**Batch process in D365 finance and operations**

Many tasks in finance and operations can be run as part of batch jobs. For example, batch jobs can include tasks for printing reports, performing maintenance, or sending electronic documents. By using batch jobs, you can avoid slowing down your computer or the server during typical working hours.

The tasks in a batch job can run either sequentially or at the same time. Additionally, you can create dependencies between tasks. In other words, the sequence of tasks can differ, depending on whether an earlier task succeeds or fails.

You can set up recurrence patterns for batch jobs. For example, you can set up a job to process invoices automatically at the end of every month.

To monitor batch jobs, you can set up alerts. Alerts can be sent when the batch job succeeds, fails, or finishes.

After a batch job is processed, you can view the history. The history includes any messages that were encountered while the job was running.

Use batch groups to categorize batch tasks and run them on specific servers. The servers in your environment might have different software installed, or they might be available at different times of the day. Batch groups are used to direct batch tasks to the most appropriate server. Tasks in the same batch job can belong to different batch groups.

For example, server A is set up to print reports, and server B is set up to send electronic documents. You can use batch groups to make sure that reporting tasks are run on server A and electronic documents are processed by server B.

**Batch functions**

Administrators and Batch managers can perform common tasks including creating and copying batch jobs, changing a batch job user, and specifying a time period in which a job shouldn't run.

 A batch job is a group of tasks that are submitted to an Application Object Server (AOS) instance for automatic processing. Batch jobs are run by using the security credentials of the user who created the job. Use the following procedure to create a batch job. The demo data company used to create this procedure is USMF.

**Create the batch job**

1. Go to System administration > Inquiries > Batch jobs.
2. Select New.
3. In the Job description field, enter a description of the batch job.
4. In the Scheduled start date/time field, enter the date and time when the batch job should run.
5. Select Save.

**Create a recurrence**

1. On the Action Pane, select Batch job.
2. Select Recurrence. Use these options to enter a range and pattern for the recurrence.
3. Select OK.

**Note**

All recurring batch jobs are automatically returned to the waiting state, regardless of whether they fail or succeed. This behaviour ensures that recurring jobs can complete any pending work during the next run if the previous run failed. This functionality can be enabled only if the batch job's recurrence conditions are still valid. For example, the batch job must have a remaining recurrence count or a recurrence end date that hasn't passed.

**Add alerts**

1. On the Action Pane, select Batch job.
2. Select Alerts. Indicate if you want alert messages sent when the batch job ends, has an error, or is cancelled. Then specify if you want the alerts to be displayed as pop-up messages.
3. Select OK.

**Add a task to a batch job**

1. On the Batch jobs page, select View tasks.
2. Select Ctrl+N to create a task.
3. Enter a description of the batch task.
4. In the Company accounts field, select the company database that the task should run in.
5. In the Class name field, select the process that the task should run.
6. As appropriate, select a batch group for the task.  
   Client tasks must be assigned to a batch group. They're automatically assigned to the default batch group (also known as the Empty batch group).
7. Select Ctrl+S to save the task.
8. To make the selected task dependent on another task in the job, select the Has conditions grid, and then follow these steps for each condition that you want to define:
9. Select Ctrl+N to create a condition.
10. Select the task ID of the parent task.
11. Select the status that the parent task must reach before the dependent task can run.
12. Select Ctrl+S to save the condition.  
    If you define more than one condition, and if *all* the conditions must be met before the dependent task can run, select a condition type of All. If the dependent task can run after *any* of the conditions is met, select a condition type of Any.
13. Select how task failures should be handled. To ignore the failure of a specific task, on the General tab, select the Ignore task failure option for that task. If this option is selected, failure of the task doesn't cause the job to fail. You can also use the Maximum retries field to specify the number of times that a task should be retried before it's considered to have failed. As a best practice, we recommend that you not set the Maximum retries field to a value that is more than 5.
14. Batch job history
15. Under the Batch Jobs in Save Jobs to History, you can select one of three options: Always, Errors Only, or Never.
    * Always – The history for the job is always created, irrespective of terminal status of the batch job.
    * Errors Only – The history of the job is only created if the job ended in the error state.
    * Never – No history is created for the batch job.
16. If the batch job has many batch tasks, we recommend that you set this field to Errors Only or Never.

**Important**

Starting with release 10.0.39, If the batch job has more than 5,000 batch tasks, then the corresponding job history would only save first 2,500 tasks, preferring tasks with status in following order: Error > Cancelled > Finished > Not Run. This measure has been implemented to prevent blocking batch-related tables that might occur due to such large jobs.

**Adjust batch job status**

1. Go to System administration > Inquiries > Batch jobs.
2. Select the appropriate batch job.
3. On the Action Pane, select Batch job > Functions > Change status.
4. Select the appropriate status:
   * Withhold – Set the batch job as withhold so it's withheld from the batch job scheduler. Equivalent to *stop*.
   * Waiting – Set the batch job as waiting so it's waiting to be picked up by the batch job scheduler. Equivalent to *go*.
5. Select OK.

In finance and operations apps, batch processing is used to run tasks asynchronously in the background. Batch jobs can range from simple tasks such as data import to complex calculations or integrations.

There are various situations where it might be necessary to update batch parameters and version them. Here are some examples:

* Configuration changes – Changes in the configuration settings or parameters might be required for batch processing. For example, a batch job that previously processed data in a specific way must now accommodate new fields or data sources. In these cases, you might have to update the batch parameters.
* Performance optimization – As your system evolves and grows, you might find opportunities to optimize batch processing for better performance. This optimization might involve adjusting batch parameters such as batch size or datasets to make the processing more efficient.
* Software updates or enhancements – If bugs are found in batch processing, or if new features affect batch jobs, you might have to change the batch parameters to fix issues or include new functions.
* Integration changes – Changes might be made in external systems or interfaces that interact with batch jobs, such as API endpoints or data formats. In these cases, you might have to update the batch parameters to accommodate the changes and ensure seamless integration.

Versioning of batch parameters is essential for maintaining a record of changes and ensuring consistency and reliability in batch processing. It lets you track the history of parameter changes, revert to previous versions if you must, and maintain documentation for auditing purposes. It also helps you manage and deploy changes across different environments, such as development, testing, and production environments.

Why you might receive errors during batch parameter unpack

When you run a batch, you might receive an error message like "An error occurred while unpacking parameters for batch job XXXXX." This error occurs when the batch job can't correctly unpack the parameters because of issues such as the following list:

* Parameter changes – When you update batch parameters, you might encounter errors during the pack/unpack process if the batch job isn't designed to handle different versions of the parameters list. The pack/unpack process is used to serialize and deserialize batch parameters for storage and execution. If the batch job expects a specific set of parameters but receives a different set because of version changes, errors can occur during execution.
* Custom code or extensions – If custom code or extensions are used with the batch job, errors in the code or extensions might cause issues when parameters are unpacked. Review the custom code or extensions for any errors or inconsistencies and address them accordingly.
* Security permissions – Insufficient permissions or security settings can sometimes prevent the batch job from accessing the necessary data or resources. As a result, errors can occur when parameters are unpacked. Ensure that the user who runs the batch job has the appropriate permissions to access all required resources.

**Best practices**

The following list describes recommendations for future changes to batch job parameters:

* Maintain versioned parameter lists. Ensure that both old and new versions of the parameter list are retained. Versioning allows for backward compatibility and helps smooth the transition between parameter versions.
* Adapt the unpacking process. Modify the unpacking process so that it handles both old and new versions of the parameter list. The unpacking mechanism can seamlessly identify and process parameters from either version.
* Consider the functional logic. The functional logic of the batch job should be designed to accommodate scenarios where an old version of the parameter list is provided. In these scenarios, the batch job should revert to the previous behaviour and adhere to the specifications that were defined before the parameter change.

Implementation of these recommendations enables the batch job system to effectively manage changes to parameters. It ensures compatibility and consistency in functionality across different versions of the parameters list.