TM1 Technical Bulletin

BatchUpdateFinishWait TurboIntegrator Function

Date: February 12, 2007

Relevant TM1 Versions: 9.1

This technical bulletin describes a new TurboIntegrator function introduced in TM1 9.1.

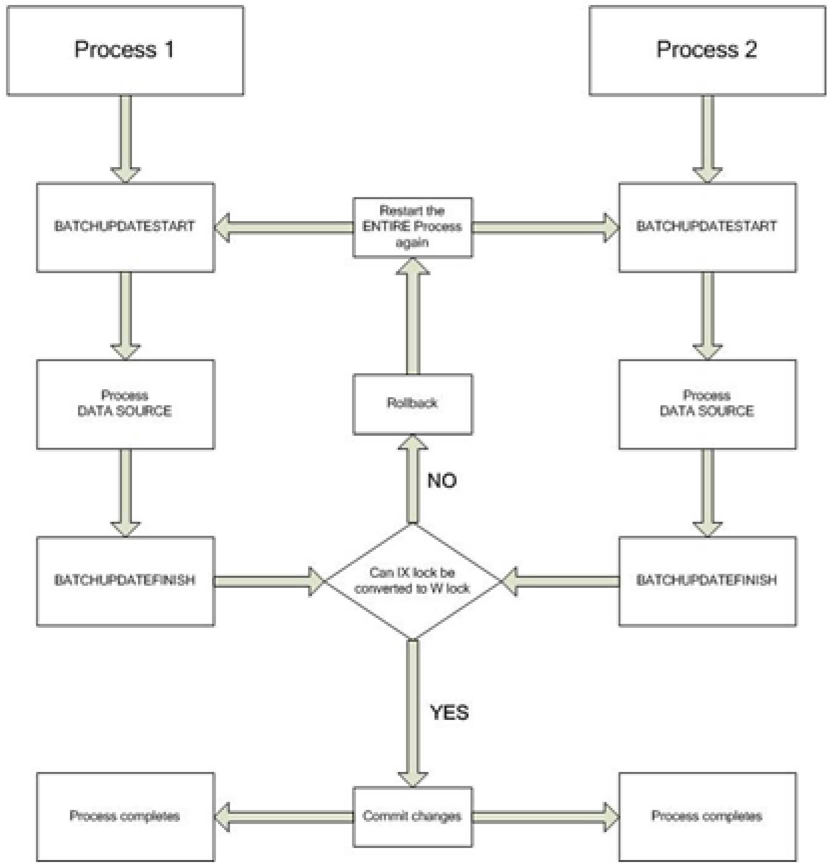
Background

In TM1 9.1, a new locking scheme has been introduced that provides greater concurrency and overall stability. Under the new locking scheme, the behavior of the BatchUpdateFinish TurboIntegrator function has been slightly modified in instances when multiple simultaneous processes are running in batch update mode and applying changes to a single cube.

This change can be illustrated using an example of two processes, Process 1 and Process 2, that update a single cube.

- Both processes start and call the BatchUpdateStart function to initiate batch updates.
- Each process operates on a unique data source.
- Process 1 completes processing data and calls the BatchUpdateFinish function. The process obtains a write lock to the cube and commits changes.
- While Process 1 still holds a write lock to the cube, Process 2 completes processing data and calls the BatchUpdateFinish function. However,

because Process 1 retains the lock, Process 2 cannot obtain a lock to the cube. All data changes applied in Process 2 are rolled back and Process 2 is restarted. This ensures data integriry.



Depending on the size of the datasource for Process 2, the data rollback and process re-execution can cause a noticeable decrease in performance. To address this performance issue, TM1 9.1 includes a new TurboIntegrator function, BatchUpdateFinishWait.

BatchUpdateFinishWait

This TurboIntegrator function is identical to the BatchUpdateFinish function with the following exception:

If a process calls BatchUpdateFinishWait, but is unable to secure a cube write lock to commit changes, the process will wait until the lock becomes available and then commit changes. Data changes applied in the process are not rolled back and the process is not re-executed.

IMPORTANT: While waiting for the cube write lock, the process releases any read locks it acquired for other objects during process execution. Because these read locks are released before the process can commit changes to the cube, the objects for which the read locks are released can be modified *before* the cube is updated. This can lead to data inconsistency when using BatchUpdateFinishWait.

Applix recommends that BatchUpdateFinishWait be used only in controlled situations where you know that other processes are not modifying data or metadata related to the process that calls BatchUpdateFinishWait.

Syntax

```
BatchUpdateFinishWait(SaveChanges);
```

Arguments

SaveChanges	A flag that instructs the server to either save or discard changes committed while in batch update mode. Specify 0 to save changes, 1 to discard changes.

Example

```
BatchUpdateFinishWait(0);
```

This example instructs the TM1 server to save changes to TM1 data and exit batch update mode.