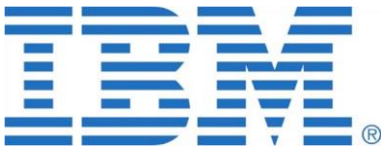


How to Modernize Your CQM Solution on DQM

Technical Client Advocates Team

IBM COGNOS ANALYTICS

IBM® COGNOS® ANALYTICS

Contents

Purpose

The purpose of this document is to convey steps, options, and considerations for modernizing legacy Cognos Business Intelligence content built using the Compatible Query Mode (CQM) and ODBC to the Dynamic Query Mode (DQM) and JDBC.

Disclaimer

The information contained within is subject to change over time and is correct as to the knowledge of the author(s) at the time of writing. This document is provided gratis based on the expertise of field practitioners and is not a result of a thorough consultancy investigation of any one particular circumstance. The reader should consider this as the scope and plan for their conversion project.

This document is informational only and does not convey a contract, obligation, or requirement on behalf of IBM or the customer.

Overview

Many legacy Cognos solutions were built using Compatible Query Mode (CQM) and have not yet been modernized on the newer Dynamic Query Mode (DQM).

Running queries using DQM is essential to taking advantage of the enhanced features in Cognos Analytics. Further, it is required if migrating the legacy content from a local, on-premise installation to the IBM Cognos Analytics on Cloud offering, as CQM is not supported on that platform (the only exception is for Powercube connectivity) as of the date this document was written.

This document will illustrate why converting to DQM is worthwhile, how to get there, and what to consider when troubleshooting.

While CQM packages will be converted to DQM at run time for new features such as Dashboard and Exploration, it is a best practice to still convert the underlying Framework Manager model to DQM to avoid design practices not supported by DQM.

What is Dynamic Query Mode?

The Dynamic Query Mode (DQM) is an enhanced Java-based query mode that offers the following key capabilities:

- Query optimizations to address query complexity, data volumes and timeliness expectations with improved query execution techniques
- Significant improvement for complex OLAP queries through an intelligent combination of local and remote processing and better MDX generation
- Support for relational databases through JDBC connectivity
- Improved caching for Dimensionally Modeled Relational (DMR) package
- Security-aware caching
- New data interfaces leveraging 64-bit processing

Expectations with DQM:

- The drivers are different than CQM (eg. JDBC vs ODBC)
- The queries generated by Cognos may be different than CQM
- Some development techniques *might* need to be optimized to take advantage of DQM's full capabilities, especially where best practices were not previously observed
- Some tuning of the environment and data sources may be needed as the queries are capable of scaling more efficiently than CQM

Assess Your Inventory

Performing an inventory assessment and removing unused reports is recommended as a best practice for regular maintenance of your Cognos environment. However, it is helpful to perform this activity before converting your CQM content to DQM if it has not been done in a while.

Reducing your content will reduce the effort, time, and risk associated with this project by letting the project team focus on the content that matters and not waste resources on reports that are no longer used.

Prepare Your Environment

Consider where the initial conversion should take place. This environment should allow for experimentation with the conversion activity as well as the test/fix cycles required before implementing in a production location.

> *Tip: Many companies perform the conversion from CQM to DQM as part of an upgrade to a newer release of Cognos Analytics. An upgraded development or sandbox environment will likely already exist. In most cases, reusing this environment for the conversion work will minimize risk and disruption to the business users in Production. If one does not exist, consider creating a development/sandbox environment specifically for this activity.*

If your target platform is the Cognos Analytics on Cloud offering, then consider performing the conversion locally, on-premise first. Doing this before uploading the content to the cloud has many benefits, some of which are:

- Less simultaneous changes help isolate the cause for any issues that occur; for example, if performing this directly in the cloud, it will be difficult to identify which was the culprit: the conversion activity itself or the cloud configuration.
- Immediate access to review traces/logs if necessary, without requiring a support case
- Flexibility to start earlier with the conversion and not have to wait for the cloud offering to be fully configured

Configure a JDBC Connection

DQM requires a JDBC connection to your database vendor.

These steps use DB2 as an example. You will need to adjust these steps for your vendor of choice.

1. Copy the following files into the driver directory of your dispatcher install.
   1) Universal driver file: **db2jcc.jar**

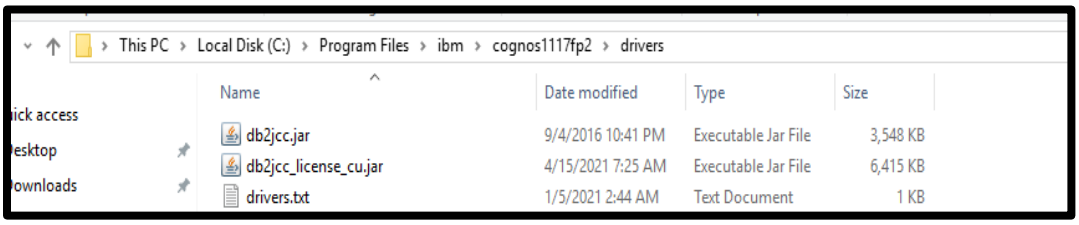
   License files, based on the IBM Db2 platform

   2a) Db2 on Linux®, UNIX, or Windows: **db2jcc_license_cu.jar**

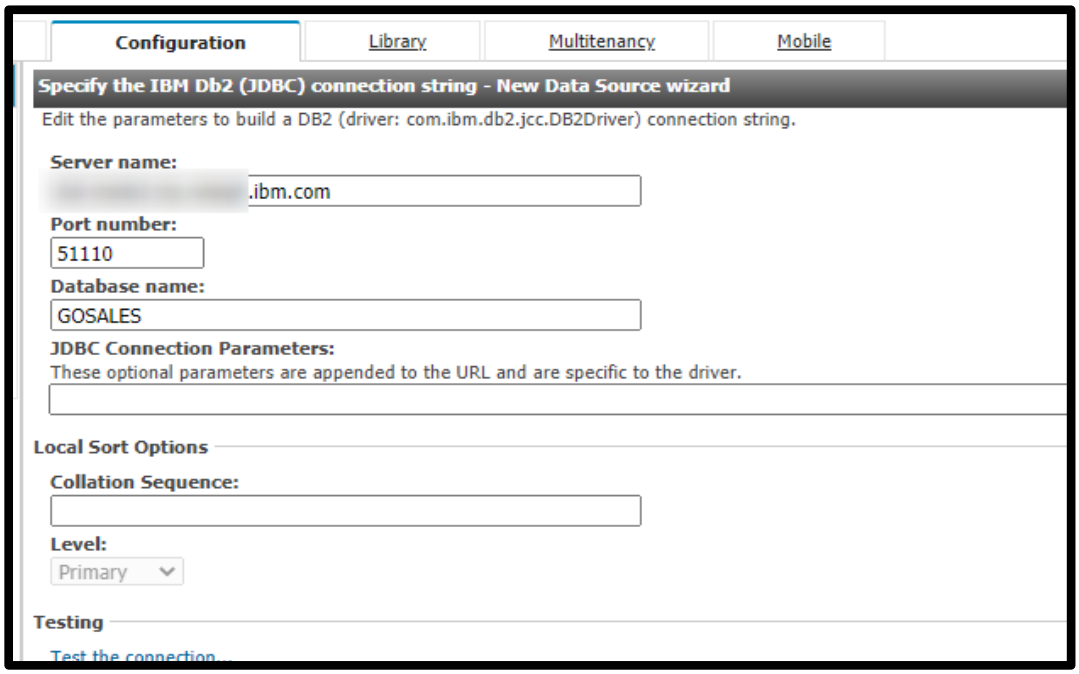
   or

   2b) Db2 on z/OS®: **db2jcc_license_cisuz.jar**

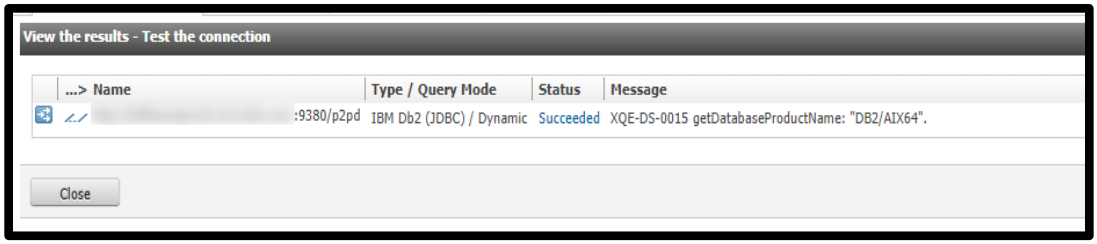
   You can download the files from here



2. Stop and start the the Query Service from Cognos Administration or you can restart your Cognos Analytics environment
3. Login to Cognos and go to *Manage -> Administration console -> Configuration -Data Source Connection*
4. Update your existing CQM DB2 data source connection to include the JDBC details



5. Test the data source to make sure the (JDBC) DQM connection is successful



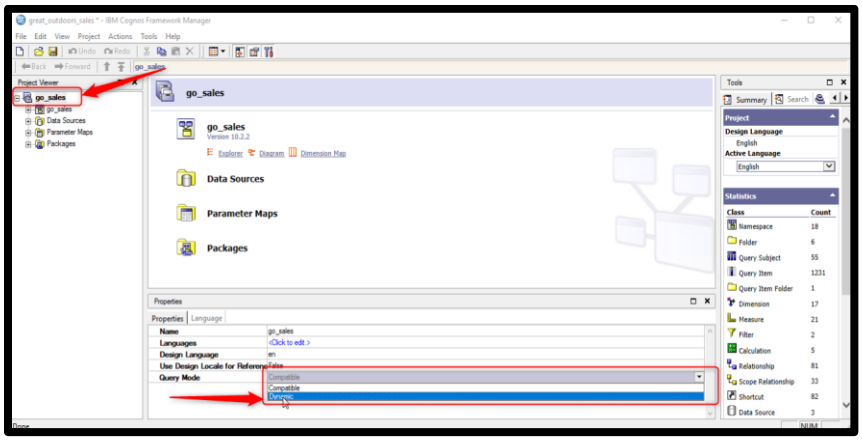
The Conversion

The conversion of Framework Manager packages from CQM to DQM occurs within Framework Manager. You can either convert the entire model or convert the individual packages within the model.

Tip: Decide whether you will perform the conversion before or after the upgrade. In most cases, performing the conversion after the upgrade is the best choice, because the newer version will likely have enhanced fixes to DQM within Cognos Analytics.

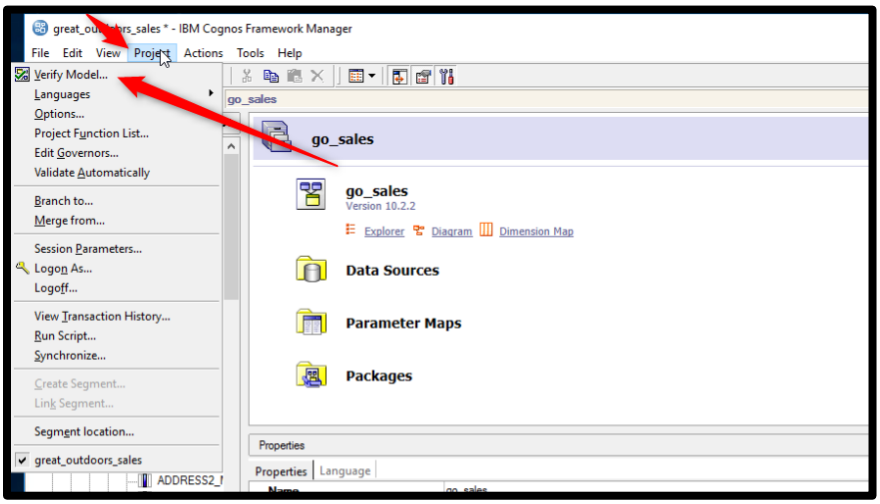
The steps to convert the entire FM Model:

1. Before converting, backup your existing Framework Manager models
2. Launch Framework manager and load the Framework Manager model
3. Select the top level in the Project Viewer and from the Properties pane, change the Query Mode from **Compatible** to **Dynamic**

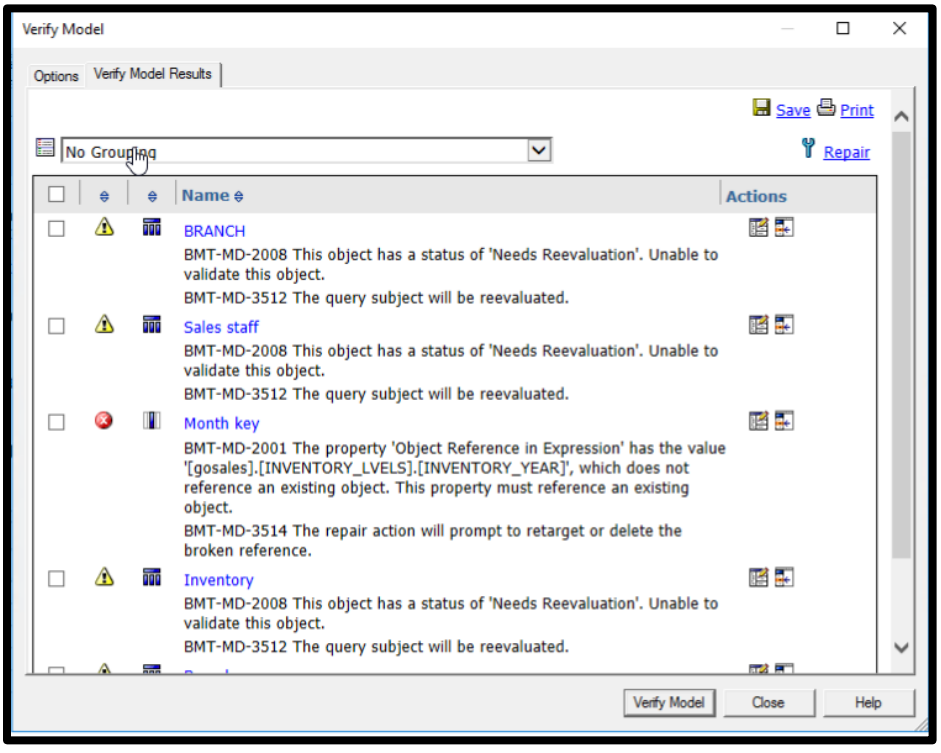


Note – Depending on the size of the model it may take some time to convert the model from CQM to DQM as Framework Manager will validate the Query Subjects.

4. Once the model has been converted, you will want to Verify the model



5. Correct any errors or warnings that are presented after verifying the model

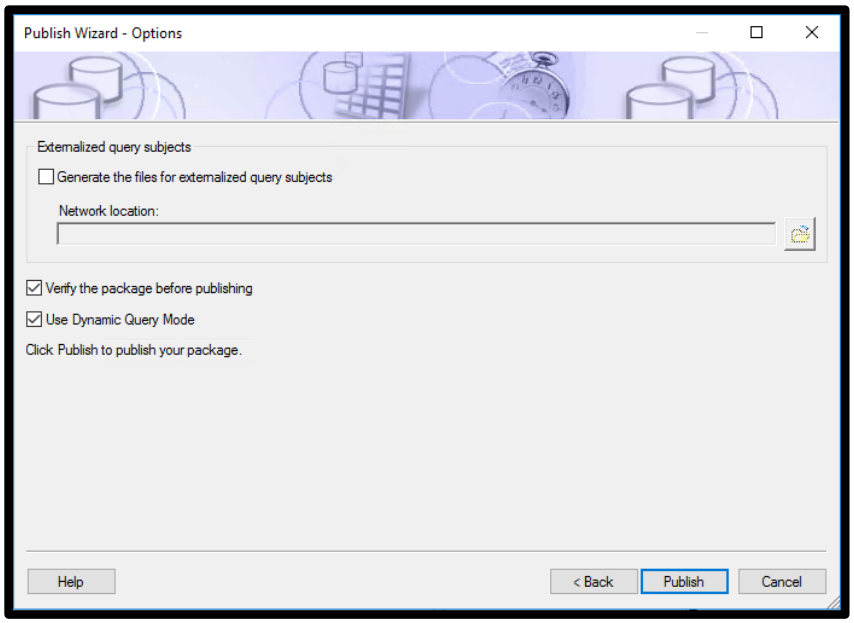


6. Once completed, you can publish the package(s) and repeat these steps for each Framework Manager model.

Note:
While you can publish a CQM model as a DQM package, it is not a best practice since the model will not be validated against the DQM query rules.

The steps to publish a CQM model as a DQM package:

1. Open the publish package wizard
2. Follow your normal steps and settings in the wizard as you typically do
3. In the wizard, select the box that states “Use Dynamic Query Mode”



4. Click Publish

Testing Your Reports

Timing of Testing

If performing an upgrade of the content at the same time as the conversion of CQM to DQM, decide if testing and remediation will occur twice: once after the upgrade and again separately after the conversion or just once after both the upgrade and conversion are performed.

If performing twice, it will be simpler to identify the root cause of any issues that are found; however, it will be more time-consuming to perform. If performed once at the end, it will be faster to perform but could present challenges during troubleshooting.

Automation Tools

To help you validate your reports after modernizing your packages on DQM, there are tools created by IBM Business Partners and sold for a fee that help automate the testing, so that

batch testing can be scheduled during off-hours and repeated on command. There are tools to help with troubleshooting and applying bulk changes as well. If these are of interest, be sure to explore the options online.

Sample Size

Whether or not you use automation tools, you will need to choose a reasonable sample size across several report profiles. If you only have a handful of reports in your overall solution, then testing all of them is feasible. However, if you have hundreds or thousands, you will likely choose a representative set of reports for various categories, such as:

- 100% of business-critical reports, CFO dashboards, etc…
- 5-10% of reports that are profiled as simple, medium, or complex in design
- 5-10% of Highly interactive vs batch scheduled

The percentages above are examples only, but you should decide on how much to test and which categories provide you with a reasonable indicator of where issues may occur and where you might need to further explore and concentrate your efforts.

Test Criteria

During testing you will want to validate the following:

1. Successful Executions – Do the reports run as expected?
2. Are the results correct after comparing the CQM to DQM output?
3. Are the report execution times the same, longer, or shorter in CQM versus DQM?

Troubleshooting Considerations

Typically, if your report and model design follow best practices, there should be little to no correction needed on the reports. However, if you do notice differences, here are some more common areas to review:

1. Many-to-Many Joins are not supported in DQM. DQM requires a bridge table
2. The sort order may be different for reports if not explicitly set
3. Join path resolution may be different
4. DMR/OLAP – Not able to nest objects in a crosstab outside of their natural hierarchical order
   Example - Not able to nest Product Line under Product Type as Product Line is the parent
5. DQM will push as much of the workload to the underlying database. This may cause SQL Generation to be different compared to CQM
6. DQM enforces SQL best practices, meaning queries that “worked” in CQM may fail to execute or provide a warning message when validating the report.

7. Vendor-specific functions may invalidate query cache and cause longer-running reports. Where possible, use Common functions.
8. Where possible, use the DQM Filter Join optimization setting when querying multiple data sources. More details can be found here.
9. DQM automatically converts CLOB columns into VARCHAR columns by using the CAST function. More details can be found here.

If you do encounter issues with your reports or models after migrating to DQM, there are built-in utilities to help you troubleshoot various issues.

Troubleshooting Error Messages

For issues related to error messages, you will want to try and understand what the error message is telling you. Some error messages might be cryptic, but you can search the IBM knowledge base for assistance. Even searching the error message in google can provide helpful hints and solutions on how to correct it.

If you have a report that is failing, and you cannot determine the cause, you will want to try and simplify the report as much as possible to remove the "noise" and isolate the offending part of the report.

If the item is related to an expression in the report, consider if the expression can be re-written to support DQM.

Lastly, if you cannot determine the cause or fix, you can log a support case with our support team for them to assist you. Please provide the following when logging a support case:

1. Screenshot of the error message
2. If possible, a simplified report spec
3. Framework Manager model
4. Data source vendor and version

Troubleshooting Incorrect Results

If after moving from CQM to DQM your report(s) are returning different results, it is best to confirm if the DQM results are correct. Just because they are different does not mean they are incorrect. It is possible for a poorly designed model and report to have presented incorrect results in CQM that went unnoticed by the end-users. DQM follows best practices so it might be interpreting the SQL differently than CQM and provide the correct results compared to CQM.

If you confirm the DQM results are incorrect, you will want to review the Native and Cognos SQL from the report for both CQM and DQM to understand what part of the SQL may be causing the issue. Again, simplifying the report by removing queries and data items that are not tied to

the issue can help speed up the understanding of the issue and solution as this will help simplify the SQL statement.

You will want to review the DQM SQL to see if the join paths are the same as CQM. If you have ambiguous join paths in your model, DQM may be using a different join path between the tables. You will need to restructure the model to resolve ambiguous join paths if this is the case.

Troubleshooting Poor Performing Reports

If you see a performance impact in DQM, you can use the Interactive Performance Assistant that comes with Cognos Analytics to help you narrow down what container on the report is causing the issue. It is a useful tool to help you understand if the issue is related to the query or the rendering of the container. Details on how to enable and use the performance assistance can be found here.

After narrowing down the issue to one or more queries, you will again want to compare the SQL from CQM and DQM to see if there any differences in the query's logic. Are there items present in the Cognos SQL that are not in the Native SQL? If so, most likely additional local processing on the Cognos server is occurring on the result set. If that is the case, you need to review your report for any functions that the database may not support. Can you switch the function to something that the database does support so that the entire query will be processed on the database and not on the local Cognos server?

You can also Validate the report at a level of "Key Transformation" to understand if query cache is being applied. If not, it will give you details on why it is not being applied. The most common causes are user-defined functions (Vendor specific functions) like Oracle's NVL function. Using vendor specific functions in a report can invalidate query cache and cause performance issues. You will want to replace your vendor specific function with a Cognos Common function. In this case, NVL can be replaced with the Cognos Common function coalesce.

More details on troubleshooting report issues can be found here