Tuning Content Platform Engine for High Volume Workloads

For customers migration from Content Manager OnDemand to Content Cortex, it is important to tune the Content Cortex object store databases to support high volume report ingestion. When working with reports, the Sections table in the object store database can grow very quickly. For instance, after importing a large number of reports or importing reports with a large number of sections, the Sections database table can exceed one billion rows. When the Content Engine processes reports, it commits all sections within a single report as one transaction. This can cause issues if the backend database is not optimized for high-volume workloads. For example, importing a report with two million sections might cause a transaction to fail due to exceeding the default timeout or because the database transaction logs are full.

To prepare the system for large workloads, implement the following tuning recommendations:

- Adjust the transaction timeout in the application servers hosting the Content Platform Engine.
- Increase the available number of database transaction logs.
- Pre-allocate space in the database tablespaces.
- Create custom database indexes to optimize search performance and retrieval rates.
- Adjust the batch insert size to optimize report ingestion.

Each of these recommendations is described in more detail in the following sections.

Increase the Transaction Timeout

In a Content Cortex configuration that uses an Oracle 21c database and a WebSphere application server without any tuning, the following transaction timeout errors occur when processing reports that exceeded 700,000 sections:

> FNRCD0009E: DB_ERROR: The database access failed with the following error: ErrorCode 17,008, Message 'Closed Connection' ObjectStore: "OS" failedBatchItem=0

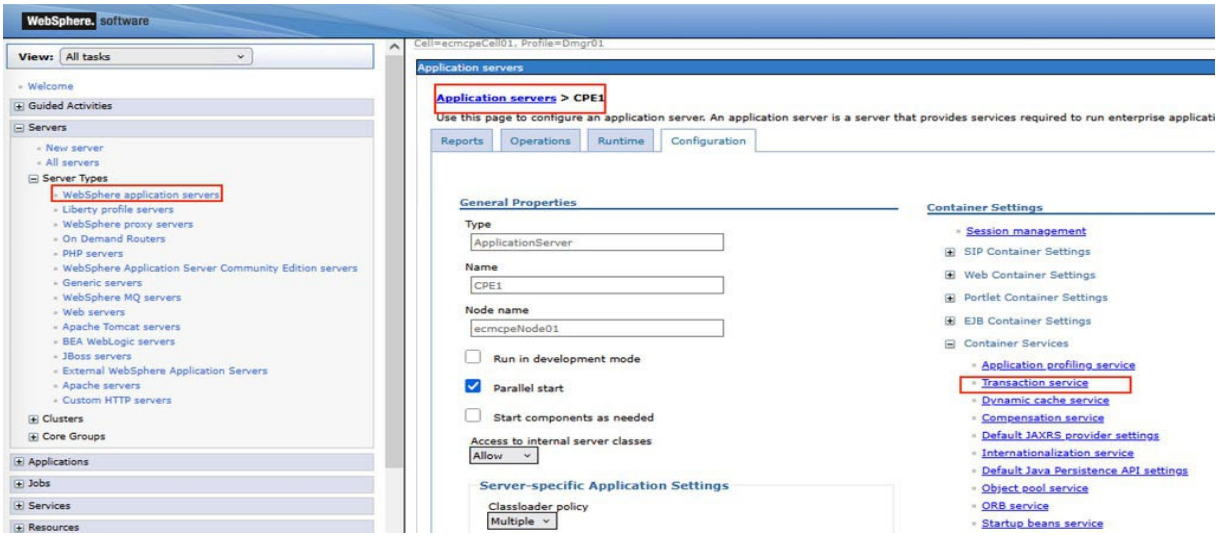
This same type of issue can also occur with other databases and application servers.

To resolve the issue, make both of the following updates:

- Update the Transactions timeout duration to 360 seconds on all Content Platform Engine servers. On WebSphere, by default, the timeout is set to 120 seconds and is set in the following location:

  Servers > Server Types > WebSphere application servers >CPE> Container Services > Transaction Service.

  Note that the best timeout value for your environment is affected by factors such as database disk performance, network speed, and system configuration.



- Set the following custom property on all data source definitions, ensure that individual SQL queries do not time out before the overall transaction is complete:

  SyncQueryTimeoutWithTransactionTimeout = true

Increase the Number of Database Transaction Logs

The following error can occur even if the database has been optimized for large workloads:

SQL0964-The transaction log for the database is full

In Db2 when this error occurs it means the single uncommitted transaction requested more log space than the combined capacity of the primary and secondary log file.

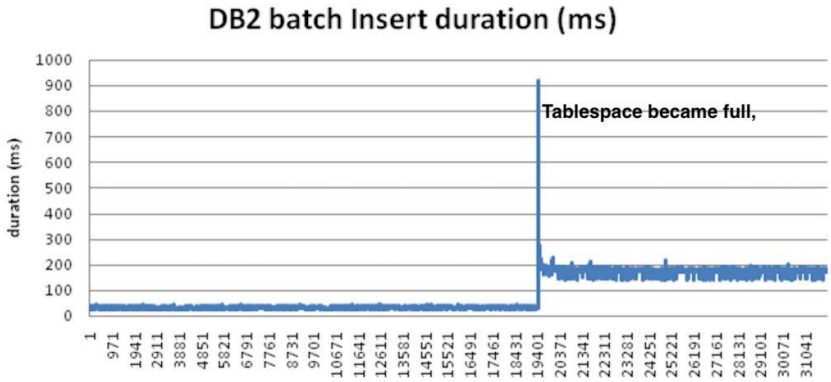
Note that sometimes the logs are structurally large enough to hold the required data, but the underlying disk drive or file system hosting the log files is out of space.

The following commands illustrate updating Db2 log allocations:

```
db2 update db cfg for <dbname> using LOGFILSIZ 20000
db2 update db cfg for <dbname> using LOGPRIMARY 50
db2 update db cfg for <dbname> using LOGSECOND 100
```

Pre-Allocate Space in the Database Tablespaces

As databases reach the pre-allocated capacity, ingestion performance can decline significantly as the database performs on-demand expansion. For example, as the Db2 section tables approached 500 million rows, report ingestion rates declined significantly due to a substantial increase in the DB INSERT processing time. The following chart shows the increase in time for a batch insert that can occur when the tablespace becomes full.



Pre-allocating space in the database tablespaces prevents transaction failures, eliminates the latency caused by the on-demand file growth, and avoids disk fragmentation.

While highly effective for performance, this proactive storage strategy requires adequate base hardware and careful capacity planning to ensure resources are not over-provisioned.

The following command increases the pre-allocated space in a Db2 schema to 200G.

```
db2 "ALTER TABLESPACE   DBNAME_DATA_TBS INCREASESIZE 200G";
```

Add Custom DB indexes to Improve Search and Retrieval Rates

To optimize report searching and retrieval performance, and to avoid transaction timeouts, create the appropriate indexes before starting the ingestion process or in the early stages of ingesting reports.

You can use ACCE to add the indexes if the Sections tables are small, but to avoid potential transaction timeouts on larger tables, add the indexes using the native database tools.

When creating indexes, consider using composite indexes on relevant column combinations, such as "customer number and report date range" or "statement_date and object_id". Ensure that you follow standard best practices for creating database composite indexes.

Adjust the Database Batch Size

Reports can contain thousands of sections. When uploaded, the section metadata is inserted into the database in batches, with the batch size controlled by the *com.filenet.db.CMODJDBCBatchSize* JVM parameter.

The *com.filenet.db.CMODJDBCBatchSize* parameter is implemented as a JVM system property (or startup argument) on the application servers running the Content Platform Engine. By default, the value is set to 2000 and usually works well with all database types. Changing the batch size to over 5000 is not recommended. Note that in your environment, it might also be appropriate to reduce the batch size.

Use the following syntax to add the property to the Content Platform Engine server's startup arguments:

```
-Dcom.filenet.db.CMODJDBCBatchSize=value
```