Incremental push errors

Thursday, May 1, 2025 2:44 PM

Dynamics 365 F&O BYOD Incremental Push and Change Tracking

Overview: BYOD Incremental Exports and Change Tracking

Bring Your Own Database (BYOD) in Dynamics 365 Finance & Operations lets you export data entities from the application into your own SQL database for reporting or integration . You can run these exports in full push mode (re-copy all records each time) or incremental push mode (only send changes). Incremental push relies on SQL change tracking to detect which records have been inserted, updated, or deleted since the last export . Change tracking is a SQL Server/Azure SQL feature that records data changes (including deletes) on tables, and Dynamics 365 adds logic at the data entity level to use this information . In an incremental export, only records that have changed are exported, whereas if change tracking isn't enabled for an entity, you can only do full exports each time .

Important: The first time you run an incremental push for an entity, it will behave like a full push. The initial run must copy all records so that SQL change tracking can establish a baseline for that entity . After this initial full sync, subsequent runs will pick up only new or changed records. Because the first incremental run doubles as the full load, Microsoft recommends enabling change tracking before the first export and then running the export in incremental mode (which will automatically do the first full load) . There's no need to manually run a full push prior to enabling change tracking . On each incremental cycle, any new inserts or updates in Finance & Operations will be detected via change tracking and sent to the BYOD database, and any deletions in the source will be reflected by removing those records in the target (as long as the entity's root table deletion is tracked) .

How Change Tracking Identifies New or Updated Records

When change tracking is turned on for an entity, Dynamics 365 F&O enables SQL change tracking on the underlying table(s) that compose that data entity. Each time records are created or modified in the application, the change tracking mechanism logs the primary key of the affected record (and flags whether it was an insert or update). The BYOD incremental export job uses these logs to query "what has changed" since the last run, then exports only those records. Under the hood, the Data Management framework uses the change tracking version to get all changes (in a given company, if applicable) since the last exported version. It then performs the necessary operations in the external database – typically inserting new rows, updating existing ones, and processing deletions. Deletions are captured via change tracking as well (for BYOD scenarios), so if a record was deleted in the source, the incremental push will issue a delete for that record in the BYOD database table . This ensures the target reflects the source's current state without re-copying everything.

The granularity of what counts as a "change" is determined by the entity's change tracking configuration. In the Data Management workspace, after publishing the entity to BYOD, you choose one of three tracking options: Primary table, Entire entity, or Custom query .

- Primary table only: The system tracks changes only on the entity's primary (root) table. Any inserts or updates to that main table mark the entity record as changed. However, changes to related secondary tables in the entity will not trigger a change record. In other words, if only a child/related table field was modified, the entity would not be flagged as changed under this mode .

- Entire entity: Changes to any table that makes up the entity (primary or related tables) will mark the entity as changed . This means if, for example, a detail/child record or a joined table field within the entity is updated, the entity's record will be picked up in the incremental export. This option ensures broader change detection across all tables of the entity.
- Custom query: A developer-defined query specifies which tables/fields to monitor for changes. This is used for complex scenarios to fine-tune change tracking (for example, if only a subset of fields or a particular join condition should trigger an update) . The custom query must include the entity's root as the primary datasource, and you must enable change tracking on all tables involved in that query .

In summary, when properly configured, change tracking provides a reliable feed of new/updated records. The incremental push job will retrieve all changes since the last run and apply them to the BYOD database table. This approach is far more efficient than full reloads, and it maintains data consistency by also propagating deletions (with the caveat that only deletions of the entity's primary records are tracked – more on this below) .

Impact of Misconfigured Change Tracking

If change tracking is not configured correctly, the incremental push can fail to capture all changes or may behave unpredictably. Misconfiguration can include scenarios such as not enabling change tracking at all, enabling it on the wrong level (primary vs. entire entity) for your needs, or issues with the underlying SQL change tracking state.

- Change tracking not enabled: If an entity isn't change-tracked, you cannot truly run incremental exports – the system will default to full export each time because it has no way to detect incremental changes . In practice, the Data Management UI requires you to enable a tracking option before using incremental push. Failing to do so means every "incremental" job will actually copy everything (or nothing will happen, depending on the version). Always verify that the entity's change tracking is turned on in the Configure Entity export settings. If not, you will only get full-push behavior and might think incremental is "missing" updates, when in reality it never knew about them.
- Wrong tracking option (missing related changes): Using Primary table tracking when your data entity includes important fields from secondary tables can lead to missing updates. For example, say your entity joins a primary table with a child detail table. If you only track the primary table, an update to a child record (or insert of a new child) won't be flagged as a change. The incremental export will ignore that change, since from the primary table's perspective nothing changed . This misconfiguration results in out-of-date data in BYOD for those fields until some primary table change occurs. If you need to capture changes in related tables, ensure you select Entire entity tracking (or configure a custom query) so that these changes trigger an entity update .
- Missing unique key: A more structural misconfiguration is attempting to enable change tracking on an entity that lacks a unique key. Some out-of-box entities (especially ones meant primarily for imports) do not have a unique index/primary key defined. Change tracking can only be enabled on entities with a unique key, so if none is present, incremental push isn't supported for that entity . In such cases, the publish process will not allow enabling change tracking. The workaround is to extend the entity and add a suitable key. If this step is overlooked, you might think you've set up incremental export, but it either won't run or will fall back to full push due to no tracking data.
- SQL change tracking state issues: In some cases the change tracking feature itself can get out of sync or "stuck" for an entity. For example, Microsoft documents a scenario where after a full push, a subsequent incremental push resulted in only a few records appearing in BYOD – it looked as if the incremental job deleted all existing rows and only inserted the newly changed records . This obviously leads to missing data in the target. The root cause in such situations can be an inconsistent state in the SQL change tracking tables or the last sync version. The recommended solution is to disable and re-enable change tracking for that entity, essentially resetting the tracking state . Toggling change tracking forces a re-initialization on next run (which will be a full push again) and usually resolves the issue where changes weren't being picked up correctly. It's also advised to verify that no other processes are interfering – e.g. ensure no multiple incremental jobs or integrations are using the same underlying tables, as overlapping change tracking consumers could cause confusion .
- Out-of-range change tracking window: SQL change tracking uses a retention period to clean up old change

- Out-of-range change tracking window: SQL change tracking uses a retention period to clean up old change logs. If an incremental export job has not run for longer than the retention period, the system may no longer have a complete history of changes since the last sync. In such a case, Dynamics 365 will automatically fall back to a full push to avoid losing data . This is by design – it's better to re-send everything than to miss some changes that were pruned – but it can be surprising if you weren't expecting a full reload. If change tracking retention is misconfigured (too short for your schedule) or the job scheduling is not frequent enough, you might see unexpected full exports or missing data if the fallback doesn't occur in time. To avoid this, ensure your retention period covers the maximum interval between exports, or run the jobs more frequently. By default, the retention might be a few days (for example, 3 days) – if your incremental job pauses longer than that, consider increasing the retention setting on the SQL database or doing a manual full sync when restarting the schedule.

In summary, misconfigurations of change tracking often lead to either all-or-nothing outcomes (no incremental changes at all, or sudden full pushes) or partial data (some updates not captured). The key is to properly set up the entity's change tracking and monitor that it's functioning. If anomalies are observed (like missing records or an incremental acting like a full), reconfigure or reset change tracking and run a full sync to realign the data .

Capturing Changes in Related (Non-Primary) Tables

One common concern is whether changes in related tables (i.e. tables that are part of an entity but not the primary table) will be picked up by the incremental push. The answer depends on the change tracking configuration for the entity:

- Primary-table-only tracking: In this mode, only the primary (root) table's changes trigger an export . Fields coming from any secondary or joined tables are effectively static from the perspective of change tracking. If a field in a secondary table is updated, added, or deleted, the system will not recognize that as a change to the entity, so no update will flow to the BYOD database for that entity record . This can result in stale data for those fields. For example, if an entity combines a Sales Order header (primary) with some data from the Customer table (secondary) and you're only tracking the Sales Order table, a change to the Customer's name will not trigger an update in the exported data. The BYOD table's Customer name would remain old until the Sales Order record itself changed in some way.
- Entire-entity tracking: In this mode, any change in any table that comprises the entity will mark the entity as changed . This means the system enables change tracking on all those underlying tables and monitors them. If a secondary table record that contributes to the entity is modified, the entity's record will be included in the next incremental push (with all its fields) so that the BYOD data stays in sync . This is the safest option to ensure completeness when your entity spans multiple tables.
- Custom query tracking: This is a tailored approach where you specify exactly which tables (and even filter which records) to watch for changes. It's useful for complex multi-table entities where you might not want every related table change to trigger an export, but only certain ones. For example, you could track a subset of related tables that are critical. The custom query must join the primary table with those specific related tables to decide if an entity record has changed . With a proper custom query, changes in the tables included in the query will trigger an incremental update. Changes in tables not part of the query will be ignored.

When it comes to related table deletions, there is a limitation: change tracking will log delete operations only for the root table of the entity. In other words, if a record in a secondary table is removed, that fact is not directly treated as a "delete" event for the entity . Only if the primary record itself is deleted will the incremental push recognize a deletion and remove the corresponding row in BYOD . What happens if a related entity component is deleted? Typically, if a child record that contributed data is deleted, the entity's data might change (some fields might become blank or default). Ideally, this should count as an update to the entity. With Entire Entity tracking, any change in any table (including the removal of a related record) should mark the main entity as changed – however, because deletion tracking is only guaranteed for the root, it's possible that a child deletion might not be caught unless it somehow updates the primary record (e.g. via a cascading effect or if your custom query logic accounts for missing child records). This is a known design limitation: deletions are only tracked at the root level . As a result, you should be cautious if your scenario involves removal of related records. In some cases, a custom

a result, you should be cautious if your scenario involves removal of related records. In some cases, a custom change tracking query can be written to treat a missing child as a change (for instance, by left-joining and checking for nulls), but such solutions require custom X++ code on the entity.

Best practice: If your data entity pulls in data from secondary tables and it's important that any changes in those tables reflect in the export, use Enable entire entity tracking . Only limit to primary table tracking if you are sure that changes in related tables can be safely ignored between full refreshes. And remember that even with entire-entity tracking, if a related record is deleted, you may need to rely on periodic full pushes or additional logic if you need to capture that scenario. The vast majority of incremental changes (inserts/updates in related tables) will be handled with entire-entity tracking, but deletions are a special case to be mindful of.

Known Limitations and Issues (Missing Lines in SQL Export)

There are some known limitations and quirks in BYOD incremental pushes that can lead to missing records or other unexpected results:

- Entities without unique keys: As mentioned, you cannot enable change tracking on an entity that lacks a unique index. These entities cannot be incrementally exported . Any attempt will result in only full pushes. If you require incremental export for such data, you must modify the entity to include a unique key (for example, combination of fields) . Failing to do so means the entity might never export data (if you set it to incremental only) or will always do full replacements – which could be misconstrued as "missing" data if you expected incrementals.
- Composite entities not supported: BYOD does not support composite entities (entities that are made up of multiple standard entities in one export). You must export each component entity individually . If you tried to use a composite entity, you would get no data or errors. This is a limitation of the BYOD framework itself, and it means you should design exports around single entities and then combine them in the destination if needed.
- Change tracking retention and long gaps: As noted, if the time between incremental runs exceeds the SQL change tracking retention window, some changes might age out. The system is designed to detect this and automatically revert to a full push to prevent data loss . However, if for some reason this mechanism fails or is not triggered, you could end up missing intermediate changes. Typically, you will see a full reload happen (which in itself might be an unexpected heavy operation). It's important to schedule the incremental export at a frequency that avoids surpassing the retention period, or to adjust the retention period on your BYOD SQL database if you need longer intervals. This will ensure no changes are missed or purged.
- Concurrent exports and data loss: Running multiple BYOD exports on the same entity concurrently (especially across different companies or in parallel jobs) has been known to cause issues. High concurrency can lead to SQL throughput issues (DTU/CPU spikes) and even data loss for incremental exports due to overlapping change tracking usage . Microsoft addressed some of this in version 10.0.16 by forcing cross-company exports for the same entity to run sequentially (per company) rather than in parallel . Still, if you manually set up multiple export projects that include the same entity, or if you run exports in rapid succession, you risk one job interfering with the other. This could manifest as missed records (one job might reset the tracking watermark while another is still processing). The guideline is to never have the same data entity in more than one active export project at the same time . Let them run one after the other, not simultaneously.
- Partial data due to change tracking glitches: The scenario described earlier – where an incremental push resulted in only the changed rows being present (and others seemingly deleted) – is a known issue that can occur if the change tracking metadata is out-of-sync. Essentially, the incremental job believed that any record not reported by change tracking had been deleted, thus it removed them from the target, even though in reality they just hadn't changed . This is not normal behavior, but it has been observed. If you see a dramatic drop in record count after an incremental run (when there weren't mass deletions in the source), it's likely this issue. The fix is to reset change tracking on that entity (turn it off and on) and then do a full push to restore missing records . Also double-check that no other processes (e.g. a second export job, or perhaps Management Reporter or Retail jobs that use change tracking on the same tables) are interfering .
- Deleting data in target vs. source: Incremental push will delete records in the target BYOD database only if those records were deleted in the source and the entity is tracking deletes (which, as noted, only applies to

entity that lacks a unique index. These entities cannot be incrementally exported . Any attempt will result in only full pushes. If you require incremental export for such data, you must modify the entity to include a unique key (for example, combination of fields) . Failing to do so means the entity might never export data (if you set it to incremental only) or will always do full replacements – which could be misconstrued as "missing" data if you expected incrementals.

- Composite entities not supported: BYOD does not support composite entities (entities that are made up of multiple standard entities in one export). You must export each component entity individually . If you tried to use a composite entity, you would get no data or errors. This is a limitation of the BYOD framework itself, and it means you should design exports around single entities and then combine them in the destination if needed.
- Change tracking retention and long gaps: As noted, if the time between incremental runs exceeds the SQL change tracking retention window, some changes might age out. The system is designed to detect this and automatically revert to a full push to prevent data loss . However, if for some reason this mechanism fails or is not triggered, you could end up missing intermediate changes. Typically, you will see a full reload happen (which in itself might be an unexpected heavy operation). It's important to schedule the incremental export at a frequency that avoids surpassing the retention period, or to adjust the retention period on your BYOD SQL database if you need longer intervals. This will ensure no changes are missed or purged.
- Concurrent exports and data loss: Running multiple BYOD exports on the same entity concurrently (especially across different companies or in parallel jobs) has been known to cause issues. High concurrency can lead to SQL throughput issues (DTU/CPU spikes) and even data loss for incremental exports due to overlapping change tracking usage . Microsoft addressed some of this in version 10.0.16 by forcing cross-company exports for the same entity to run sequentially (per company) rather than in parallel . Still, if you manually set up multiple export projects that include the same entity, or if you run exports in rapid succession, you risk one job interfering with the other. This could manifest as missed records (one job might reset the tracking watermark while another is still processing). The guideline is to never have the same data entity in more than one active export project at the same time . Let them run one after the other, not simultaneously.
- Partial data due to change tracking glitches: The scenario described earlier – where an incremental push resulted in only the changed rows being present (and others seemingly deleted) – is a known issue that can occur if the change tracking metadata is out-of-sync. Essentially, the incremental job believed that any record not reported by change tracking had been deleted, thus it removed them from the target, even though in reality they just hadn't changed . This is not normal behavior, but it has been observed. If you see a dramatic drop in record count after an incremental run (when there weren't mass deletions in the source), it's likely this issue. The fix is to reset change tracking on that entity (turn it off and on) and then do a full push to restore missing records . Also double-check that no other processes (e.g. a second export job, or perhaps Management Reporter or Retail jobs that use change tracking on the same tables) are interfering .
- Deleting data in target vs. source: Incremental push will delete records in the target BYOD database only if those records were deleted in the source and the entity is tracking deletes (which, as noted, only applies to root table deletions) . If you manually delete or truncate data in the BYOD database, the system doesn't automatically refill those on an incremental run unless it thinks they were new changes. Essentially, the BYOD process isn't bi-directional sync; it's one-way. So manual target-side changes can cause mismatches. A full push is needed to correct any manual deletions in BYOD. Similarly, if a source record was accidentally not exported (due to a previous glitch) and remains missing in BYOD, that record won't appear until it changes in the source (triggering change tracking) or you do a full push.

Recognize that incremental exports transfer only what's new/changed; they do not continuously verify every record's existence. That's why ensuring the incremental process doesn't miss changes is critical.

- Timeouts and large volumes: While not a data correctness issue per se, very large data volumes can lead to export timeouts (default is 10 min for deletion phase, 1 hour for bulk copy) . If a job times out, it might not complete the data copy, effectively leaving the target incomplete. The fix is to increase the timeout settings (in Data management > Framework parameters > Bring your own database) if you have huge entities . If an export regularly times out and is not noticed, it could give the impression that data is missing, when in fact the job failed each time before finishing. Monitoring job success is thus important for completeness.

Best Practices for Ensuring Data Completeness in Incremental Exports

To maximize the reliability and completeness of BYOD incremental pushes, consider the following best practices:

- Enable change tracking correctly: Always turn on change tracking for each entity you plan to export incrementally, and choose the appropriate scope. Use Entire entity tracking if you need to capture changes in related tables or any part of the entity . Only use Primary table tracking for simple entities or when you explicitly don't care about secondary table changes. If a custom tracking query is used, ensure it covers all necessary tables/fields. Double-check that the SQL change tracking is indeed enabled on the database and tables after publishing (the system usually does this for you).
- Ensure a unique key on the entity: Verify that the data entity has a unique key and that it's properly set as the entity's primary key in the schema. If not, add one. Without a unique identifier, you cannot enable change tracking , and any attempt at incremental load will not function. This is often an issue when using or extending standard entities that were not intended for export – you may need to add an index (via extension) to use BYOD incremental export.
- Run incremental jobs within the retention window: Schedule your incremental exports to run frequently enough so that no change tracking data is lost. For example, if the retention period is 3 days, make sure the job runs at least every 1-2 days (or adjust the retention setting in your SQL database to a longer period if needed). This prevents the scenario where too much time passes and the system has to do a full push (or worse, misses deletes) . Regular, smaller incremental runs are generally better for data freshness and reduce the load of occasional huge syncs.
- Monitor export job results: Keep an eye on the Data Management job history. Ensure incremental jobs are succeeding and check the record counts. If you notice an abnormal drop in the number of records exported or any error messages, investigate immediately. An unexpected small number of exported records (when there were no major deletions in F&O) could indicate the change tracking issue discussed above . If needed, re-publish the entity or disable/re-enable change tracking to reset things, then run a full push to recover the data. It's wise to periodically compare the record count in BYOD vs the source system for key entities to ensure they match (at least within the expected range minus any legitimate deletes).
- Avoid concurrent and overlapping exports on the same data: Design your batch schedule such that the same entity isn't being exported in two different projects or for two different companies at the same time . In versions prior to 10.0.16, cross-company exports in parallel could corrupt incremental tracking; even with fixes, it's best to run exports sequentially per entity . If you need to export the same entity for multiple companies, use the "Export all companies" option (which is now handled