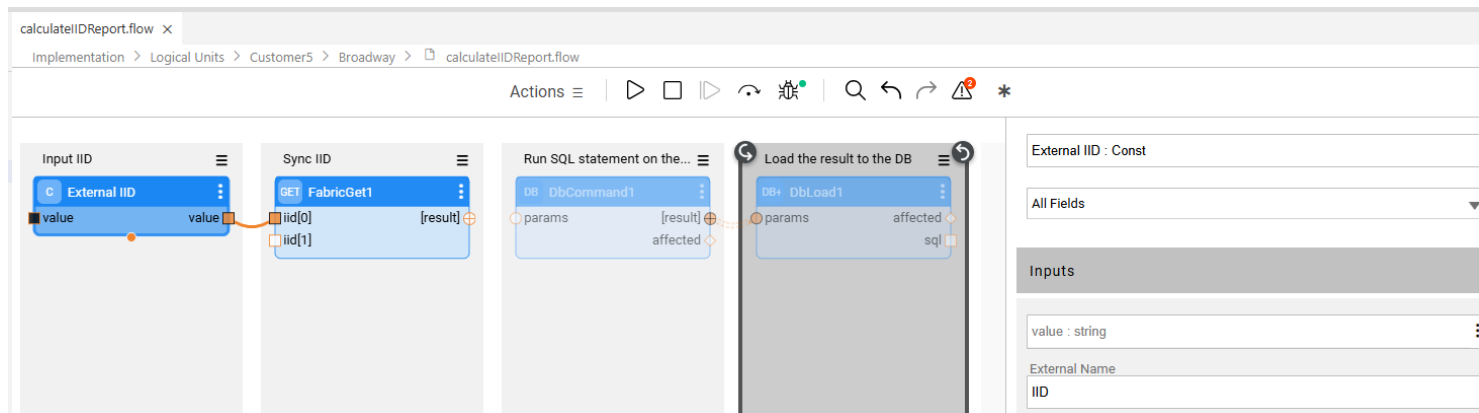
```
BATCH LU FABRIC_COMMAND="broadway  
LU.SampleFlow SampleIID=?" with async=true;
```

Example

Consider a Broadway flow that generates report data for a single instance and stores the result in a database table.

The flow performs the following steps:

- Defines an external parameter named IID, which receives its value from the batch command
- Executes a GET operation on the given IID
- Calculates the report and loads the results into the database



You can use the batch command to run this flow across a list of IIDs:

```
batch Customer5 from CRM_DB using ('select customer_id from public.customer') fabric_command='broadway Customer5.calculateIIDReport IID=?' with async=true;
```

Once the batch completes, the aggregated report data will be available in the database.



Batch Command

The Fabric Batch command allows you to execute a **fabric_command** across a set of instances, with full control over execution behavior and distribution.

Each Batch Command Defines:

1. **List of Instance IDs (IIDs)** – Which instances to process
2. **Fabric Command** – The operation to run on each instance
3. **Execution Nodes** – Which nodes in the cluster will participate
4. **Thread Configuration** – Number of threads per node (each handling one IID)



Batch Command

The Fabric Batch command allows you to execute a **fabric_command** across a set of instances, with full control over execution behavior and distribution.

Syntax:

```
BATCH IID_LIST FABRIC_COMMAND=FABRIC_COMMAND  
[WITH  
  [AFFINITY='<affinity>']  
  [JOB_AFFINITY='<job_affinity>']  
  [ASYNC=true|false]  
  [GENERATE_ENTITIES_FIRST=true|false]  
  [ALLOW_MULTIPLE=true|false]  
  [MAX_WORKERS_PER_NODE=<number>]  
  [ESTIMATED_ENTITIES_COUNT=<number>]];
```

Batch Command

IID_LIST Options:

1. Full Fabric Population

- **All instances in a LUT:**

batch `Customer` fabric_command='<fabric command>'

* query is built automatically using the SourceDbQuery of the root table

- **Based on existing entity table records :**

batch `Customer from fabric` fabric_command='<fabric command>'

2. Subset of Instances

- **Explicit list:**

batch `Customer.('1','2','3')` fabric_command='<fabric command>'

- **Instance Group:**

batch `Customer.ig10CustomersList` fabric_command='<fabric command>'

- **Source query:**

batch `Customer from CRM_DB using ('select customer_id from CUSTOMER where customer_id <= 1000')`

fabric_command='<fabric command>'

Batch Command

- **Broadway result set (must return IID list):**

batch Customer from fabric using ('broadway Customer5.getList') fabric_command='<fabric command>'

Note: The BROADWAY command, executed within the 'USING' clause, will run as if the RESULT_STRUCTURE=CURSOR

3. Messaging Queue

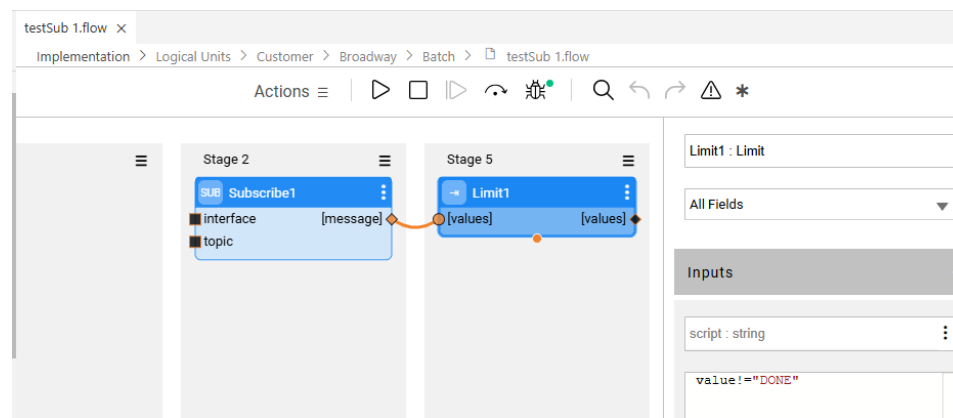
- **Dynamic instance set from a Kafka topic:**

batch Customer from KafkaInterface1 using ('topic1') fabric_command='<fabric command>'

Note: Use batch_cancel to stop the process when using a dynamic feed like Kafka.

Example:

```
batch test1 from fabric USING('BROADWAY test1.testSub ,\"topic\"=${topic} \"interface\"=\"${interfaceName}\"')
fabric_command='sync_instance test1.?' WITH ASYNC=true;
```



Batch Command

Batch command output:

Batch_id – unique identifier of the batch process. To be used later in all Fabric's batch commands.

Execution_id – used internally by Fabric.

```
fabric>batch Customer from CRM_DB using ('select customer_id from public.customer limit 200') fabric_command='sync_instance Customer.?' with async=true;
|Batch id|Execution id|Notes|
+-----+-----+-----+
|7e1d395a-1470-4426-9b3b-ee9214e5f317|913e266a-a446-407f-9ceb-ac594954a807|null|
```

Batch Command

FABRIC_COMMAND Options:

This is the operation to be executed per IID. It must contain ? as a placeholder for the instance ID.

Examples:

- **Sync (legacy migrate command):**
"sync_instance Customer.?"
- **Run a Broadway flow:**
"broadway Customer5.SampleFlow SampleIID=?"
- **Republish via CDC:**
"cdc_republish_instance Customer.?"

Batch Command

WITH Options:

Parameter	Description
AFFINITY	List of nodes or data centers to execute the Fabric Command on
JOB_AFFINITY	Affinity for the batch job itself
ASYNC	If true, runs without job mechanism (default: false)
GENERATE_ENTITIES_FIRST	If true, pre-generates all entities before execution. Mostly used for checking the performance of the functionality fetching the instance group.
ALLOW_MULTIPLE	Allows concurrent execution of the same batch command (default: false). When ALLOW_MULTIPLE is set to true, a unique UID is generated for a batch process, allowing running the same command again - before the first command is completed. The reason for that can be for example, when the subset of instances is created based on a random entity selection
MAX_WORKERS_PER_NODE	Limits the number of worker threads per node (cannot exceed the config-defined max)
ESTIMATED_ENTITIES_COUNT	Estimated number of entities to process (for monitoring/statistics only). Used until generate_iid_list ends or exceed the provided estimation
MAX_NODES	The maximum number of Fabric nodes that can participate in the batch process (random nodes). If Affinity is used – select random from the suitable nodes. Can be used, for example, when the number of connections is limited on the source side

A vertical photograph on the left side of the slide shows a person wearing a red jacket and dark pants standing on a rocky, snow-covered mountain peak. The person is looking out over a vast, snow-covered mountain range under a clear blue sky with some light clouds.

Managing Batch Processes

Canceling a Running Batch

Use the **batch_cancel** command to stop a running batch process.

- `batch_cancel '<batch_id>';`
Cancels the specified batch process, regardless of the coordinating node.
- `batch_cancel;`
Cancels the most recent **async** batch process started in the current session.

Pausing a Running Batch

Use the **batch_pause** command to pause an active batch. The batch status will be set to PAUSED, and it can later be resumed using `batch_retry` command.

- `batch_pause '<batch_id>';`
Pauses the specified batch process.
- `batch_pause;`
Pauses the most recent **async** batch process started in the current session.



Managing Batch Processes

Resuming a Paused or Cancelled Batch

Use the **batch_retry** command to resume a paused batch, or optionally retry a cancelled one.

```
batch_retry '<batch_id>' [allow_cancelled=true|false];
```

Note:

- Applicable only for async batch process.
- If the batch completed before the pause, only failed instances will be retried.
- If the batch was paused mid-execution, all remaining unprocessed instances will be executed.
- The **allow_cancelled** flag (default: false) determines whether cancelled batches can be retried. Set to true to allow retrying a cancelled batch.
- Use **batch_pause** when you plan to stop the batch temporarily:
 - A fix is needed (e.g., in the DB or implementation).
 - After the fix, you can use **batch_retry** to continue processing both failed and unprocessed entities.
- Use **batch_cancel** when you do not intend to resume the batch:
 - The process is being terminated permanently.
 - No further execution or retry is expected.

A vertical photograph on the left side of the slide shows a person wearing a red jacket and dark pants standing on a rocky, snow-covered mountain peak. The person is looking out over a vast, snow-covered mountain range under a clear blue sky with some light clouds.

Managing Batch Processes

Editing Batch Parameters at Runtime

Use the **batch_edit** command to update certain parameters of a running batch.

```
BATCH_EDIT ['<batch_id>'] param1=<value1> param2=<value2>
```

Supported Parameters: MAX_WORKERS_PER_NODE – Set the number of workers per node (must not exceed the limit in config.ini).

Note: All batch processes share the total worker pool per node.

Legacy Support: Migrate Command

The migrate command is a legacy, simplified form of the batch command, specifically designed for migrating instances into the Fabric database.

```
MIGRATE <LU>[@<DC>] WITH ASYNC='true';
```

How It Works:

Behind the scenes, Fabric translates the migrate command into a batch command. This means all Batch-related options and parameters can be used with migrate – **except**, no need to explicitly define the FABRIC_COMMAND.

Example Equivalence:

The following two commands are functionally identical:

- `MIGRATE Customer@DC1;`
- `BATCH Customer@DC1 FABRIC_COMMAND='sync_instance Customer.?';`

Use migrate for convenience when running standard sync operations. For advanced control, use batch directly.

Batch Monitoring Commands

List Batch Processes

`batch_list [STATUS='<status>'] [FROM_DATE='<from date>'] [TO_DATE='<to date>'] [FILTER='<filter criteria>'];`

Description:

- Lists batch processes based on status, time range, or specific filters.
- If no arguments are provided, only *active* batch processes are listed.

Options:

- STATUS: Possible values: NEW, GENERATE_IID_LIST, IN_PROGRESS, FAILED, CANCELLED, DONE, ALL
- FROM_DATE, TO_DATE: Use DATE_FORMAT or DATETIME_FORMAT as formatted in config.ini
- FILTER: Filters batch processes based on a substring or regex match within the batch command, fabric command, or execution ID.

Examples:

- `batch_list STATUS='ALL';`
- `batch_list STATUS='ALL' FILTER='sync_instance';`

```
fabric>batch_list STATUS='DONE' FROM_DATE='2025-04-30' filter='Customer5';
|Id|Status|Start date|Completion %|End date|Fabric command|
|-----|-----|-----|-----|-----|-----|
|Execution id|Error|
+-----+-----+-----+-----+-----+-----+
|7d922810-ee32-45c0-bf45-6cdba2c8ecc4|DONE|Wed Apr 30 19:08:13 UTC 2025|100|Wed Apr 30 19:08:15 UTC 2025|broadway Customer5.calculateIIDReport I|
|27a95a-e054-4087-985b-acc9cf05a184|null| | | | |
|a84ba697-clde-465b-8abf-c20dc2a926ab|DONE|Wed Apr 30 19:08:54 UTC 2025|100|Wed Apr 30 19:08:55 UTC 2025|broadway Customer5.calculateIIDReport I|
|27a95a-e054-4087-985b-acc9cf05a184|null| | | | |
|5a304c0f-8493-4d26-a829-01bd44bd1d58|DONE|Wed Apr 30 19:09:30 UTC 2025|100|Wed Apr 30 19:09:31 UTC 2025|broadway Customer5.calculateIIDReport I|
|27a95a-e054-4087-985b-acc9cf05a184|null| | | | |
```

*The second command returns the same results as `migrate_list STATUS='ALL';`

Batch Monitoring Commands

View Batch Command Details

batch_info '<batch_id>';

```
fabric>batch_info '7e1d395a-1470-4426-9b3b-ee9214e5f317';
|key                                     |value
+-----+-----+
|Batch command                         |batch Customer from CRM_DB using ('select customer_id fro
|LU Name                               |Customer
|Command Type                          |Sync
|Fabric Command                        |sync_instance Customer.?
|Max no. workers per node              |8
|Entity Inclusion                       |select customer_id from public.customer limit 200
|Source Interface                      |CRM_DB
|Generate Entities First               |false
|Affinity                              |
|Job Affinity                          |DC1
|Async                                 |true
|Max No. of Nodes                      |All
|Environment                           |_dev
|Allow Multiple Batch Executions       |false
|Execution id                          |913e266a-a446-407f-9ceb-ac594954a807

(15 rows)
```

Batch Monitoring Commands

View Batch Execution Summary

batch_summary '<batch_id>';

Provides:

- Execution Levels: Per Node / Per Data Center (DC) / Per Cluster
- Timing: Start and end time
- Total duration
- Progress Estimation: Remaining time and instances (available after IID generation)
- Statistics: Number of instances: synced/failed, added/updated/unchanged
- Pace Metrics: Sync rate per second (last BATCH_PACE_CALC_TIME_WINDOW_MS time window and average since the batch started)

```
fabric>batch_summary '35408af6-b26a-4243-bc95-f114335bfa5e';
```

Level	Name	Status	Start time	End time	Duration	Remaining dur.	Remaining	Total	Succeeded	Failed	Added	Updated	Unchanged	% Completed	Ent./sec (pace)	Ent./sec (avg.)
Node	13f54bc8-84de-49bc-aeb5-45cc98cb5854		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	37	0	37	0	0	18.5	29.16	29.16
Node	5b8182f9-4c3a-473a-8993-698a84c0047b		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	35	0	35	0	0	17.5	27.58	27.58
Node	8a449d3b-feb0-41c4-812a-8ead6c0a503f		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	36	0	36	0	0	18	28.37	28.37
Node	d0d60734-3ef1-4e1d-b163-9c83e1b93958		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	39	0	39	0	0	19.5	30.73	30.73
Node	e28619fd-9ea4-4d9f-ae91-3f1879d012fd		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	22	0	22	0	0	11	17.34	17.34
Node	f527581b-113a-4e0f-bfb6-37525e96d501		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	31	0	31	0	0	15.5	24.43	24.43
DC	DC1		2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0	--	200	0	200	0	0	100	157.6	157.6
Cluster	--	DONE	2020-08-12 12:20:07	2020-08-12 12:20:09	00:00:01	00:00:00	0		200	200	0	200	0	0	100	157.6

(8 rows)

Batch Monitoring Commands

View Currently Running Instances

batch_in_process FILTER='<filter regex> ';

Displays active instance processing across all running batch processes.

```
fabric>batch_in_process;
```

Node	Batch process id	Entity id	Lu type	Duration (ms)	Exeid	Task id	Command	Notes
dev-fabric-deployment-848b7d9f89-ddm97	1617cc02-1f24-42b5-9826-a1f33b044ff1	565	Customer5	93	b2e83	100161	sync_instance Customer5.?	
dev-fabric-deployment-848b7d9f89-ddm97	1617cc02-1f24-42b5-9826-a1f33b044ff1	564	Customer5	69	b2e83	100235	sync_instance Customer5.?	
dev-fabric-deployment-848b7d9f89-ddm97	1617cc02-1f24-42b5-9826-a1f33b044ff1	574	Customer5	26	b2e83	100376	sync_instance Customer5.?	
dev-fabric-deployment-848b7d9f89-ddm97	b06cbcf6-a9b5-4b7a-b523-150ea96a7df9	539	Customer	185	b9a48	99938	sync_instance Customer.?	
dev-fabric-deployment-848b7d9f89-ddm97	b06cbcf6-a9b5-4b7a-b523-150ea96a7df9	541	Customer	158	b9a48	100059	sync_instance Customer.?	
dev-fabric-deployment-848b7d9f89-ddm97	b06cbcf6-a9b5-4b7a-b523-150ea96a7df9	548	Customer	108	b9a48	100133	sync_instance Customer.?	
dev-fabric-deployment-848b7d9f89-ddm97	b06cbcf6-a9b5-4b7a-b523-150ea96a7df9	546	Customer	80	b9a48	100221	sync_instance Customer.?	
dev-fabric-deployment-848b7d9f89-ddm97	b06cbcf6-a9b5-4b7a-b523-150ea96a7df9	552	Customer	36	b9a48	100346	sync_instance Customer.?	

(8 rows)

Notes:

- Use the *process ID* to identify the batch.
- The number of records shown are limited by MAX_WORKERS_PER_NODE shared across all active batches.

Batch Monitoring Commands

View Instance-Level Sync Details

```
batch_details '<batch_id>'
[STATUS='<status>']
[ENTITIES='<entity1,entity2,...>']
[AFFINITY='<node_or_dc>']
[LIMIT=<limit>]
[SORT_BY_PROCESS_TIME=true|false];
```

Options:

- STATUS: WAITING, COMPLETED, FAILED
- ENTITIES: Comma-separated list of specific entity IDs
- AFFINITY: Filter by data centers or nodes
- LIMIT: Restrict result count (default limit is 10,000)
- SORT_BY_PROCESS_TIME: If set to true, displays only the 10 entities with the longest process times (overrides all other filters)

```
fabric>batch_details '1617cc02-1f24-42b5-9826-a1f33b044ff1' sort_by_process_time=true;
|Entity ID|Node id|Processed time (ms)|Status|Results|
+-----+-----+-----+-----+-----+
|218|dev-fabric-deployment-848b7d9f89-ddm97|225|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|221|dev-fabric-deployment-848b7d9f89-ddm97|219|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|649|dev-fabric-deployment-848b7d9f89-ddm97|216|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|222|dev-fabric-deployment-848b7d9f89-ddm97|216|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|509|dev-fabric-deployment-848b7d9f89-ddm97|211|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|505|dev-fabric-deployment-848b7d9f89-ddm97|197|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|510|dev-fabric-deployment-848b7d9f89-ddm97|193|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|660|dev-fabric-deployment-848b7d9f89-ddm97|192|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|730|dev-fabric-deployment-848b7d9f89-ddm97|188|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
|901|dev-fabric-deployment-848b7d9f89-ddm97|187|COMPLETED|{"Added":0,"Updated":1,"Unchanged":0}|
(10 rows)
```

```
fabric>batch_details '002943b4-1386-4d5f-860f-5a38a2a8e31f' limit 6;
|Entity ID|Node id|Status|Error|
+-----+-----+-----+-----+
|215|dev-fabric-deployment-848b7d9f89-ddm97|COMPLETED|
|{"Added":0,"Updated":1,"Unchanged":0}|
|216|dev-fabric-deployment-848b7d9f89-ddm97|FAILED|com.k2view.broadway.exception.StageException: Fl
not a function class org.openjdk.nashorn.internal.runtime.ECMAException Cause: TypeError: null is not a func
|
|217|dev-fabric-deployment-848b7d9f89-ddm97|COMPLETED|
|{"Added":0,"Updated":1,"Unchanged":0}|
|218|dev-fabric-deployment-848b7d9f89-ddm97|COMPLETED|
|{"Added":0,"Updated":1,"Unchanged":0}|
|219|dev-fabric-deployment-848b7d9f89-ddm97|COMPLETED|
|{"Added":0,"Updated":1,"Unchanged":0}|
```

Batch behind the scenes

Batch Flow

1. Batch Command Execution

- A new record is created in the k2batchprocess.batch_list table.
- Key fields include:
 - *bid*: Unique batch ID (UUID)
 - *FABRIC_COMMAND*, *JOB_UID*, and *session_scope* (containing the globals set on the session)
 - *Batch command metadata* (source DB, LU, interface, affinity, etc.)
 - *status*: NEW

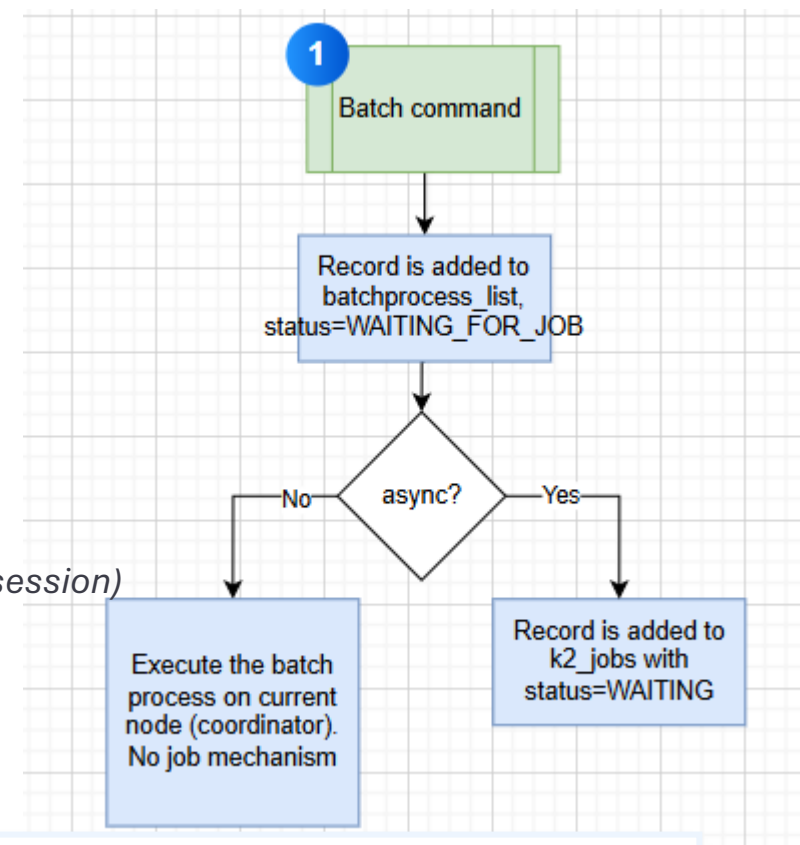
Async vs. Sync

Async Mode (async = true)

- A job record is created in k2_system.k2_jobs
 - *Type* = BATCH_JOB
 - *UID* = The batch command itself
- The job is picked up by one of the nodes, which becomes the **Coordinator Node**

Sync Mode (async = false)

- The batch runs directly on the current node, which acts as the Coordinator Node
- No job record is created
- Not backed by the job mechanism (no automatic retry or error recovery)

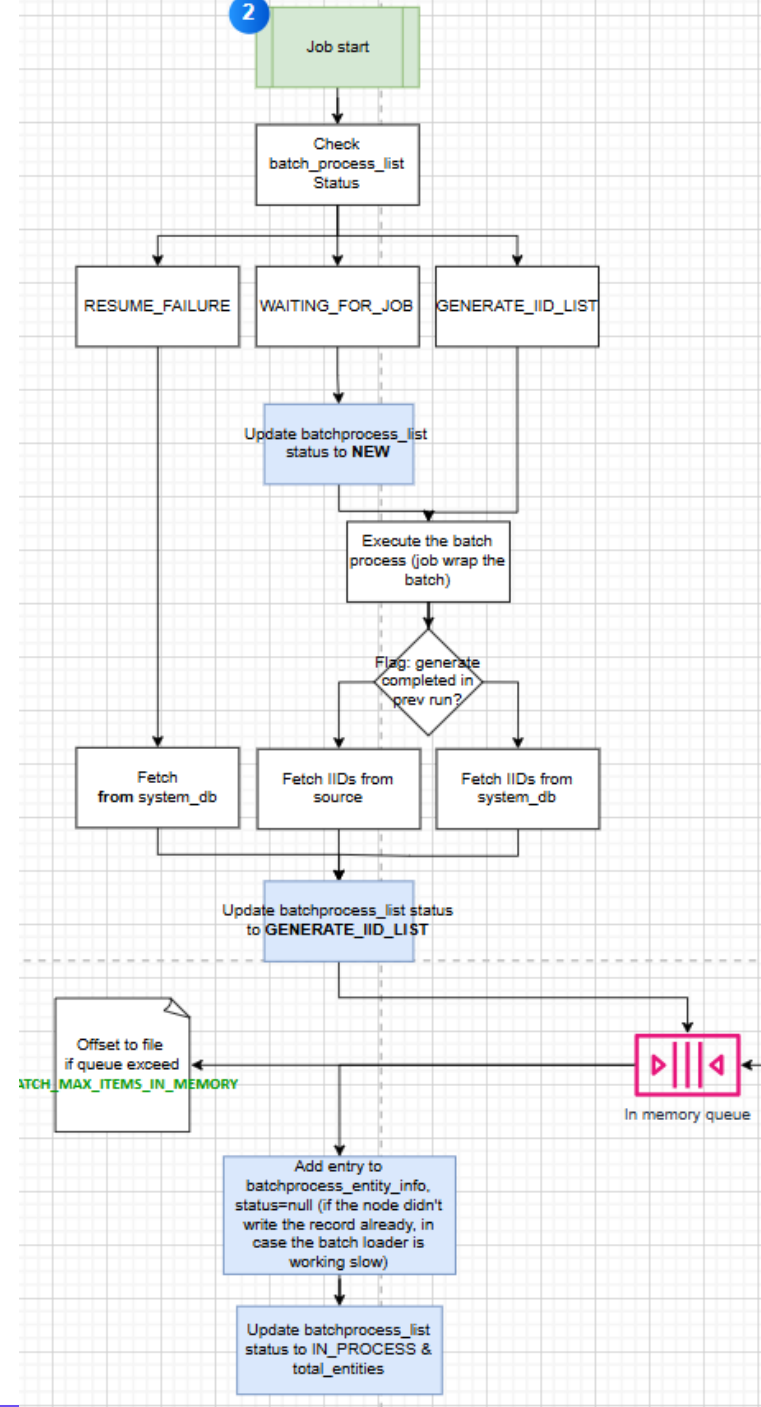


Batch behind the scenes

Batch Flow

2. Coordinator Flow - Generate IID list

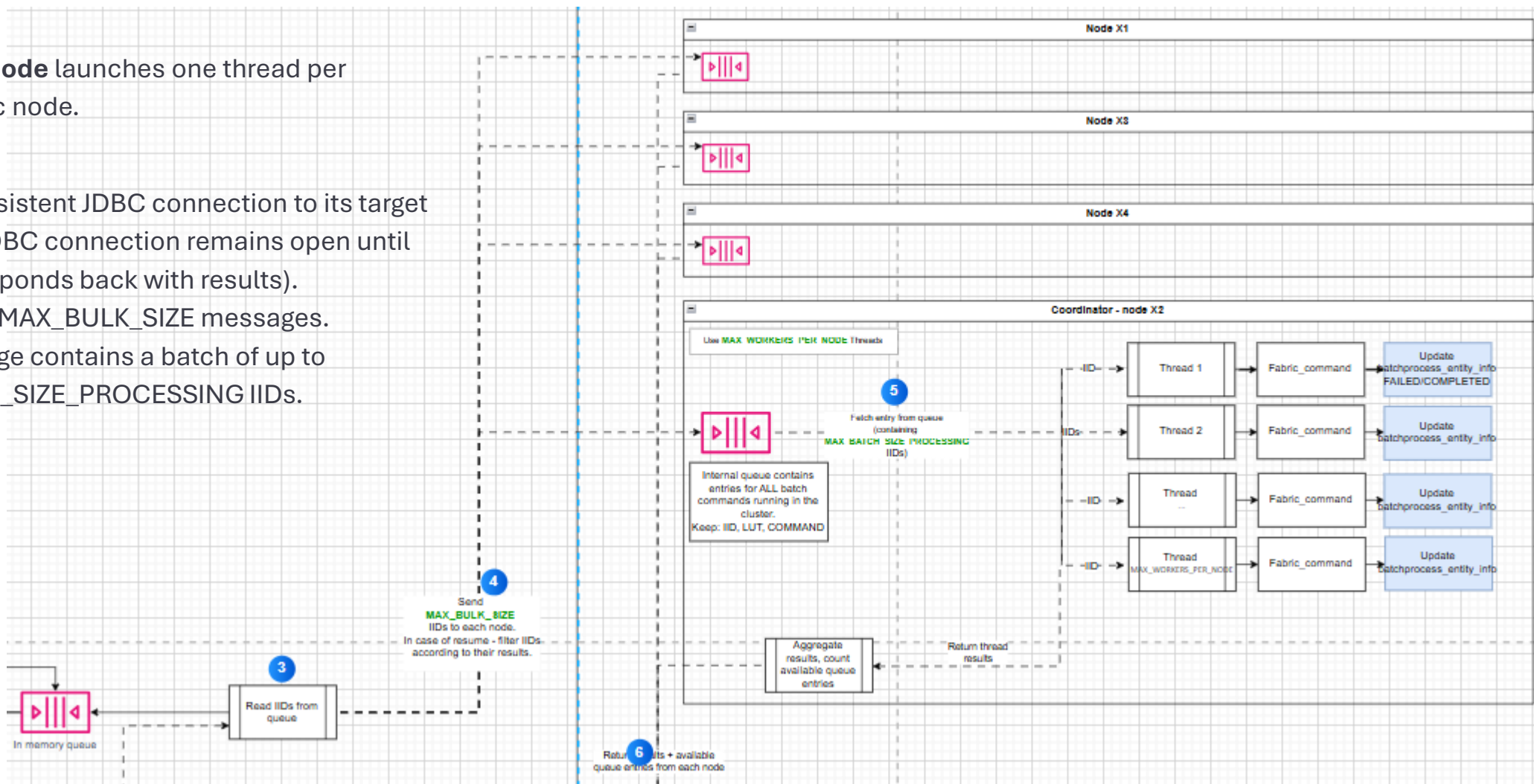
- The batch status in batch_list is updated to GENERATE_IID_LIST.
- IID Retrieval Begins, based on the IID_LIST command
 - If this is a batch_retry:**
 - Retry only failed IIDs → Exclude COMPLETED IIDs
 - Retry all unprocessed IIDs → Exclude both COMPLETED and FAILED IIDs
 - Note: Filtered IIDs are not re-sent to nodes, but their previous results are reused in the final batch statistics.
 - IIDs are pushed into an in-memory queue** (one queue per batch command):
 - The queue has a size limit to prevent memory overuse (BATCH_MAX_ITEMS_IN_MEMORY config.ini).
 - If the queue fills up, remaining IIDs are written to a file and then reloaded into the queue as space frees up.
 - For each IID**, a record is created in batch_entity_info with a null status (initial state before processing).



Batch behind the scenes

Batch Flow

- The **Coordinator Node** launches one thread per participating Fabric node.
- Each thread:
 - Opens a persistent JDBC connection to its target node (The JDBC connection remains open until the node responds back with results).
 - Sends up to MAX_BULK_SIZE messages.
 - Each message contains a batch of up to MAX_BATCH_SIZE_PROCESSING IIDs.

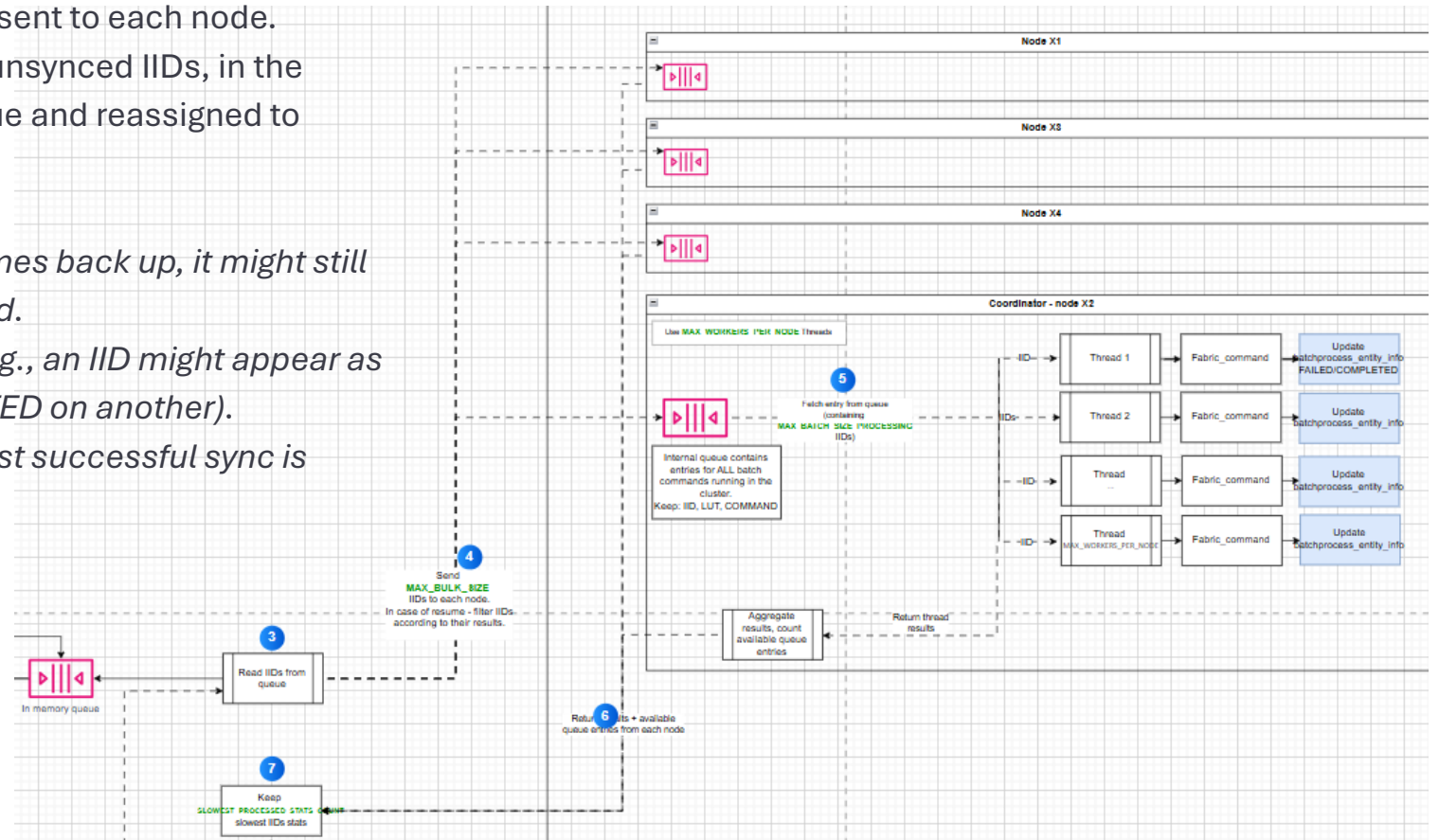


Batch behind the scenes

Batch Flow

• IID Tracking for Reliability

- The Coordinator keeps track of which IIDs are sent to each node.
- This ensures recovery if a node goes down — unsynced IIDs, in the node that went down, are returned to the queue and reassigned to other nodes.
- Note:
 - *If a node that previously went offline comes back up, it might still try to process IIDs it had already received.*
 - *This can lead to duplicate processing (e.g., an IID might appear as ADDED on one node and later as UPDATED on another).*
 - *However, for batch statistics, only the first successful sync is counted — duplicates are ignored.*

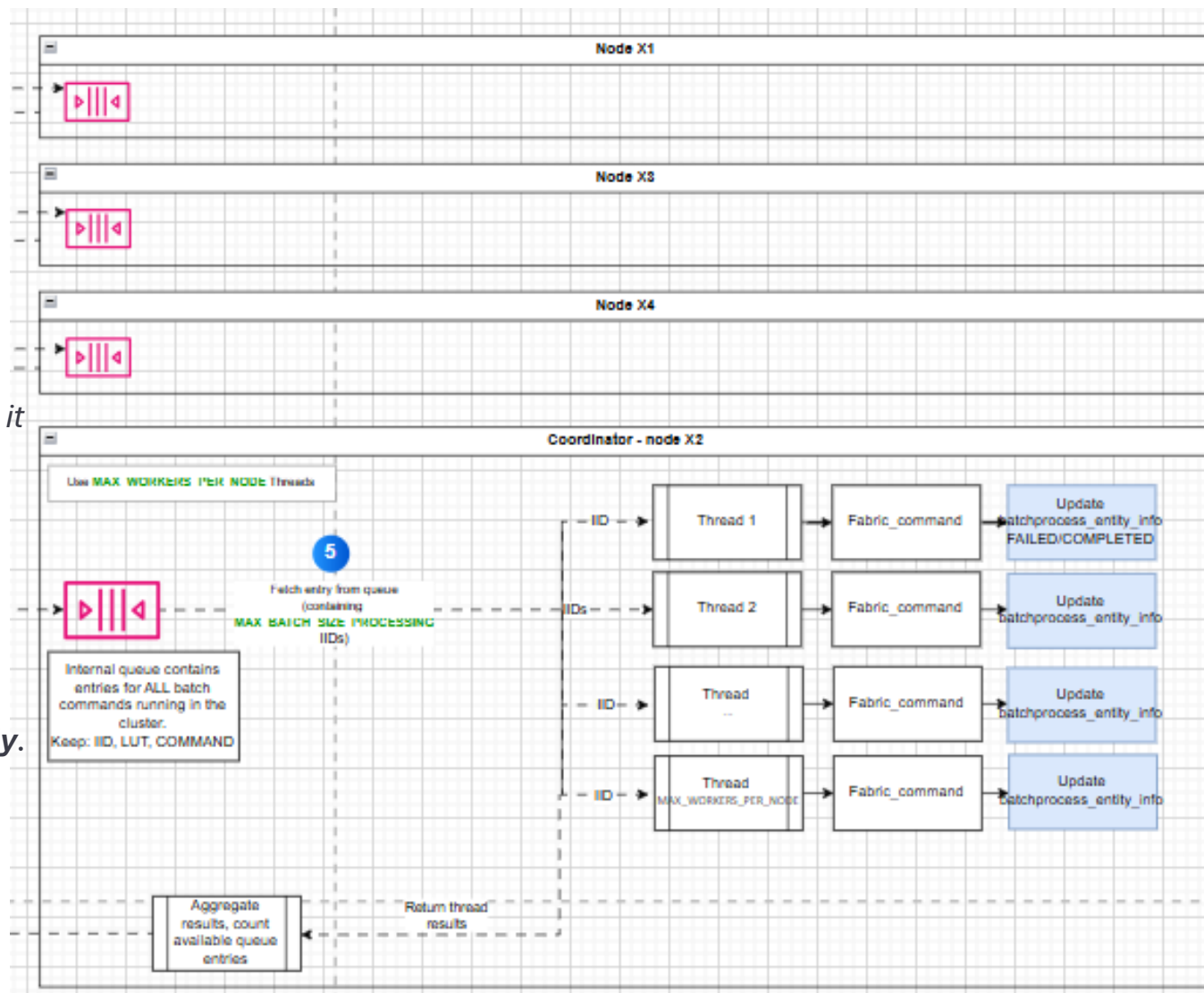


Batch behind the scenes

Batch Flow

3. Node-Side Processing

- a. Each node receives IIDs from the coordinator and places them into a **local queue**.
 - i. The local queue is shared across all active batch commands running on the cluster.
 - ii. Each queue entry contains more than just the IID — it includes metadata like:
 1. Logical Unit (LU)
 2. Fabric command
 3. Execution context
 - iii. Because the queue is shared, **threads on the node may serve multiple batch processes concurrently**.

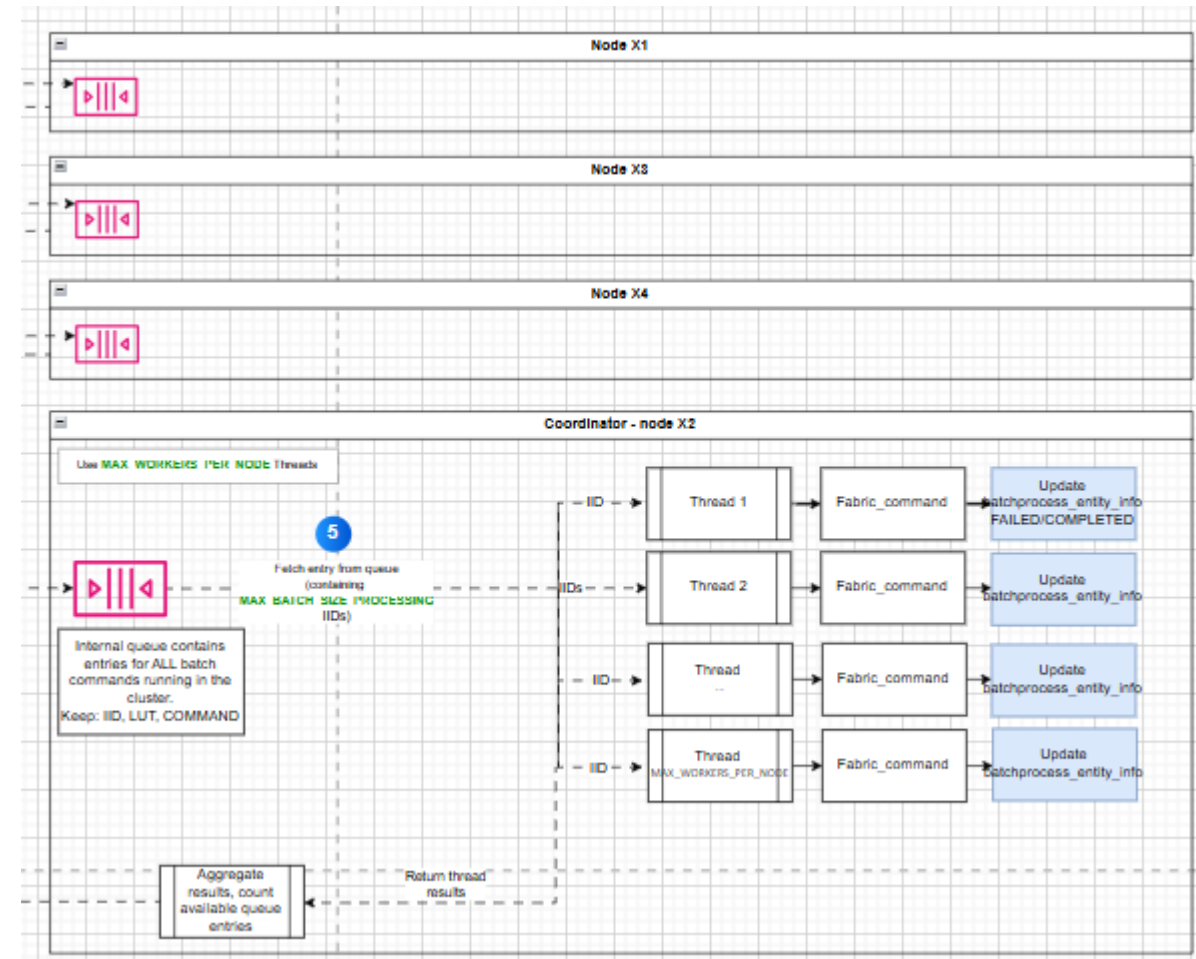


Batch behind the scenes

Batch Flow

b. Thread Management

- i. Each node runs `MAX_WORKERS_PER_NODE` threads.
- ii. Each thread:
 - 1. Pulls a message from the queue (default: up to `MAX_BATCH_SIZE_PROCESSING` IIDs).
 - 2. Processes the IIDs **sequentially**.
 - 3. Removes the entry from the queue after picking it up.
- iii. Once a thread completes its set of IIDs:
 - 1. It collects execution stats for each IID (start time, end time, status, errors, etc.).
 - 2. It updates the `batchprocess_entities_info` table, marking each IID as `COMPLETED`.



Batch behind the scenes

Batch Flow

c. Returning Results to the Coordinator

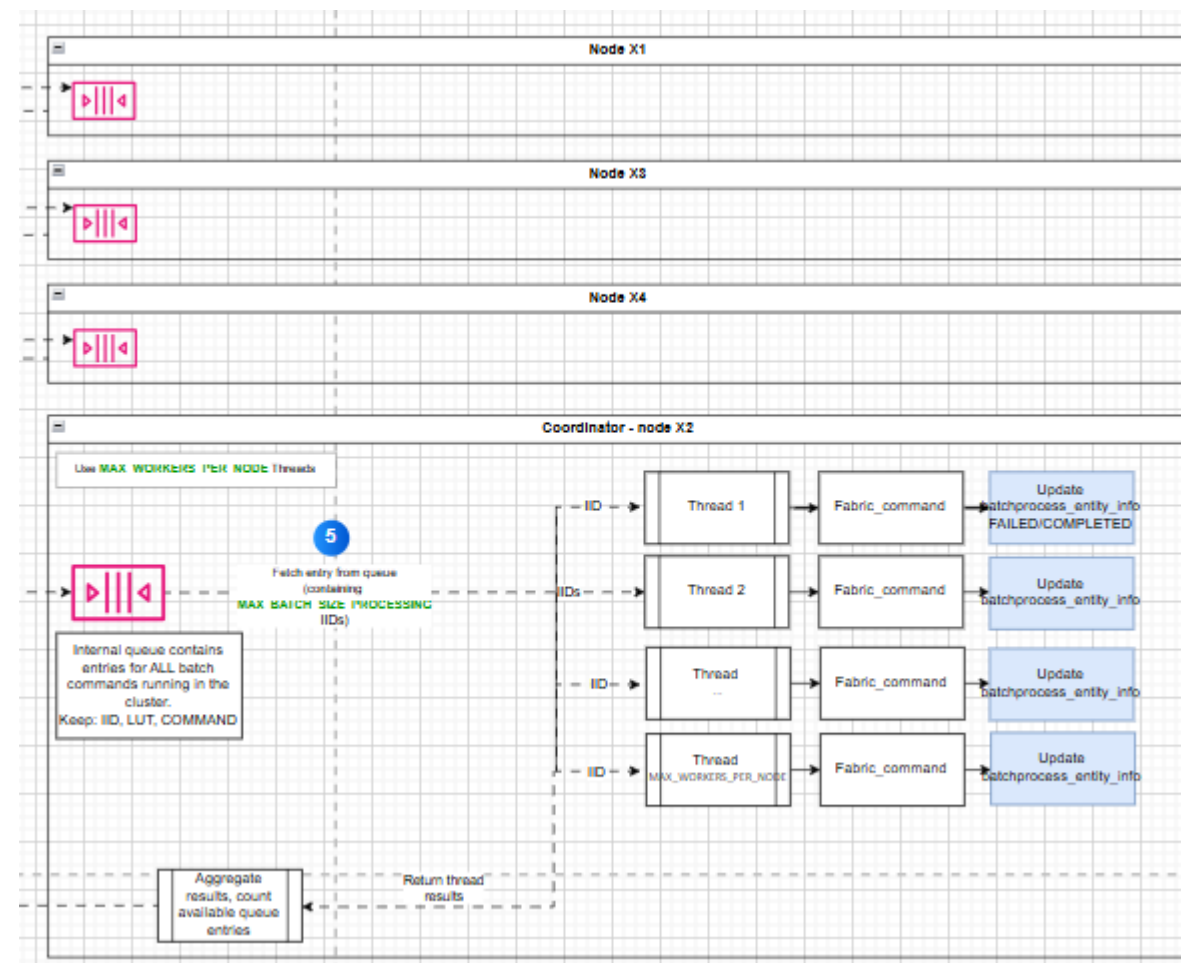
i. Once any thread on a node finishes processing its assigned IIDs, the **main thread** on that node sends a response back to the **Coordinator Node** over the open JDBC connection.

ii. The response includes:

1. **Available queue slots** on the node

2. **Aggregated execution results** from all threads:

- Batch ID (bid)
- IID
- LU
- Start time, end time
- Execution result (status, error, etc.)



Batch behind the scenes

Batch Flow

4. Coordinator Result Handling

a. Upon receiving results, the Coordinator:

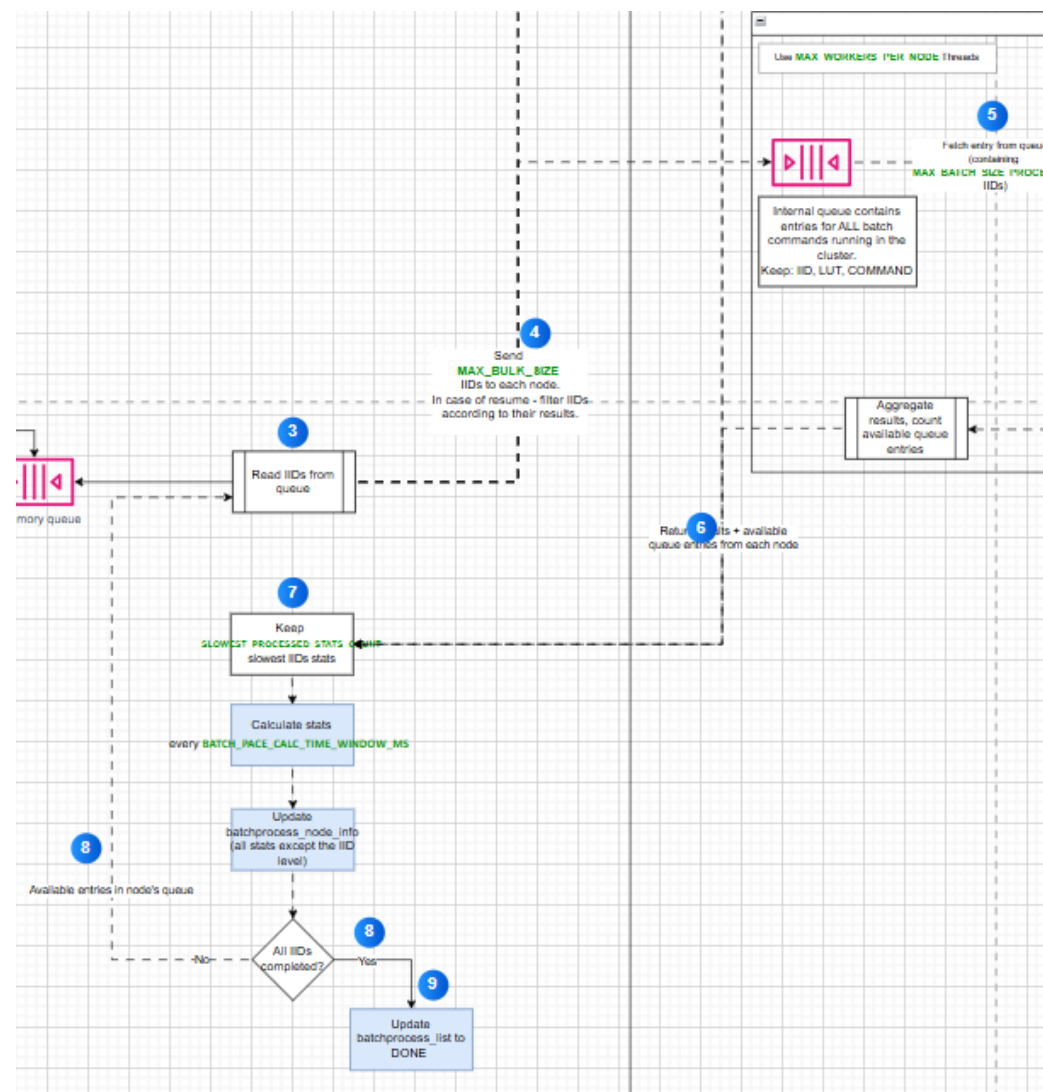
i. Maintains a Top-10 Slowest IIDs List

1. Compares the new results to the current slowest IIDs across all nodes and threads.

ii. Updates Batch Metrics

1. Calculates overall statistics
2. Updates the batchprocess_list and batchprocess_node_info tables with the latest execution data.

b. Once all IIDs processed, the batchprocess_list status is updated to DONE.





Batch behind the scenes

MAX_WORKERS_PER_NODE

When MAX_WORKERS_PER_NODE is specified in the batch command, no queue is created on the Fabric nodes and the batch will work slower:

- The Coordinator sends a fixed-size batch of MAX_BATCH_SIZE_PROCESSING IIDs.
- The node must complete all IIDs in that batch before sending results back.
- Only after sending the results back will the next batch of IIDs be sent



Batch behind the scenes

Batch Statuses

- **WAITING_FOR_JOB** - The batch command was executed, but the job has not started yet.
- **NEW** - The job has started, but the batch process has not begun execution.
- **GENERATE_IID_LIST** - The batch is generating the list of instance IDs (IIDs) to process.
- **IN_PROCESS** - All IIDs have been fetched, and sync operations are ongoing.
- **DONE** - The batch has completed successfully.
- **FAILED** - The batch command failed (not individual IIDs). See `batchprocess_list.error` for the failure message.
- **CANCELLED** - The batch was cancelled using the `batch_cancel` command.
- **PAUSED** - The batch was paused using the `batch_pause` command.
- **RESUME_FAILURES** - A `batch_retry` was issued after a previous execution completed. Only previously failed instances will be reprocessed.



Batch behind the scenes

Handling Dynamic Cluster Nodes During Batch Execution

Node Failure (Node Going Down)

- If a node goes down during a batch:
 - The **Coordinator** tracks which IIDs were sent and not yet acknowledged.
 - These unprocessed IIDs are returned to the coordinator's internal queue and redistributed to other active nodes.
- If the failed node was the **Coordinator**:
 - The batch job will be restarted on another node (that will become the new coordinator).

Node Recovery

- If a failed node comes back online, it may still attempt to process previously assigned IIDs.
- Duplicate processing may occur, but only the **first successful sync** is counted in the statistics.



Batch behind the scenes

Handling Dynamic Cluster Nodes During Batch Execution

Node Join (New Node Added)

During the migration, the Coordinator checks for newly added nodes (every `CHECK_FOR_ADDED_NODES_INTERVAL_MS` ms, default: 10 seconds).

When a new node joins, the Coordinator starts a **new thread** for it.

The node is then assigned IIDs and begins participating in the batch execution



Batch behind the scenes

Batch cancel/resume

When a batch_cancel/batch_pause command is issued:

- Fabric sends a **JDBC-based notification** to all participating nodes indicating that the batch is being cancelled.
- Nodes will:
 - Not process any IIDs that are still in the queue (related the cancelled/paused batch id).
 - **Allow currently running IIDs** to complete gracefully.
- The batch job updates the batchprocess_list table with status = CANCELLED/PAUSED.

```
fabric>batch_pause '14a8f7ed-c3dd-4a51-a26b-54cd4451efe8';  
(1 row affected)  
fabric>
```



Batch behind the scenes

Batch retry (resume)

When the `batch_retry` command is executed, Fabric determines how to resume the previous batch based on its last known status:

1. If the previous status was `DONE`, the new status is set to `RESUME_FAILURES`.
2. For all other statuses, it is set to `GENERATE_IID_LIST` (not `NEW`).

Job Startup Logic:

Upon job start, Fabric checks the current state to determine how to load the IID list:

- **If status = `DONE`:** IID list is loaded from `system_db`.
- **Otherwise:**
 - If the IID generation step was **not completed**, the system re-fetches the IID list from the source and inserts it into Cassandra.
 - If IID generation **was completed** but the batch did not finish, the system reuses the IID list stored in `system_db` from the previous run.



Batch behind the scenes

Batch retry (resume)

Distributing IIDs to Nodes:

- All IIDs are fetched from the system_db .
- The Coordinator:
 - Sends only IIDs that need to be synchronized.
 - Skips IIDs that were already successfully processed, but uses their previous results to update batch statistics.

Note:

When the IID_LIST is based on pub/sub, the Broadway flow that streams IIDs must clearly indicate when the IID list is complete (e.g., via not nextPage or an end-of-stream signal).

```
fabric>batch_retry '14a8f7ed-c3dd-4a51-a26b-54cd4451efe8';  
|Result|  
+-----+  
|Batch process resumed|  
  
(1 row)
```



Batch behind the scenes

Sync mode

When a batch is executed **without** `async=true`, it runs in **synchronous mode** directly on the current node, without using the job mechanism. As a result:

1. The **prompt remains blocked** until the batch completes.
2. **Pause and resume** options are **not supported**.
3. There is **no automatic recovery** if the Coordinator node fails—Fabric will not resume the process, unlike in `async` mode with job support.

A vertical photograph on the left side of the slide shows a person wearing a red jacket and dark pants standing on a rocky, snow-covered mountain peak. The person is looking out over a vast, snow-covered mountain range under a clear blue sky with some light clouds.

Batch behind the scenes

Batch_in_process

When the batch_in_process command is executed, the **Coordinator** sends a request to all participating nodes to **report the status of currently running IIDs**.

Session scope

Just like in the job mechanism, **session scope variables** are shared across all threads involved in the batch execution, and executed before running the bulk of IIDs (MAX_BATCH_SIZE_PROCESSING) on the node side.

- The session scope is stored in the arguments column of the batchprocess_list table.
- These variables ensure consistent context and global values across the entire batch process.



Batch behind the scenes

Common tables dependencies

If the LU Schema relies on a **Common Table**, Fabric performs a dependency check **before the batch starts**:

- The JDBC connection first requests the **remote node** to verify that the required common table has been executed.
- If a node does not have the required common table loaded, it will not synchronize any IIDs.
- The JDBC connection will withhold results until the common table has completed its sync.
- This prevents worker threads from getting stuck on IIDs that depend on missing common data.



Batch system_db tables

All batch-related tables are stored under the **k2batchprocess** schema.

These tables have a **Time-To-Live (TTL) of 7 days**.

TTL Handling:

- If system_db is **Cassandra**: TTL is enforced at the **table level** using Cassandra's native TTL configuration.
- For **other database types**: A **dedicated cleanup job** runs periodically to delete records older than 7 days.

Note: The 7-day retention period is hard-coded and cannot be modified by Fabric.

Batch system_db tables

batchprocess_list table

Stores metadata for each executed batch command.

< bid ▼	< arguments ▼	< command ▼	< creation_time ▼	< end_time ▼	< error ▼	< extra_stats ▼	< lut_name ▼	< owner ▼	< start_time ▼	< status ▼	< total_entities ▼
29757041-394d-481...	{"FABRIC_COMMAND":...	batch Customer fro...	2025-05-21 12:18:0...	2025-05-21 12:22:3...	null	{"slowestProcessed":...	Customer	shani@k2view.com....	2025-05-21 12:18:0...	DONE	1000

- **Bid** - Unique batch identifier (UUID)
- **arguments** - JSON containing:
 - Batch command parameters (e.g., FABRIC_COMMAND, SRC_DB_QUERY, lu_name, IS_ASYNC, etc.)
 - session_scope object with global variables and authenticated user details
- **command** - The full original batch command string
- **create_time, start_time, end_time** - Timestamps for when the batch was created, started, and completed
- **error** - Error details if the batch process itself failed (not per-entity errors)
- **extra_stats** - Tracks the **10 slowest processed**. Example: {"slowestProcessed": [{"entityId": "246", "processTimeMS": 10372, "status": "COMPLETED", "result": {}, "nodeId": "shani-mountain8-solutions"}, {"entityId": "236", "processTimeMS": 10373, "status": "COMPLETED", "result": {}, "nodeId": "shani-mountain8-solutions"},]}
- **lut_name** - Logical Unit name of the instances being processed
- **owner** - User who executed the batch (e.g., shani@k2view.com.k2v)
- **status** - Current state of the batch (e.g., NEW, IN_PROCESS, DONE, etc.)
- **total_entities** - Total number of IIDs processed (populated after generate_iid_list)

Batch system_db tables

batchprocess_node_info table

Stores a **per-node summary** of entities handled during a batch process.

< bid ▾	< nodeid ▾	< aggregated_results ▾	< dc_name ▾	< failed_co... ▾	< pace ▾	< succeeded_count ▾	< updated_time ▾
29757041-394d-4816-9951-c46c...	shani-mountain8-solutions	{"Added":210,"Updated":10,"Unchanged":0}	DC1	0	2.9608987198467296	220	2025-05-21 12:19:06.195

- **aggregated_results**

Total number of entities processed on the node, categorized by:

- added
- updated
- unchanged

- **pace** - Processing speed — number of entities handled in the last BATCH_PACE_CALC_TIME_WINDOW_MS ms (default 10 sec).
- **failed_entities_count** - Total number of entities that failed during processing on the node.
- **succeeded_entities_count** - Total number of successfully processed entities on the node.

Batch system_db tables

batchprocess_entities_info table

Stores detailed execution data for each individual entity (IID) processed within a batch command.

< bid ▾ ▴	< entityid ▾ ▴	< creation_t... ▾ ▴	< end_time ▾ ▴	< error ▾ ▴	< nodeid ▾ ▴	< results ▾ ▴	< start_time ▾ ▴	< status ▾ ▴
f4a0c1cc-6c22-4785...	863	2025-05-21 12:28:2...	null	null	null	null	null	null
f4a0c1cc-6c22-4785...	253	2025-05-21 12:28:2...	2025-05-21 12:29:0...		shani-mountain8-sol...	{}	2025-05-21 12:28:5...	COMPLETED
f4a0c1cc-6c22-4785...	323	2025-05-21 12:28:2...	2025-05-21 12:30:2...		shani-mountain8-sol...	{}	2025-05-21 12:30:1...	COMPLETED
f4a0c1cc-6c22-4785...	386	2025-05-21 12:28:2...	null	null	null	null	null	null
f4a0c1cc-6c22-4785...	529	2025-05-21 12:28:2...	null	null	null	null	null	null
f4a0c1cc-6c22-4785...	830	2025-05-21 12:28:2...	null	null	null	null	null	null
f4a0c1cc-6c22-4785...	244	2025-05-21 12:28:2...	2025-05-21 12:29:0...		shani-mountain8-sol...	{}	2025-05-21 12:28:5...	COMPLETED

Purpose:

Tracks the outcome and status of every entity processed in a batch, including:

- Execution time
- Status (e.g., COMPLETED, FAILED, null)
- Errors (if any)
- **Results** – for Broadway-based batches, this field captures the external output values returned by the flow execution.

Batch system_db tables

batchprocess_entities_errors tables

Contains **detailed error information** for entities that failed during batch execution.

< bid ▾ ▴	< entityid ▾ ▴	< error ▾ ▴	< nodeid ▾ ▴
51c19243-6c68-4bc9-b86c-bca005d08493	219	com.k2view.broadway.exception.StageException: Flow: BatchFlowSleepLog Le...	shani-mountain8-solutions
51c19243-6c68-4bc9-b86c-bca005d08493	221	com.k2view.broadway.exception.StageException: Flow: BatchFlowSleepLog Le...	shani-mountain8-solutions
51c19243-6c68-4bc9-b86c-bca005d08493	222	com.k2view.broadway.exception.StageException: Flow: BatchFlowSleepLog Le...	shani-mountain8-solutions

Purpose:

For each failed entity (IID), this table records:

- The **node ID** that attempted to execute the sync
- The **error message** returned by the sync process

Batch Actors

BatchWait

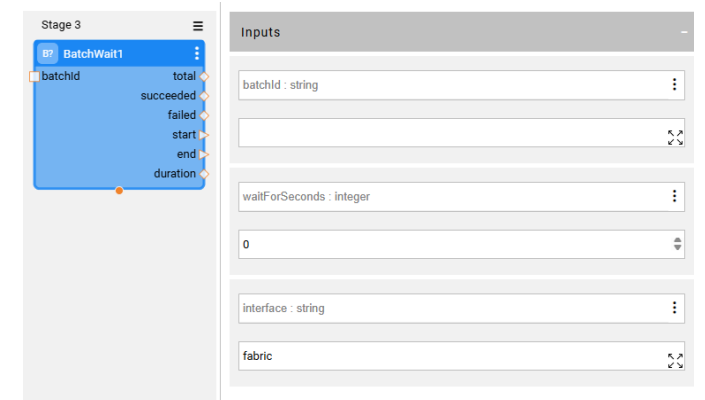
The **BatchWait** Actor waits for a batch process to complete.

Behavior:

- Completes when the batch finishes successfully.
- Throws an error if the batch fails or exceeds the **Wait For Seconds** timeout.
- If **Wait For Seconds** set to zero or a negative value, waits indefinitely.
- Interface it is set to fabric as default and can be changed to remote Fabric Interface
- If the batch is paused, the actor continues waiting. If the batch is cancelled, the actor will exit gracefully without error.

Output:

Batch stats.



The screenshot shows the configuration for the 'BatchWait1' actor in a workflow editor. The actor is located in 'Stage 3'. The configuration panel on the right lists the following inputs:

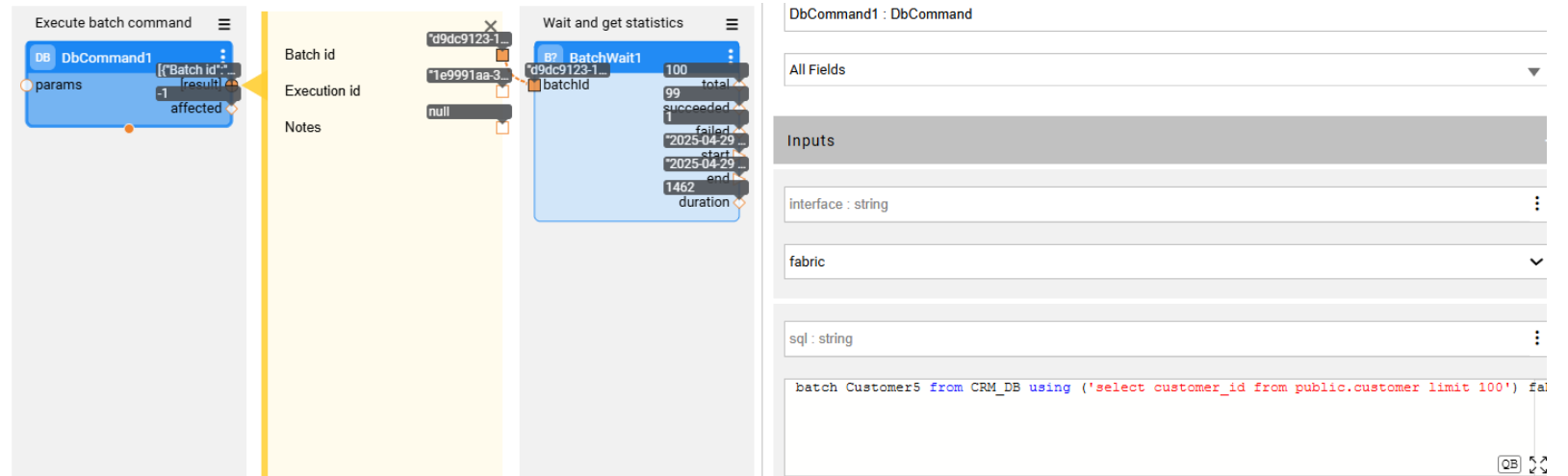
- batchid** : string (with a dropdown arrow)
- waitForSeconds** : integer (with a dropdown arrow, currently set to 0)
- interface** : string (with a dropdown arrow, currently set to 'fabric')

The actor's output is shown as a list of statistics: total, succeeded, failed, start, end, and duration.

Batch Actors

DbCommand

Use the DbCommand actor to execute batch command.



The screenshot displays a workflow editor with three main panels: 'Execute batch command', 'Wait and get statistics', and a detailed view of the 'DbCommand1' actor.

Execute batch command: Shows a 'DB DbCommand1' actor with a 'params' input. The output is a table with columns: Batch id, Execution id, Notes, result, affected, and duration.

Batch id	Execution id	Notes	result	affected	duration
"d9dc9123-1"	"1e9991aa-3"	null	1	1	1462

Wait and get statistics: Shows a 'BatchWait1' actor with a 'batchid' input. The output is a table with columns: total, succeeded, failed, start, end, and duration.

total	succeeded	failed	start	end	duration
100	99	1	"2025-04-29"	"2025-04-29"	1462

DbCommand1 : DbCommand: Shows the configuration for the 'DbCommand1' actor. The 'sql' field contains the following command:

```
batch Customer5 from CRM_DB using ('select customer_id from public.customer limit 100') fail
```




Batch Config

config.ini – batch_process Section

Primary Configuration Parameters:

- **MAX_WORKERS_PER_NODE** (*default: 8*)
Number of threads allocated per node for running all batch processes.
**Replaces the deprecated JOB_SERVERS_WORKERS_COUNT parameter.*
- **MAX_BATCH_SIZE_PROCESSING** (*default: 5*)
Maximum number of entities processed per message on the node side.



Batch Config

config.ini – batch_process Section

System DB Loader Configuration

Used when writing batch metadata to system_db (IIDs to batchprocess_entities_info). Fabric automatically detects the database type and applies the appropriate loader:

Defaults by DB Type:

- **For JDBC-based DBs:** jdbc_default_loader
- **For Cassandra:** default_loader

You can override the settings by adding **batch_process_loader** Section.

It is recommended to create the **batch_process_loader**, to not override other loaders functionality, with the below parameters

MODE = TOKEN_AWARE_BATCH (for Cassandra)/BATCH (for jdbc)

BATCH_SIZE = 100 (should be tuned based on the system_db type – may lead to a huge performance improvement)



Batch Config

config.ini – batch_process Section

[jdbc_default_loader]

#MODE = BATCH ; Execution mode: SINGLE or BATCH

#QUEUE_SIZE = 10000

#BATCH_SIZE = 1000 ; Applies only to BATCH mode

[default_loader]

#MODE = SINGLE ; Options: SINGLE / BATCH / **TOKEN_AWARE_BATCH**

#QUEUE_SIZE = 10000

#NUMBER_OF_THREADS = 1

#SESSION_NAME = loader

#MAX_IN_FLIGHT = 1024

#CONSISTENCY_LEVEL = LOCAL_QUORUM

#IS_NOP = false



Batch Config

config.ini – batch_process Section

Hidden / Advanced Parameters

- **BATCH_DETAIL_MAX_ROWS_SIZE** (*default: 10,000*)
Maximum rows fetched when executing batch_details. Required due to Cassandra consistency limitations.
- **BATCH_MAX_ITEMS_IN_MEMORY** (*default: 100,000*)
Max queue size held in memory. Overflow is stored in file on the disk.
Should be tuned if Instance IDs are large.
- **BATCH_PACE_CALC_TIME_WINDOW_MS** (*default: 10,000*)
Interval for Pace value calculation and system_db updates.
- **BATCH_UNITS_CACHE_EXPIRATION_IN_MIN** (*default: 1,440 minutes / 24 hours*)
Time to retain nodes IIDs results in case Coordinator is down.
- **CHECK_FOR_ADDED_NODES_INTERVAL_MS** (*default: 10,000*)
Frequency to check for new nodes joining the cluster.
- **COMMAND_JOB_UID_MAX_LENGTH** (*default: 1024*)
Max length allowed for the batch command string used as the job UID (limitations in keyspace for 1024 bytes)
- **SLOWEST_PROCESSED_STATS_COUNT** (*default: 10*)
Number of slowest processed IIDs to track in batch statistics.