

You can implement Reporting Workbench reports called Cogito SQL reports that run searches on Clarity, Caboodle, or custom SQL databases instead of Chronicles. Report templates that query SQL databases use the same infrastructure as other Reporting Workbench templates. Users can access them from the Analytics Catalog, and the report results appear in the report viewer.

The primary difference between Cogito SQL reports and standard Reporting Workbench reports is the timeliness of the data. Reporting Workbench reports that run searches in Chronicles return real-time results. Because Cogito SQL reports use data from a SQL database, the results are from the last time data was extracted from Chronicles.

You might implement Cogito SQL reports if you're interested in:

- Reporting on long-term organizational data, potentially with complex SQL queries, and filtering that data using real-time report parameters.
- Reducing the performance burden on Chronicles for large data sets, such as historical orders. You can filter a large number of records using SQL to return a more targeted data set in Hyperspace.
- Leveraging external data in SQL to help filter the result set that users can report on. For example, you could use external patient satisfaction scores in your SQL query to filter a result set so it includes only patients with lower scores.

Drug Utilization [25471] as of Fri 10/15/2021 1:11 PM

Order Hx Saved Views - Original View

Detail List Explore Re-run Report Select All

Filter

ID	Charge Date	Medication Name	Department	Charge	Billing Quantity	Cost Used for Billing	Amount Dispensed	Amount Dispensed Unit
87038	09/22/2021	PREDNISONE INTENSOL 5 MG/ML PO CONC [19210]	MRJ CENTRAL DEPARTMENT	4.15	0.1	3.08	3.000	mL
86457	09/07/2021	DEXTROSE 10 % IV SOLN [2357]	TMK DEPARTMENT	1.01	0.005	0.01	5.000	mL
86458	09/07/2021	DEXTROSE 10 % IV SOLN [2357]	TMK DEPARTMENT	1.68	0.333	0.67	333.000	mL
86470	09/07/2021	DEXTROSE 10 % IV SOLN [2357]	TMK DEPARTMENT	3.04	1	2.00	1000.000	mL
86480	09/08/2021	DEXTROSE 10 % IV SOLN [2357]	TMK DEPARTMENT	3.04	1	2.00	1000.000	mL
86471	09/07/2021	DEXTROSE 10 % IV SOLN [2357]	TMK DEPARTMENT	3.04	1	2.00	1000.000	mL
86481	09/08/2021	DEXTROSE 10 % IV SOLN [2357]	TMK DEPARTMENT	3.04	1	2.00	1000.000	mL
87039	09/22/2021	CARBOPLATIN 50 MG/5ML IV SOLN [65582]	BMK ICU PACIFIC	275.67	13.464	269.28	67.320	mL
87040	09/22/2021	SODIUM CHLORIDE 0.9 % IV SOLN [27838]	BMK ICU PACIFIC	13.75	3	12.50	250.000	mL
87220	09/24/2021	ACETAMINOPHEN ER 650 MG PO TBCR [13410]	OB L&D	1.29	1	0.28	1.000	tablet
87580	09/29/2021	ACETAMINOPHEN 325 MG PO TABS [101]	RXIP HOME INFUSION DEP	0.15	24	0.00	4.000	tablet

Additional Epic Resources

- [Reporting Workbench Setup and Support Guide](#) provides step-by-step instructions for everything you need to do to build and configure Reporting Workbench report templates and reports.
- [Radar Setup and Support Guide](#) provides additional information about setting up the metric framework, which is also used by Cogito SQL reports.
- [COG2030v Cogito SQL](#) training companion walks you through how to plan and build Cogito SQL report templates.

Set Up the Infrastructure to Run Cogito SQL Reports

Before you start using Cogito SQL reports, you need to make sure the necessary infrastructure is in place. Much of the infrastructure used for populating Cogito SQL reports is shared with Radar SQL metrics and SQL metric drilldown reports, so we expect you have completed some of this setup already.

At a high level, here's the infrastructure you need in place for Cogito SQL reports:

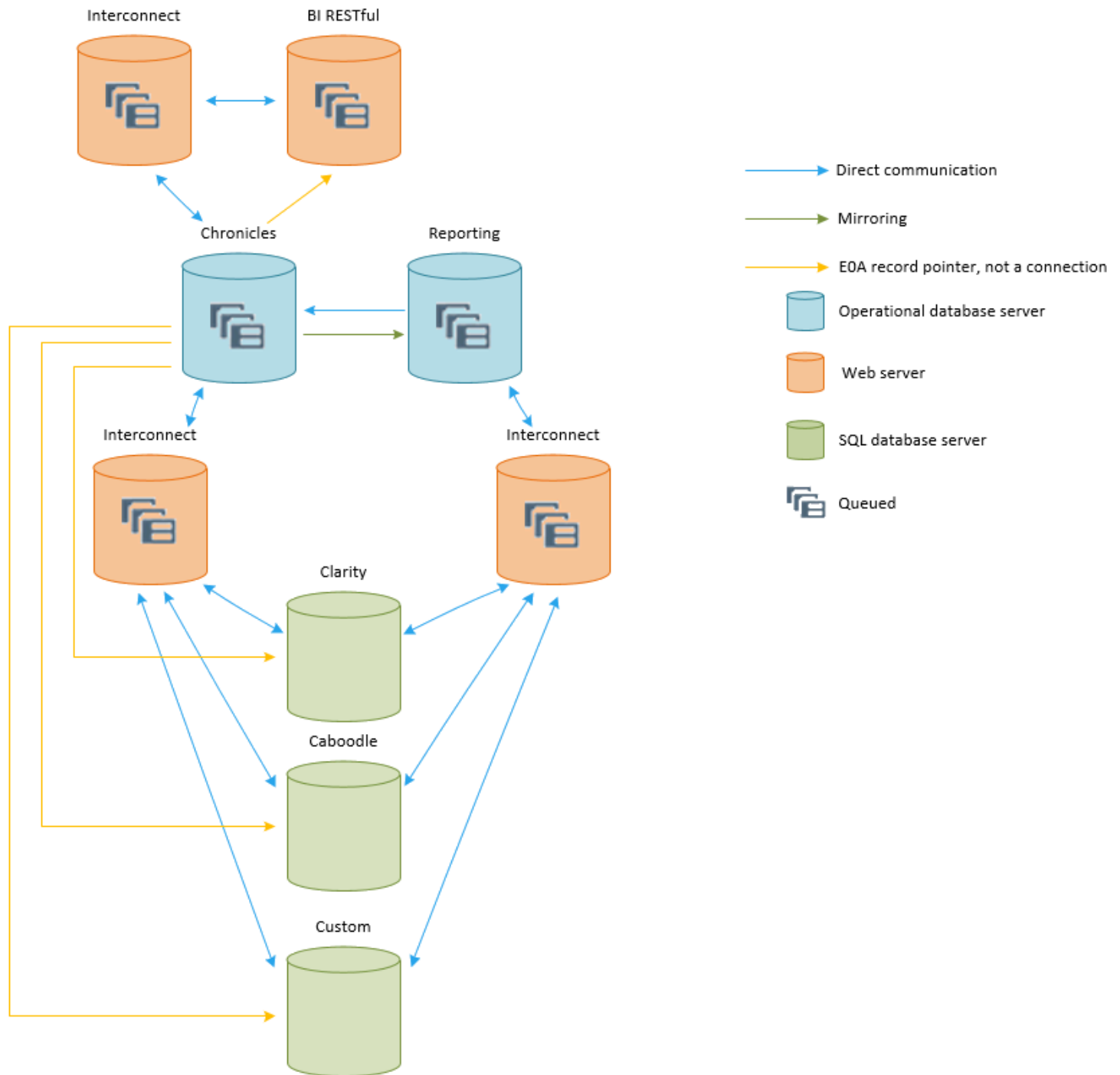
- Database connections for Clarity, Caboodle, or custom SQL databases, which are described in the [Prepare Your Environment to Connect to Your Query Database](#) topic.

- BI RESTful server, which is described in the [Ensure the BI RESTful Server Is Configured](#) topic.
- An Interconnect instance to request and receive SQL query results, which is described in the [Ensure Results Can Be Sent Between SQL Databases and Cogito Using Interconnect](#) topic.
 - If you run reports on a reporting server, you need to set up a separate Interconnect instance that is unique to Cogito SQL reports.
 - If you run reports on a production or utility server, you can use the same Interconnect setup that SQL metric drilldown reports use to run reports.

If you want more information about how these pieces of infrastructure work together for Cogito SQL, refer to the Refer to the [Understand How Cogito SQL Works Behind the Scenes](#) topic.

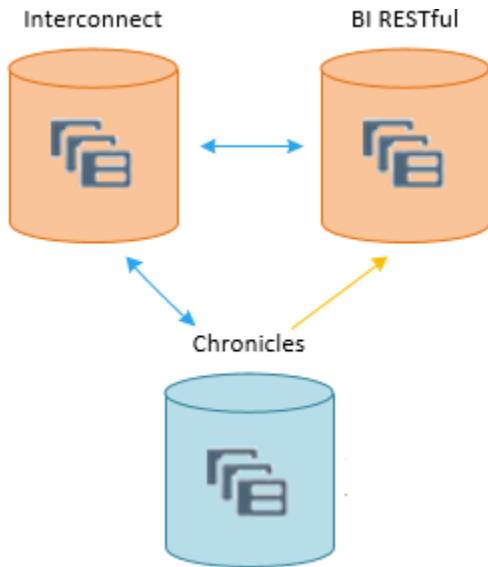
Understand How Cogito SQL Works Behind the Scenes

Cogito SQL report templates and reports use the same Hyperspace framework as any other Reporting Workbench report template. However, behind the scenes, Cogito SQL report templates and reports are driven by additional servers and processes working together to query data from Clarity, Caboodle, or custom SQL databases for report results. The diagram below provides a birds-eye look at how the technical infrastructure works together to ensure users can run and view results for Cogito SQL reports. Keep reading to learn more about each piece of infrastructure.



Dependency Checking for SQL Queries

When an administrator creates or edits a SQL query for a report template, the query is stored in a SQL query (IDJ) record in Chronicles. The system then calculates all tables, columns, and procedures that the query relies on from the SQL database and stores that list of dependencies in a dependency (HMD) record. You can search for dependency records in the [BI Dependency Search activity](#) in Hyperspace to understand the downstream effects that a data model change has on reporting content. Note that dependency checking is available only for SQL queries that search on Clarity or Caboodle.



Here's how dependency checking works in more detail.

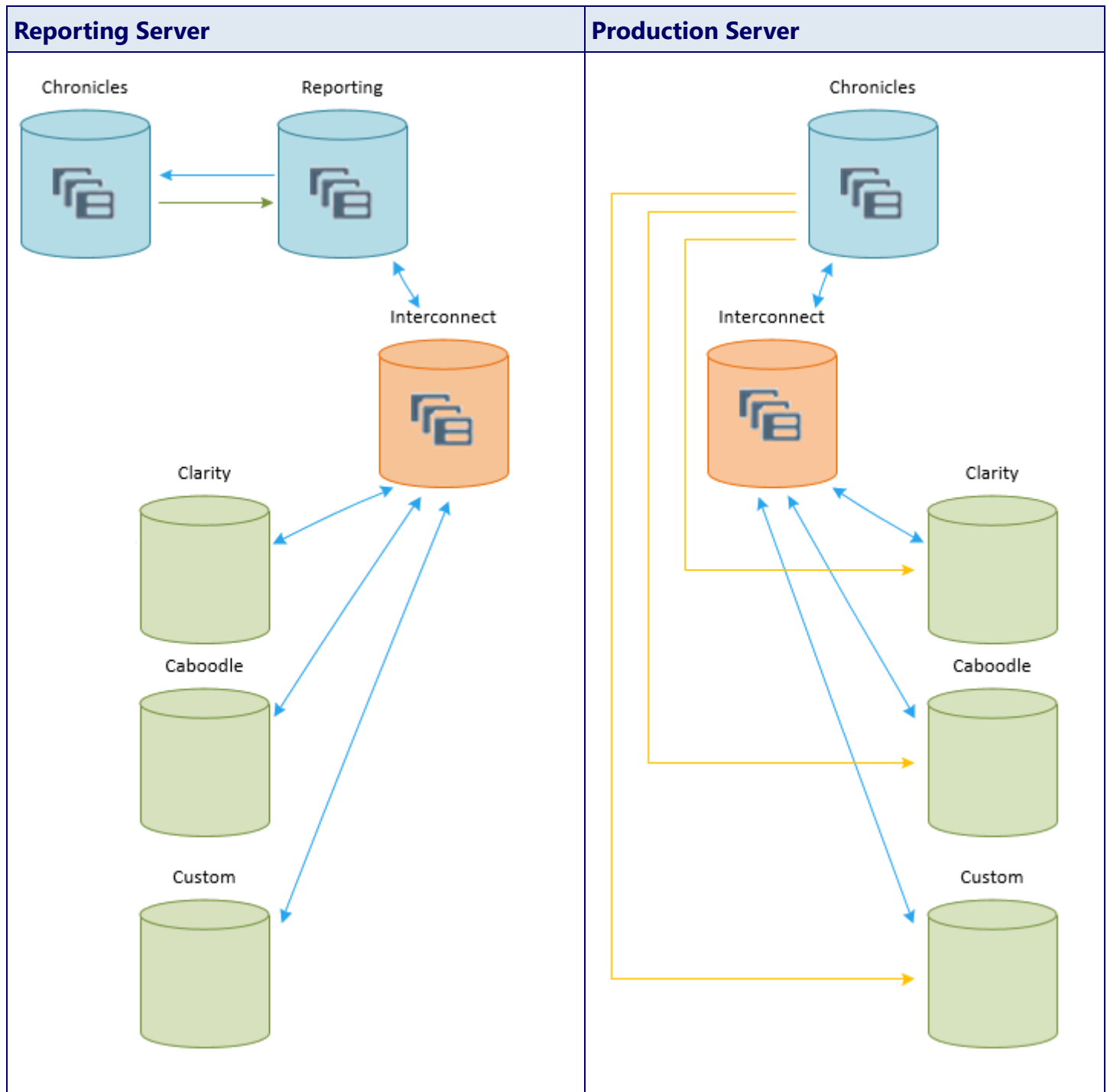
1. The Cogito SQL query record is sent as a job from Chronicles to the CSFSYNC Interconnect queue.
 - This queue is configured as part of your BI RESTful server implementation. For more information, refer to the [Configure an Interconnect Queue for the BI RESTful Server](#) topic.
2. The Interconnect queue listener picks up the job from the Interconnect queue and passes it to the Interconnect instance. Both the listener and the Interconnect instance are set up by your Client Systems team as part of your overall BI RESTful configuration.
3. The Interconnect instance passes the job to the BI RESTful server so it can calculate dependencies.
4. The BI RESTful server parses the query and passes the dependency calculations back to Chronicles using the same Interconnect queue as step 1.
5. Chronicles saves the SQL query's dependencies in a dependency (HMD) record.

Note that some organizations connect their POC environment to a TST BI RESTful server so administrators can modify queries in POC without getting dependency errors when POC doesn't have its own Cogito stack.

Running a Report on the Reporting or Production Server

Users create and run Reporting Workbench reports that query SQL databases the same way they would run any other report. However, under the hood, the query first runs on Clarity, Caboodle, or a custom SQL database to get an initial result set instead of just querying the Chronicles database. Queries are sent to the SQL database through Interconnect. If you follow Epic's recommendation to run these reports on a reporting server, you need to set up an Interconnect queue that's separate from the queue you use for production servers.

The following diagrams depict the overall infrastructure used to run a report on a reporting or production server.



Here's how this process works in more detail:

1. A user or scheduled batch runs a Cogito SQL report.
2. The SQL query for the report is sent from Chronicles as a job to one of the following Interconnect queues:
 - When you follow Epic's recommendations to run reports on a reporting server, the job is sent to the ICSHDACSFASYNC Interconnect queue as described in the [Allow SQL-Based Reporting Workbench Reports to Run in Your Reporting Environment](#) topic.
 - If you run reports on a production server, the job is sent to the CSFASYNC Interconnect queue. This queue is also used for features like SQL metric drilldown reports.
3. The Interconnect queue listener picks up the job from the Interconnect queue and passes it to the Interconnect instance. Both the listener and the Interconnect instance are set up by your Client Systems team.
 - If the external server (EOA) record for the connection to this SQL database is configured to use Windows authorization, it uses the Interconnect server's Windows account for this connection.

- If your organization uses Oracle as your Clarity database platform, the tsnames.ora file also needs to be included in the install directory of the Interconnect instance. If you use multiple Interconnect servers for this workflow, it should be on each server. This setup is required for SQL metric drilldown reports as described in the [Connect the Interconnect Server to Oracle for SQL Drilldown](#) topic.
4. The Interconnect instance passes the job to a SQL database like Clarity or Caboodle based on the connections you define in the [Create a Database Connection Record for a Database](#) topic. The system stores SQL login information in Chronicles in the external server (EOA) record for the SQL database.
 5. The SQL database runs the query to generate the initial set of results.
 6. The query results are passed back through Interconnect to your [specified report queues](#) in Chronicles.
 7. Chronicles further filters down the result set based on any additional report criteria and compiles record data to populate the report's columns.

Prepare Your Environment to Connect to Your Query Database

Cogito SQL reports can run on Clarity, Caboodle, or another data warehouse of your choice. Before Hyperspace can send the template's SQL query to a database, you need to define the connection in every environment. You likely already completed this setup if you use SQL metrics on Radar dashboards.

If you do need to add a non-Epic database or define database connections in your environment, complete the following setup.

Add a Non-Epic Database as a Data Model Type

Complete this setup only if you use a non-Epic database and have not yet added it as a data model type. If you use this database for other reporting tools, like SQL metrics, you completed this setup already.

If you need to add a non-Epic database as a data model type, work with your database administrator to complete the steps in the [Create a Category List Value for a Database](#) topic.

Define Database Connections and Map Them to Each Database

Complete this setup only if you added a non-Epic database with the Add a Non-Epic Database as a Data Model Type section above or you have not yet defined database connections for Clarity or Caboodle. If you use SQL metrics, you likely completed this setup for Clarity and Caboodle already.

To configure database connections, work with your database administrator to complete the steps in the [Create a Database Connection Record for a Database](#) topic. This topic provides more information about how to create a database connection (EOA) for each database and how to map each database connection to a database.

Create and Run Datalink Executions

Complete this setup only if you configured a new database connection for Caboodle or a non-Epic database with the Define Database Connections and Map Them to Each Database section above.

Work with your Clarity administrator to ensure Datalink executions are set up and scheduled to run as described in the [Create the Datalink Action Criteria and Run It](#) topic. This topic provides information about how to:

- Create an action criteria using the Dictionary Refresh Datalink action for Caboodle and any non-Epic databases.
- Create a Datalink execution using that action criteria and schedule the execution to run in the Clarity Console.

Then, work with your Clarity administrator to verify you have executions configured with the released action criteria that are used to populate SQL metrics and scheduled to run. Refer to the [Create a New Execution for Datalink to Populate Metrics](#) topic for more information.

Ensure the BI RESTful Server Is Configured

Cogito SQL uses the BI RESTful server to calculate dependencies for the tables and columns you specify in the SQL query, which is described in more detail in the [Understand How Cogito SQL Works Behind the Scenes](#) topic. The BI RESTful server handles several common BI processes, like SQL metric parsing, so you might have completed this setup already.

If you need to set up the BI RESTful server, refer to the [RESTful Server Setup and Support Guide](#).

Ensure Results Can Be Sent Between SQL Databases and Cogito Using Interconnect

Prerequisites

This topic assumes you've already configured the queues Reporting Workbench uses to run reports as described in the [Establish Where Reporting Workbench Runs Reports](#) topic. This setup is foundational infrastructure for Reporting Workbench, so you should have it set up already.

Reporting Workbench reports that run on a SQL database receive SQL query results from an Interconnect queue. You can determine whether these reports run on the reporting or production server based on the queue you configure.

- Cogito SQL reports run on production servers using the same Interconnect queue (CSFASYNC) and Caché listener as SQL metric drilldown reports. You might have completed this setup already if you previously implemented SQL metric drilldown reports.
- Cogito SQL reports that run on a reporting server need a separate read-only Interconnect instance that is unique to Cogito SQL reports in addition to the CSFASYNC queue on the production server. We recommend running Cogito SQL reports on a reporting server.

Note that, if you have a reporting environment, the system tries to run reports in that environment by default, even if you set up the CSFASYNC queue to run these reports on production servers.

Run Cogito SQL Reports on a Production or Utility Server

Considerations

If your organization uses Windows authentication to connect to a database used for SQL drilldown reports, we recommend you create a separate Interconnect instance for Cogito SQL reports as described in the [Create an Interconnect Instance](#) topic. This instance should be for an Interconnect queue with a type of CSF.

Make sure both of the following are true of the Windows user you configure to run this instance:

- The user is a domain user with only read permissions on the database.
- The user has the same permissions to run Interconnect as the default Local Service user.

Cogito SQL and SQL metric drilldown reports use the same Interconnect queue to send and receive data from a SQL database. Complete this setup only if you have not already set up SQL metric drilldown reports as described in the [Allow Users to View Drilldown Reports for SQL Metrics](#) topic. If you decide to run Cogito SQL reports on a reporting server, this queue still needs to be set up in your production environment. Queues need to be in both RPT and PRD environments since the system defaults to reporting environments.

First, complete the steps in the [Create an Interconnect Queue](#) topic to create a queue specifying the following, if you haven't already:

- Name: CSFASYNC
- Does this queue handle synchronous outgoing messages: No
- Queue type: CSF

Then, work with your ECSA so they can complete the Interconnect configuration described in the [SQL Drilldown Reports](#) topic.

Run Cogito SQL Reports on a Reporting Server

Skip this setup if you are configuring an environment that does not have a reporting mirror environment, like POC. To run Cogito SQL reports on a reporting server, you need to work with your Client Systems team to use an Interconnect instance connected to the reporting environment. Keep in mind that, if you have a reporting environment, the system tries to run reports in that environment by default, even if you set up the CSFASYNC queue to run these reports on production servers.

The Interconnect instance you set up to run reports on a queue in your reporting environment is similar to your Interconnect setup for SQL metric drilldown reports. When you configure this Interconnect instance in the Configuration Editor on the Interconnect server, be sure to:

- Point your Interconnect instance to your reporting server or servers on the Chronicles Connections tab.
- Specify the below information for the Interconnect queue. Refer to the [Cogito SQL](#) topic for more information.
 - Queue name and descriptor: ICSDACSFASYNC
 - Does this queue handle synchronous outgoing messages: No
 - Queue type: ICSDACSF
 - Threads: 5-10
 - The number of threads determines the number of Cogito SQL reports that can run at any given time. The optimal number of threads varies based on your organization's size and number of report runs. We recommend matching the number of threads you use to the number of daemons you run for your default Reporting Workbench queue. In the Foundation System, the default queue is REPORT, and you can find more information about queue daemons in the [Establish Where Reporting Workbench Runs Reports](#) topic.
- Select the Enable read-only mode check box on the Business Services tab. Keep in mind that this setting affects any queues that use this instance, so you should complete this setup on an instance specific to read-only environments.
- If you decide to run Cogito SQL reports on a reporting server, this queue still needs to be set up in your production environment because the system defaults to reporting environments.

Refer to the [Create an Interconnect Queue](#) topic for more information about creating queues for your reporting environment.

Connect the Interconnect Server to Oracle for Cogito SQL

Complete this setup only if your organization uses Oracle as your Clarity database platform to ensure your Interconnect server can send and receive data from Oracle. Connect your server to Oracle by copying your tnsnames.ora file to the Interconnect server. You need one tnsnames.ora file for each Interconnect instance you're connecting to the Clarity database. If you don't complete this setup, Cogito SQL reports don't run.

To place this file on your Interconnect server:

1. Go to the folder for your Interconnect instance in which Interconnect.exe resides.
2. Copy your tnsnames.ora file to this location.
3. Verify that your tnsnames.ora file includes an entry that's identical to the TNS name you specified in the Database hostname field of the database connection as described in the [Create and Configure the Database Connection for SQL Drilldown](#) topic. For example, if the TNS name in the database connection is oraclere1, your tnsnames.ora file should also have an entry called oraclere1. Note that if the TNS name in your tnsnames.ora file includes .world at the end, the Interconnect server won't connect to Oracle correctly.

Create a Custom Cogito SQL Report Template

If Epic-released Cogito SQL report templates don't meet your needs, you can create your own report templates that query SQL databases. When you create a Cogito SQL report template, you still determine the data set to search, the filters that users can add to narrow results, and the information the report shows. However, behind the scenes, the data that's pulled in to the report is driven by a SQL query you write instead of a search in Chronicles.

When Are Key Features Available?

You can create Cogito SQL templates starting in November 2019, but additional features to make Cogito SQL reports more robust are available in later versions. Here's a high-level summary of the key features that are available each release.

Version	Feature	Setup Information
November 2019	Create reports that run searches on SQL databases to return Chronicles data.	Configure Basic Settings in a Cogito SQL Template
May 2020	Add parameters to the report template's SQL query.	Use Report Parameters to Change the Result Set for a SQL Report Template
November 2020	Include data from SQL columns in Cogito SQL reports.	Include Columns That Show Non-Chronicles Data
February 2021	Create non-Chronicles Cogito SQL reports, which return result sets that are stored outside of Chronicles.	Configure a Cogito SQL Template to Return Non-Chronicles Results
May 2021	Create grouped summaries.	Add a Summary to a Report
August 2021	Create report columns for non-Chronicles data automatically in bulk.	Create SQL Columns in Bulk

Determine Which Type of Cogito SQL Template to Create

Cogito SQL templates provide a flexible framework you can use to pull information from Clarity, Caboodle, or a non-Epic database. When you configure a Cogito SQL template, you need to consider where the data you want to show in both the result set and the report columns is stored. The data sources for your template determine how you configure the template going forward.

Start by determining where a report's results, or individual rows, should be pulled from.

- If results are stored as records or contacts in a Chronicles master file, your report returns results that align with those records or contacts.
 - For example, a report that returns information about patients in your system returns records in the Patient (EPT) master file.
 - You still need to specify the search master file in the template so the report knows what type of results to expect behind the scenes. The primary difference between these Cogito SQL reports and standard Reporting Workbench reports is the search engine used to query results. For Cogito SQL reports, the SQL search engine runs a SQL query in a Clarity, Caboodle, or non-Epic data model to filter the set of report results.
- If results are stored in Clarity, Caboodle, or a non-Epic database and not in Chronicles, your report pulls results from a SQL database. This type of Cogito SQL report is called a non-Chronicles report.
 - For example, a report that returns performance reviews submitted through a third-party site pulls the result set from the SQL database where those reviews are stored.
 - While these report templates still use the SQL search engine to run a SQL query, you define in that SQL query where the report should find results. The result set isn't tied to Chronicles, so you don't enter a master file in the template. Other settings that are associated with Chronicles items, like search extensions, are also disabled in the Template Editor.

After you determine which data source your report should search to find results, you or a BID creates the SQL query to determine the logic the report uses to find results. For more flexible reports, you can include parameters in the SQL query. Report users configure these parameters to adjust the result set. Refer to the [Use Report Parameters to Change the Result Set for a Cogito SQL Report](#) topic for more information.

You also need to decide which information you want to return for each result and where that information is stored. In the report, you add report columns (PAFs) that pull data from Chronicles or a SQL database. If you're configuring a non-Chronicles report, you need to complete additional setup if you want to include relevant information from a Chronicles

master file in report columns. Depending on your use case and where certain data is stored, your template might have a mix of report columns that show either Chronicles or SQL data. Refer to the [Determine Which Columns Appear in a Cogito SQL Report Template](#) for more information.

At a high level, here are the primary differences between a Cogito SQL report that returns Chronicles results and one that returns non-Chronicles results.

Chronicles Cogito SQL Template	Non-Chronicles Cogito SQL Template
Returns only results that are stored as records or contact in a Chronicles master file. Finds results based on the master file and the way in which the data is stored. SQL query must include a column that’s mapped to a Chronicles ID. Can be configured with user-entered parameters. Can include report columns (PAFs) that show SQL data and Chronicles data. Supports launch activity actions configured in the Template Editor or Analytics System Settings and extension-based actions configured only in the Template Editor.	Returns results that are stored in Clarity, Caboodle, or a non-Epic database. Finds results based on the logic defined in the SQL query. SQL query can include any set of SQL columns. Can be configured with user-entered parameters. Can include report columns (PAFs) that show SQL data and, with a setup data extension, Chronicles data. Supports launch activity actions configured in the Template Editor or Analytics System Settings (starting in May 2024).

Configure Basic Settings in a Cogito SQL Template

Most of the steps for creating a Cogito SQL report template are identical to creating any other report template, so refer to the [Create a Template from Scratch](#) topic for complete information. These steps focus on settings that have unique considerations for Cogito SQL templates.

1

To help you track the Clarity and Caboodle objects that populate Cogito SQL reports, use the BI dependency checking framework. When you save a SQL query in a report template, a dependency (HMD) record is created automatically. You can view these dependencies right from the template by clicking View Dependencies or by opening the BI Dependency Editor. For more information about tracking report dependencies, refer to the [Manage BI Dependencies](#) topic.

These dependency records are also checked to ensure a report's dependencies are available before the report is run. For example, dependent tables might be unavailable if the tables are being extracted at the same time a user tries to run the report. If a report's dependencies are unavailable, the report is prevented from running, and an error is logged in the BI Log Viewer for administrators to follow up on. For more information about using dependency checking for Reporting Workbench SQL reports, refer to the [Manage Dependency Checking for SQL-Based Content in Radar and Reporting Workbench](#) topic.

Configure a Cogito SQL Template to Return Chronicles Results

Complete these steps to configure a template that returns results that are associated with Chronicles records or contacts. If your template should return non-Chronicles results, refer to the [Configure a Cogito SQL Template to Return Non-Chronicles Results](#) topic. If you need more information to decide, refer to the [Determine Which Type of Cogito SQL Template to Create](#) topic. Follow the steps according to your version.

Configure a Cogito SQL Template to Return Chronicles Results Starting in May 2023

1. In Hyperspace, create a report template (search: Template Editor).
2. Go to the Search form.
3. In the Search for field, select either Records or Contacts.
4. Enter your master file in the Master file field.
5. In the Query method field, enter SQL.
6. In the Data model field, enter the database the template should query on. You can enter Clarity, Caboodle, or a custom SQL database. If you include a custom SQL database, it must have a data model (E0A) record and include data that links back to Chronicles as described in the [Prepare Your Environment to Connect to Your Query Database](#) topic.

7. To continue configuring your template, go to the [Write a SQL Query to Search and Return Report Data](#) topic.

Configure a Cogito SQL Template to Return Chronicles Results in February 2023 and Earlier Versions

1. In Hyperspace, create a report template with a type of Workbench (search: Template Editor).
2. Go to the Query Template form.
3. In the Search master file field, enter the master file where the records or contacts returned by the query are stored.
4. In the Search engine field, enter SQL.
5. In the Search on field, specify whether the query searches contacts or records to find results.
6. In the Data model field, enter the database the template should query on. You can enter Clarity, Caboodle, or a custom SQL database. If you include a custom SQL database, it must have a data model (E0A) record and include data that links back to Chronicles as described in the [Prepare Your Environment to Connect to Your Query Database](#) topic.

7. To continue configuring your template, go to the [Write a SQL Query to Search and Return Report Data](#) topic.

Configure a Cogito SQL Template to Return Non-Chronicles Results

Starting in February 2021

Complete these steps to configure a template that returns results that are stored outside of Chronicles. If your template should return Chronicles records or contacts, refer to the [Configure a Cogito SQL Template to Return Chronicles Results](#) topic. If you need more information to decide, refer to the [Determine Which Type of Cogito SQL Template to Create](#) topic. Follow the steps according to your version.

Configure a Cogito SQL Template to Return Non-Chronicles Results Starting in May 2023

1. In Hyperspace, create a report template (search: Template Editor).
2. Go to the Search form.
3. In the Search for field, select Defined by search. You determine where the report finds results in the SQL query.
4. In the Query method field, enter SQL.
5. In the Data model field, enter the database the template should query. You can enter Clarity, Caboodle, or a custom SQL database. If you include a custom SQL database, it must have a data model (E0A) record and include data that links back to Chronicles as described in the [Prepare Your Environment to Connect to Your Query Database](#) topic.

6. Continue configuring your template, go to the [Write a SQL Query to Search and Return Report Data](#) topic.

Configure a Cogito SQL Template to Return Non-Chronicles Results in February 2023 and Earlier Versions

1. In Hyperspace, create a report template with a type of Workbench (search: Template Editor).
2. Go to the Query Template form.
3. In the Search engine field, enter SQL.
4. In the Search on field, enter Defined by Search. You determine where the report finds results in the SQL query.
5. In the Data model field, enter the database the template should query on. You can enter Clarity, Caboodle, or a custom SQL database. If you include a custom SQL database, it must have a data model (E0A) record and include data that links back to Chronicles as described in the [Prepare Your Environment to Connect to Your Query Database](#) topic.

6. To continue configuring your template, go to the [Write a SQL Query to Search and Return Report Data](#) topic.

Write a SQL Query to Search and Return Report Data

Starting in November 2019

Prerequisites

To write SQL queries in report templates, administrators need Radar security point 17-Metric SQL Report Writer.

Make sure a report writer who's familiar with SQL is responsible for writing the SQL query. Your SQL query can include any functions native to SQL. You can join as many tables as you want, but be conscious of performance and connection time limits when writing the query.

1. In Hyperspace, open your Cogito SQL template if it's not open already (search: Template Editor).
2. Go to the SQL Query form.
3. Click Edit to enable the SQL Script form.
4. Click Insert Script template to add a template for your SQL query, and write your query within that script template.

Refer to the sections below for more information about writing this SQL query.

Starting in August 2023, you can use the Tab key to indent within the SQL Query forms when writing or editing a query. Keyboard-only users can still leave the text field with the "Ctrl+." and "Ctrl+," hotkeys, similar to when using SmartText boxes in expanded mode.

1.

Considerations for the SELECT Statement in the Query



It is a best practice to avoid using a `SELECT *` statement to select all columns when writing a SQL query. It is better to list out the columns you want to select individually to ensure that the associated dependency (HMD) record is filled in as accurately as possible. Avoid formatting columns in the SQL query whenever possible. Formatting specified in the corresponding report column (PAF) record is compatible with sorting, filtering and summarizing in Reporting Workbench.

Cogito SQL Reports That Return Chronicles Results

If your query returns Chronicles records or contacts, you need to ensure the SQL query has a way to map data in the SQL database with records or contacts in Chronicles. To associate Chronicles and SQL results, you must include a column in your SELECT statement that shows a record ID, CSN, or date and has an appropriate alias.

- If your report template searches on records, include a column with an alias of either ID or CSN, depending on the column's data.
- If your report template searches on contacts, include either a column with an alias of CSN or two columns with aliases of ID and DATE. If your query uses ID and date columns, the date column must return the date as a decimal DTE, like the `PAT_ENC.PAT_ENC_DATE_REAL` column does. If the column doesn't return a decimal DTE, no results appear in the query. If your organization uses Oracle, the alias should be inside double quotation marks (i.e. "DATE" instead of DATE).

- All aliases in a Cogito SQL query must be in all caps. For example, if you need to use a line number, your LINE alias needs to be in all caps. Otherwise, display extensions might not work as expected.

You can include other SQL columns and standard SQL functions in the SELECT statement. These columns do not need to be associated with Chronicles data, so you have more flexibility to include data from custom columns in reports. To show data from a SQL database in a Cogito SQL report, add each SQL column to the SELECT statement and create corresponding report columns (PAFs). Refer to the [Include Columns That Show Non-Chronicles Data](#) topic for more information.

Cogito SQL Reports That Return Non-Chronicles Results

Starting in February 2021

If your report template returns non-Chronicles results, your SELECT statement can include any SQL columns or SQL functions that meet your specific reporting needs.

If you want to show data from a SQL database in a Cogito SQL report, you need to add each SQL column to the SELECT statement and create corresponding report columns (PAF). Refer to the [Include Columns That Show Non-Chronicles Data](#) topic for more information.

Use Placeholders in the SQL Query for More Flexible Reports

Placeholders filter the result set dynamically based on specific logic you write in the SQL query.


You can use the {{REPORT_START_DT}} and {{REPORT_END_DT}} placeholders when writing your SQL query to represent a report's start and end dates. The SQL query then uses the start and end dates in the Report Settings window to filter results when it runs the report. These placeholders work only if the "Date range option available to user" setting is enabled on the template.

You can add user-configurable report parameters to Cogito SQL reports. These parameters are configured behind-the-scenes as placeholders in your SQL query. For more information about enabling parameters in a Cogito SQL template, refer to the [Use Report Parameters to Change the Result Set for a SQL Report Template](#) topic.

You can also use the following released SQL query placeholders. When adding these placeholders to your query, you must use syntax such as the following: {{PLACEHOLDER [[VAL]]}}.

Placeholder	Description
IN_PROPERTY_REQ [[VAL]] [[PROPERTY]]	Determines whether the value equals one of the property's values. If the property has no values, this is false.
IN_PROPERTY_OPT [[VAL]] [[PROPERTY]]	Determines whether the value equals one of the property's values. If the property has no values, this is true.
NOT_IN_PROPERTY [[VAL]] [[PROPERTY]]	Determines whether the value equals none of the property's values. If the property has no values, this is true.
PAT_AGE [[BIRTH_DT_VAL]] [[EVENT_DT_VAL]] [[MIN_AGE_YEARS]] [[MAX_AGE_YEARS]]	Determines whether the patient's age falls within a range at a given time. For example, you could use this to determine whether the patient was a minor at the time of the particular appointment you're including in the metric.
VALUE_IN_RANGE [[CHECK_VAL]] [[MIN_VAL]] [[MAX_VAL]]	Determines whether a value falls within a range.
REPORTING_DATABASE	Replaced with the database the report is being run in.

Placeholder	Description
<code>{{CREATE_RELATED_CTE [[TableName]] [[PROPERTY-1]] [[PROPERTY-2]] ... [[PROPERTY-N]]}}</code> (Available starting in February 2024, November 2023 with special update E10700545, August 2023 with special update E10605170, and May 2023 with special update E10511616)	Creates a common table expression that supports the evaluation of a related group of criteria. This placeholder can only be used within the WITH clause. The first argument in the placeholder is the name of the table you want for the CTE being created. The remaining arguments are your criteria placeholders. After properties are loaded into the CTE, you can use them later in the query for more complex logic. Example of CREATE_RELATED_CTE in a SQL query
CURRENT_DATEKEY (Available starting in February 2024 for Caboodle-based SQL reports)	Determines the date when the query is run. This allows you to skip joining to DateDim or casting the date returned by CURRENT_DT to a date key, which can often be performance intensive.



If you're troubleshooting a report that uses a SQL query with placeholders, run a report trace in Reporting Workbench to validate the values used in the placeholders of the query.
The report's SQL query is stored in the SQL Query (I HRN 452) item in the report run record. If a report returns unexpected results, you can troubleshoot by copying the query in that item and running it in your SQL management application to further investigate.

Example SQL Query

Here's an example SQL query that could be added to a report template.

```
SELECT
    PAT_ID as ID
FROM
    PATIENT
WHERE
    BIRTH_DATE>={{REPORT_START_DT}} AND BIRTH_DATE<={{REPORT_END_DT}}
```

This query searches the PATIENT table for all patients whose birth date is within the date range specified in the Report Settings window for the Reporting Workbench report. The PAT_ID column has an alias of ID so that the Ad Hoc search engine knows to match each value in the ID column with a specific record ID in Chronicles.

Example of CREATE_RELATED_CTE in a SQL Query

Starting in February 2024
November 2023 by SU E10700545
August 2023 by SU E10605170
May 2023 by SU E10511616

The CREATE_RELATED_CTE adds support for related grouped criteria. Here's an example of a related grouped criteria in report settings.

Report Settings - (New) X

Criteria Display Appearance Summary Print Layout Toolbar Override General

Find Records

Find Criteria

Patient Information ^ i

	Height	Age	Weight
1	6' 2"	43	190 lb
2	5' 9"	23	155 lb
3	<input type="text"/>	<input type="text"/>	<input type="text"/>

Criterion Logic ☐ OR

Here's an example of a SQL query utilizing the CREATE_RELATED_CTE placeholder.

```

Microsoft® SQL Server Script

WITH
{{CREATE_RELATED_CTE [[Test_CTE]] [[C_AGE]] [[C_HEIGHT]] [[C_WEIGHT]]}}

SELECT
    enc.PAT_ENC_CSN_ID as CSN,
    pat.PAT_NAME as PatientName,
    enc.CONTACT_DATE as ContactDate,
    enc.FIN_CLASS_C as FinancialClass,
    enc.REFERRAL_ID as Referral,
    DATEDIFF(year, pat.BIRTH_DATE, enc.CONTACT_DATE) as PatientCurrentAge
FROM PATIENT pat
INNER JOIN PAT_ENC enc on enc.PAT_ID = pat.PAT_ID
INNER JOIN Test_CTE parms on
    parms.C_HEIGHT = enc.HEIGHT and
    parms.C_WEIGHT = enc.[WEIGHT] and
    parms.C_AGE = DATEDIFF(year, pat.BIRTH_DATE, enc.CONTACT_DATE)
WHERE
    enc.CONTACT_DATE >= {{REPORT_START_DT}} AND
    enc.CONTACT_DATE <= {{REPORT_END_DT}} AND
    {{IN_PROPERTY_OPT [[enc.DEPARTMENT_ID]] [[C_APPT_DEPARTMENT]]}}

```

In this example, the SQL query begins with the CREATE_RELATED_CTE in the WITH statement (Note: If you would like to add multiple related groups, separate each placeholder with a comma. This placeholder won't work outside the WITH statement.) In the second JOIN of this query, the new CTE is joined to the PAT_ENC table on the related grouped criteria. The CTE allows you to use OR logic between each new row you specify in your grouped criteria so you can filter your results to only return values within those parameters. For example, based on the values specified in the report settings image, this query will filter on patient encounters that are 6' 2" tall, 190 lbs., and 43 years old **OR** 5' 9" tall, 155 lbs., and 23 years old.

Here is an example of how Microsoft SQL Server resolves the placeholder.

```
WITH
/* Start Auto-Generated Criteria CTE */
Test_CTE AS (SELECT * FROM (VALUES
(@C_AGE_1, @C_HEIGHT_1, @C_WEIGHT_1)
) as criteria (C_AGE, C_HEIGHT, C_WEIGHT))
/* End Auto-Generated Criteria CTE */
```

Here is an example of how Oracle resolves the placeholder.

```
WITH
/* Start Auto-Generated Criteria CTE */
Test_CTE (C_AGE, C_HEIGHT, C_WEIGHT) AS (
SELECT :C_AGE_1, TO_NUMBER(:C_HEIGHT_1), :C_WEIGHT_1 FROM dual)
/* End Auto-Generated Criteria CTE */
```

Use Report Parameters to Change the Result Set for a Cogito SQL Report

Use parameters in your SQL query to return a more targeted result set from Cogito SQL reports. Users can configure these parameters in the Report Settings window. When they run the report, the SQL query returns a result set based on the parameter values the user specified. Any parameters not handled in the SQL query are used to filter the result set returned by the SQL query as usual.

Parameters in SQL queries allow you to:

- Find a more specific result set for large sets of data more quickly. When you use parameters, you ensure the data the SQL query returns is limited to the specific results and delivers a more targeted set of data to Chronicles. Reporting Workbench can search SQL databases and return data faster than Chronicles, which means your operational database shoulders less of the burden for running reports.
- Create a more flexible and extensible report template. Instead of hard-coding certain values into your SQL query, you can use placeholders for parameters so users can choose which values the SQL query uses when returning the result set for reports. For example, if the report should return results only for a clinician's login department, you can add a placeholder for a department parameter that uses a dynamic extension. Each person who runs the report then sees different results based on their login department.

Here's how it works. Let's say you create a Cogito SQL report template that is supposed to find all patients based on their legal sex, and you want to add a parameter so users can select which sex to filter results by. You have two options:

- Add the Legal Sex parameter to the template like usual. When a user runs the report, the SQL query returns a list of all patients. Chronicles then filters out any patients without the legal sex the report user specifies in Report Settings. For example, if the user searches for female patients, the SQL query returns all patients, and Chronicles filters that list to include only the female patients in the report results.
- Add the Legal Sex parameter to the SQL query to filter the result set returned from a SQL database. When a user runs the report, the query returns a list of all patients with the legal sex the report user specifies in Report Settings. If the user searches for female patients, the SQL query returns all female patients, which means Chronicles doesn't have to do additional filtering to exclude patients without a legal sex of female from the report.

You can use parameters in Cogito SQL templates that return Chronicles results and, starting in February 2021, non-Chronicles results. To add parameters to your SQL query, you need to enable the parameter for use. An analyst comfortable with SQL should then add that parameter to the SQL query. If you're adding a parameter to a template that returns non-Chronicles results in February 2021, you need to complete some additional setup to create a placeholder parameter for your template. Starting in May 2021 and in February 2021 with special update E9601694, you can use a released parameter.

Add Parameters to a Cogito SQL Template

Follow the steps according to your version to add a parameter to Cogito SQL templates that return either Chronicles or non-Chronicles results.

Add Parameters Starting in May 2023

1. In Hyperspace, go to the Template Editor (search: Template Editor) and open a Cogito SQL report template.
2. Go to the Criteria form.
3. If you need to add the parameter to the template, click Add criterion. For more information about adding parameters, refer to the [Add Parameters to a Template Starting in May 2023](#) topic. Skip this step if the parameter is already attached to your template.
 - If you need to add a custom list parameter to your template, refer to the [Add a Custom List Parameter to a Cogito SQL Template](#) topic.
4. Select the parameter you want to use in a SQL query.
 - a. Select the Required button to require that users enter a value for this parameter when they run reports.
 - b. Select the Handled in SQL Query checkbox. If you're configuring a template that returns non-Chronicles results, this checkbox is selected already.
 - c. In the Item field, determine which item users can enter in the parameter. For example, you might allow users to select from a specific category list, like Yes and No (I ECT 100).
5. Go to the SQL Query form and click Edit to enable the SQL Script form.
6. Modify your SQL query to handle the parameter you're adding and leave your cursor where the parameter placeholder should be. Select the parameter in the Parameters box and click Insert Placeholder to add it. Refer to the [Handle Different Parameter Types in a SQL Query](#) topic for more information about how to add certain parameters to a SQL query.

The screenshot shows the 'Cogito SQL Query' interface. On the left, under 'Parameters', two parameters are listed: 'Legal Sex - {{C_LEGAL_SEX}}' and 'Zip Code - {{C_ZIP_CODE}}'. Both are highlighted with red boxes. Below them is an 'Insert Placeholder' button. On the right, the 'Microsoft® SQL Server Script' area contains a SQL query template. The query includes a 'WHERE' clause with two conditions, both highlighted with red boxes: 'SEX_C = {{C_LEGAL_SEX}}' and 'ZIP IN ({{C_ZIP_CODE}})'. At the bottom, there are buttons for 'Insert Script Template', 'Manually specify dependencies', 'Generate Columns', 'Restore', and 'Stop Editing'.

In this example, the SQL query returns only patients who are of the specified legal sex and who live in one of the specified ZIP Codes.

Add Parameters in February 2023 and Earlier Versions

1. In Hyperspace, go to the Template Editor (search: Template Editor) and open a Cogito SQL report template.
2. Go to the Query Template > Parameters form.
3. If you need to add the parameter to the template, click Add. For more information about adding parameters, refer to the [Add Parameters to a Template](#) topic. Skip this step if the parameter is already attached to your template.
 - Tip: When you add an item-based parameter, write down the corresponding SQL column and table in the Parameter Description section for when you update your SQL query later.
 - If you're adding parameters to a template that returns non-Chronicles results in February 2021 or a later version, you can use only property-based parameters. The properties you add to these templates are just placeholders for the criteria logic you define later in the SQL query. You can add the same placeholder property once for each criterion you need to define.
 - Starting in May 2021 and in February 2021 with special update E9601694, add placeholder property 71800-Cogito SQL Parameter. Starting in August 2021, this property is called Placeholder Parameter.
 - If you're on February 2021 without the special update mentioned above, you need to create a placeholder property from scratch to add here. Refer to the [Create a Placeholder Parameter for Templates That Return Non-Chronicles Results](#) section below these steps for more information.
 - If you need to add a custom list parameter to your template, refer to the [Add a Custom List Parameter to a Cogito SQL Template](#) topic.
4. Select the parameter you want to use in a SQL query and go to the Additional tab.

- In the Prompt master file and Prompt item fields, determine which records or category values users can enter in the parameter. For example, you might allow users to select from a specific category list, like Yes and No (I ECT 100). Or, you might allow users to select records from a specific master file by entering the master file and .1.
 - Select the Handled in SQL Query check box. If you're configuring a template that returns non-Chronicles results, this check box is selected already.
5. Go to the User Interface > Parameters form and click the parameter you modified in step 4. Select the Required check box to require that users enter a value for this parameter when they run reports.
6. Go to the SQL Query form and click Edit to enable the SQL Script form.
7. Modify your SQL query to handle the parameter you're adding and leave your cursor where the parameter placeholder should be. Select the parameter in the Parameters box and click Insert Placeholder to add it. Refer to the [Handle Different Parameter Types in a SQL Query](#) topic for more information about how to add certain parameters to a SQL query.

RDBMS Platform

☒ SQL Server ☐ Oracle ☐ Teradata

Parameters

Legal Sex - {{C_LEGAL_SEX}}

Zip Code - {{C_ZIP_CODE}}

Insert Placeholder

SQL Script

```
/* Copyright (C) 2020 Epic Systems Corporation
*****
TITLE:
PURPOSE:
AUTHOR:
REVISION HISTORY:
*****/
SELECT
  PAT_ID as ID /* select a record ID column here */
FROM
  PATIENT /* add main logic of query here */
WHERE
  SEX_C = {{C_LEGAL_SEX}} /* add any additional filtering code here */
  AND
  ZIP IN ({{C_ZIP_CODE}})
```

Insert Script Template

View Dependencies

Restore

Stop Editing

In this example, the SQL query returns only patients who are of the specified legal sex and who live in one of the specified ZIP Codes.

Handle Different Parameter Types in a SQL Query

The syntax you need to use when modifying the SQL query depends on the type of parameter you're adding. The table below has some common examples of updates you might make based on the parameter you're adding.

Type of Parameter	How to Update the SQL Query
Multiple-response parameter	You must use the following syntax: <COLUMN> IN ({{<Placeholder>}}). For example, for a multiple-response problem list parameter, you would enter PROBLEM_LIST_ID IN ({{C_PROBLEM_LIST}}).
Category list parameter	Parameters with category list values are handled differently for queries that search Clarity and queries that search Caboodle or a custom database. Searches that query Clarity evaluate the category ID and treat it as a string. Searches that query Caboodle and custom databases evaluate the category name as a string.

Type of Parameter	How to Update the SQL Query
	One exception is when the category list is ECT 100 or 101. In this case we replace Yes and No category values with Y and N for Clarity and 1 and 0 for Caboodle queries because that is how Clarity and Caboodle store those values.
Optional parameter	<p>To make a parameter optional so that users running the report can leave the parameter blank if they don't want the query to filter on it:</p> <ul style="list-style-type: none"> Use the <code>{{IN_PROPERTY_OPT [[VAL]] [[PROPERTY]]}}</code> placeholder in your query, where VAL is the column to check and PROPERTY is the name of your parameter placeholder. <ul style="list-style-type: none"> For example, in a report that searches for patient encounters, you might include a criterion that uses the <code>C_ENCOUNTER_TYPE</code> placeholder so users can filter by encounter type. If they don't specify any encounter types, encounters of all types are included. The encounter type is stored in the <code>ENC_TYPE_C</code> column. To include this optional criterion in your query, enter <code>{{IN_PROPERTY_OPT [[ENC_TYPE_C]] [[C_ENCOUNTER_TYPE]]}}</code>. Note that the <code>[[VAL]]</code> and <code>[[PROPERTY]]</code> placeholders must start and finish with double brackets in the syntax, not double braces. When using optional parameters and a placeholder of <code>IN_PROPERTY_OPT</code>, the system automatically swaps unused parameters for <code>1=1</code>. You cannot use OR logic between optional criteria with this swap because <code>1=1</code> is always true. You can work around this with: WHERE 1 = (CASE WHEN ([[A]] IS NULL AND [[B]] IS NULL) THEN 1 WHEN ({{IN_PROPERTY_REQ [[VAL]] [[A]]}} OR {{IN_PROPERTY_REQ [[VAL]] [[B]]}}) THEN 1 ELSE NULL end)
Custom list parameter	For setup instructions, refer to the Add a Custom List Parameter to a Cogito SQL Template topic.

Add a Custom List Parameter to a Cogito SQL Template

If you want to add a parameter that allows users to select from a custom list of values, you need to configure a custom list parameter and add it to an extension.

- In Chronicles, go to the Programming Point Parameters (E3P) master file and create a parameter. Parameters typically use the following naming convention: <Prefix> CL - <name of custom list> - <Single/Multiple>.
 - Open the parameter and configure the following fields on the General Settings screen:
 - Data Type. Custom List
 - Editable. Yes
 - List. No
 - Go to the Custom List Settings screen.
 - Enter a key value and corresponding name for each custom list value you want to include. The key value is the value that's passed into SQL when you use the parameter, so make sure it's unique and in all caps.
 - For example, in a custom list that shows types of lab tests, you might have a key value of CP with the name Clinical Pathology and a key value of AP with the name Anatomic Pathology.
 - In Chronicles, go to the Extension (LPP) master file and copy extension 50095-OR/INV Supply Usage and Forecasting - Implantable Supply Custom List.
 - Open your copy of the extension and go to the Parameters screen.
 - In the Custom List Parameter parameter, enter the parameter you created in step 1.
- Then, add this extension to a criterion in the template. The following steps are true regardless of version:
- In Hyperspace, go to the Template Editor (search: Template Editor) and open a Cogito SQL template.
 - Go to the Criteria form. In May 2023 and earlier versions, go to the Query Template > Parameters form.

3. Add a criterion by clicking the Add Criterion button. In May 2023 and earlier versions, this button is called Add. In the Add Criterion window:
 - If your template returns non-Chronicles results, add placeholder property 71800-Placeholder Parameter. Note that this property functions only if your template has a master file of HGL (Linkages) in the Search master file field.
 - For templates that return Chronicles results, select an item from the search master file to use as a placeholder. You'll remove this item when you change the data type of the parameter in the next step, so it doesn't matter which one you pick.
4. In the Data type field, enter Custom List.
5. In the Custom list ext field, enter the extension you created above.
6. Update the Name field based on the custom list parameter you're creating and clear the Item field.
7. Go to the User Interface > Parameters form and add a user-friendly label and field name for your custom list parameter.

Determine Which Columns Appear in a Cogito SQL Report Template

To show additional information about a report's results, you use report columns (PAFs). You can show data from a Chronicles master file or you can create columns that show data from columns in a SQL database.

Your follow-up steps for adding columns to a template depends on the type of results returned by your Cogito SQL report and the columns you want to add.

- If your report returns Chronicles results, you can:
 - Add any existing columns that pull data from the same master file as your template's search master file.
 - Starting in February 2021, add columns from a master file other than your search master file by configuring a setup data extension.
 - Create columns that return data from a SQL database.
- If your report returns non-Chronicles results starting in February 2021, you can:
 - Add columns from a Chronicles master file by configuring a setup data extension.
 - Create columns that return data from a SQL database.

The basic steps for adding columns to a report template are the same as any report template. Refer to the [Basic Setup to Create a Report Template](#) topic for more information. If you want to add columns from a master file other than your template's search master file or create a column that shows SQL data, refer to the relevant section below.

Include Columns That Show Non-Chronicles Data

Report writers can include data from any column in the SQL database that the Reporting Workbench report queries, in addition to Chronicles data. You might want to show data from a SQL column in a Reporting Workbench report in scenarios such as the following:

- To include data that's stored outside of Chronicles, like patient satisfaction scores, in a report.
- To support an initiative to replace third-party reporting content, like Epic-Crystal reports, with native Cogito reporting content. If certain data in your third-party reports is not available in Chronicles, you can now pull that data directly into Reporting Workbench.
- To show users meaningful data from a custom Clarity column that your organization created using complex SQL logic.
- To help SQL report writers more efficiently create reporting content in Cogito by allowing them to use columns from a data model that they are already familiar with.

First, add the columns to your SQL query as described in the [Write a SQL Query to Search and Return Report Data](#) topic. After you add the columns to the SQL query, you also need to create report column (PAF) records that are mapped to the SQL column by following the steps below. You need to complete these steps only for SQL columns that you did not alias as ID, CSN, or DATE in the SQL query.

Starting in February 2021, you can't reuse columns that show SQL data across templates. For example, if you create a report column to show the date allergies were last updated and associate it with the PATIENT.ALRGY_UPD_DATE column for one template, you need to create a separate report column to show the allergy last updated in a different template. However, if you copy a template, any columns mapped to SQL columns in the original template can still be used in the copy.

Create SQL Columns in Bulk

Starting in August 2021

You can create columns for a custom Cogito SQL template in bulk using the Generate Columns button in the Template Editor. When you generate columns this way, the system gathers a list of SQL report columns the query uses to show data based on your SQL query. From that list, you can choose to create any report columns that don't exist already on the template, and the system configures the rest.

You can also see existing SQL report columns that are available for use with the template. While these columns are associated with the template, they might not be configured on the template by default. If you don't see these columns on your template, add them on the Columns form of the Template Editor.

To create columns using the Generate Columns button:

1. In Hyperspace, open the template that you want to add columns to (search: Template Editor).
2. On the SQL Query form, click Generate Columns. The SQL Columns from This Query window opens.
3. Select the check box next to the SQL columns that you want to create report columns (PAFs) for. You can optionally select the check box in the table header to create report columns for all SQL columns in the list.
4. Optionally update the names for any report columns you're creating based on your organization's naming conventions. By default, the SQL column name is used as the report column name.
5. If any columns have a data type of Instant, you can optionally change the report column data type to Date or Time in this window, depending on the use case for your column. The data types that appear in this window are pulled from the data type of the associated SQL column. SQL columns with a data type of Date, Time, or Datetime are mapped to the Instant data type in report columns by default.
 - If your SQL query returns data for a column that stores data in a different master file, like a record ID or category value, the system can't identify the correct data type automatically. Instead, it pulls in the data type of the ID, like String or Number, when generating the list of columns in this window. If you have any columns like this that should have a data type of Record or Category, create the column using the data type that's pulled in automatically. Then, open the column in the Column Editor (search: Column Editor) to change the data type to Record or Category and add the master file the data is pulled from.

6. Click the Generate <#> Column(s) button, where <#> is the number of columns you're creating.

SQL Columns from This Query

Columns returned by this Microsoft® SQL Server query:

<input checked="" type="checkbox"/>	PAF Record Name	SQL Column Name	Data Type
New Columns			
<input checked="" type="checkbox"/>	LAST_PARTY_ASGN_END_DTTM	LAST_PARTY_ASGN_END_DTTM	Instant
<input checked="" type="checkbox"/>	DEF_DIVISION_C	DEF_DIVISION_C	Numeric
<input checked="" type="checkbox"/>	DEF_STATUS_C	DEF_STATUS_C	Numeric
<input checked="" type="checkbox"/>	RECORD_STATUS_C	RECORD_STATUS_C	Numeric
<input checked="" type="checkbox"/>	LastPartyCount	LastPartyCount	Numeric
Already Built			
<input checked="" type="checkbox"/>	Deficiency Instance [38986]	ID	Record
<input checked="" type="checkbox"/>	Deficiency Status [38978]	DefStatus	Category

Generate 5 Column(s)
Cancel

- In the Set Base ID for PAFs field that appears, optionally change the base record ID for the columns you're creating. The system uses the next available record ID when creating the columns, starting at the base idea that's specified. You might modify this field, for example, if your organization uses a specific range of custom record IDs for each application.
- Click Accept to finish creating these columns. The columns are added automatically to the Columns form of the Template Editor.

Create SQL Columns Manually

Starting in February 2021, follow these steps to add individual columns from the SQL database to a report template manually.

- In Hyperspace, open the template that you want to add columns to (search: Template Editor).
- On the Columns form, select Create/Find and create a column or copy an existing column. The Column Editor opens.
- If you created a column:
 - In the Used by field, enter Reporting Workbench.
 - In the Field type field, enter SQL Column.
 - Starting in May 2021, if the template is configured to return Chronicles results, enter a master file in the Master File field. In most cases, you should enter the template's base master file unless the column is pulling data from

a networked master file.

Definition and Formatting		Application Specific Features	
Column Definition			
Field type:	SQL Column		
Field scope:	Both Filter and Displayable		
Master file:	EPT [Generic Patient Database]		
Display Extensions			
SQL Time Format:			
Multi-line Delimiter:			
SQL Column Name:	CMR_DATE		
Image Ext:			
Parameters:			
Display Ext:			

- d. If the associated SQL column shows date and time in UTC format, enter Coordinated Universal Time (UTC) in the SQL Time Format field.
- e. If your column uses a multi-line delimiter other than the default \$(10), specify that delimiter in the Multi-line Delimiter field,
4. In the SQL Column Name field, enter either the name of the SQL column or the alias used for that column in the query. This configuration maps the SQL column to the report column (PAF) record. This field is case sensitive. If you use Oracle as your database platform and enter an alias, you must use either all caps or put the alias in quotation marks.
5. Finish configuring any other display settings in the column. When you close the Column Editor, the column is added automatically to the report template.

Starting in August 2021, May 2021 with special update E9700626, and February 2021 with special update E9605337, any copies of SQL report columns (PAF) are tied to the associated released column behind the scenes, if applicable. This behavior ensures you can add copied columns to reports from the same template that the original SQL report column is on.

You can use a utility to unlink copied columns from their source column. The utility is intended primarily as a way to reduce clutter by disassociating copied columns from templates where they're no longer useful. When you unlink a copied column from the released column:

- It no longer appears in Column Search, so report writers can more easily find the columns they do want to add to reports.
- You can't add that copied column to any reports or templates going forward unless the column is included in the template's Template-specific columns field (I HGR 112).
- If the column is on any existing reports, it continues to work as expected.

To use this utility, go to Analytics Administration in text and select Analytics Utilities > Reporting Workbench Utilities > Unlink Copied RW Column.

Add Columns from Additional Master File Contexts to Cogito SQL Reports

Starting in February 2021

When you create a Cogito SQL report that shows Chronicles results, you need to specify a base master file that indicates which Chronicles records or contacts are included in report results. For example, if you created a report template that searches for orders, ORD is the base master file for that template. In most cases, you also need to return additional contextual information that is stored outside of that base master file. For example, your orders report template might also include columns that show patient (EPT) data. To ensure data from other master files appears in the report, you can use a setup data extension that links to additional master files in the template.

Setup data extensions are also useful for Cogito SQL report templates that return non-Chronicles results. While these templates aren't configured with a base master file, you need a setup data extension to include Chronicles data in columns from any master file. For example, you can create a report that returns records for patients who might not be in your system yet. With the setup data extension, you can include columns that show contextual data from Chronicles for any rows in the report with a corresponding Chronicles record or contact. The columns are blank for any rows without a corresponding Chronicles record or contact.

Create a setup data extension specifically for Cogito SQL report templates by following the steps below. The report template that you add this extension to must include a report column (PAF) that's mapped to a SQL column. For more information about this configuration, refer to the [Include Columns That Show Non-Chronicles Data](#) topic.

To create a setup data extension for a Cogito SQL report template:

1. In Chronicles, go to the Extension (LPP) master file and create an extension.
2. On the Extension screen, configure the following fields.
 - Type (I LPP 30). Enter 514-Reporting Workbench Setup Data Ext.
 - Code Template (I LPP 1000). Enter 715030-RW SQL Setup Data Extension.
3. Go to the Parameters screen and configure the following parameters.
 - Alias. Add the SQL column name or alias that you want to associate with a Chronicles context. This should be the name or alias of the corresponding column in the SELECT statement of your SQL query. The column name or alias you specify must end with one of the suffixes specified below. For example, a column that shows order dates might be named ORDER_DATE to use the required _DATE suffix.
 - _ID
 - _DATE
 - _CSN
 - _LINE
 - INI. Enter the master files that you want to link to in your report template. These master files should match the Chronicles context of the columns you specify above.
4. Go to Hyperspace and open the Cogito SQL report template you want to modify (search: Template Editor).
5. Starting in May 2023, go to the Columns form. In the Setup Data ext field, enter the extension you created in step 1.
 - a. Prior to May 2023, go to the Viewer form. In the Setup Data ext field, enter the extension you created in step 1.
6. In the Databases field, enter each master file you specified in the INI parameter in step 3.

Allow Users to Complete Actions from a Cogito SQL Template

Reporting Workbench reports allow users to complete follow-up actions based on the report's results using report actions (HGAs), which appear in the report toolbar. If you want to add an action to a Cogito SQL report, think critically about whether that action is appropriate for the report. Unlike standard Reporting Workbench reports, which return real-time results, the result set in Cogito SQL reports is based on data from the last ETL. If an action relies on results being up-to-date, it shouldn't be included in a Cogito SQL report.

Only some action types are supported in Cogito SQL reports. Refer to the table below for more information.

	Template Editor	Analytics System Settings
Launch activity actions	Supported for Chronicles Cogito SQL reports	Supported for Chronicles Cogito SQL reports. Starting in May 2024, supported for all non-Chronicles Cogito SQL reports. In February 2024 and earlier versions, supports actions that are not results specific (I HGA 160 = 1-Yes).
Extension actions	Supported for Chronicles Cogito SQL reports	Not supported
After-search extension actions	Not supported	Not supported

Here are some considerations to keep in mind when adding report actions to Cogito SQL reports or templates:

- If you're adding an action that launches an activity to a report or report template in Analytics System Settings, any report columns (PAFs) used by the report action (HGA) must be required columns on the report or report template. If the column isn't required, you can't add the action in system settings.
- Launch activity actions are only supported on non-Chronicles Cogito SQL reports if the report action (HGA) does not specify a base master file (I HGA 100).
- If you're adding an extension action in the Template Editor, the search master file specified in the Cogito SQL template must match the base master file (I HGA 100) in the action.

For more information about how to set up actions, refer to the [Allow Users to Perform Actions on Report Results](#) topic.

Support for Launch Activities on Non-Chronicles Reports

Starting in May 2024

To support actions on non-Chronicles templates, the Report Action Editor allows for multiple columns to be configured for info providers and subsets with a source of Column Data, meaning multiple columns that provide the same relevant information can be added to an Info Provider.

For non-Chronicles Templates, new column mappings will populate in the context section labeled Non-Chronicles at the bottom of Information Provider Setup in the Report Action Editor.

Non-Chronicles

Item information:

	Info Name	Source	Columns
1	PATIENTID	Column Data	Patient ID [27106] INP ID [34421]
2			

Subset information:

	Subset Name	Source	Columns
1	SUBSET	Columns	NameIDDAT Patient ID [27106]
2			

To add a Launch Activity on a template, refer to the [Use the Launch Activity Build Wizard to Configure Actions in the Template Editor](#) topic.

Finalize a Custom Cogito SQL Template

The setup in this guide covers unique configuration options that are available for Cogito SQL report templates. Most of the report template configuration options are identical between Cogito SQL and other Reporting Workbench templates. Make sure to complete any other report template setup to ensure your template meets your needs. For more template configuration options, refer to the [Customizing Reporting Workbench Templates](#) topic.

When you're finished configuring the template, you need to ensure it's accessible to report builders and users with the security to create private reports. You distribute report templates in the Template Manager using report types and report groups. For more information, refer to the [Make Released Templates Available](#) topic.

Cogito SQL: Bells & Whistles

This topic contains additional setup and maintenance tasks that you can complete for Cogito SQL templates as needed.

Configure Optional Settings in Cogito SQL Report Templates

Cogito SQL templates have a few unique considerations that might affect how you configure the template. These settings are optional and use default values if you do nothing.

Specify Search and Return Limits in a Cogito SQL Template

One set of basic settings for Reporting Workbench templates is the number of results the report is allowed to search and return. In Cogito SQL reports, there's no limit to the number of results the SQL query can return in its base result set. Instead, the SQL query returns results using the report-, template-, or system-level setting that determines the maximum records to search.

Reports based on static report templates can show a maximum number of results in the report viewer based on your report-, template-, or system-level settings.

Starting in May 2023, all Cogito SQL reports (including dynamic reports) can show a maximum number of results in the report viewer based on your report-, template- or system-level settings rather than a 10,000 row hard limit.

To ensure optimal performance, we recommend you apply criteria in the Report Settings window of a report to return a more targeted set of results. Keep in mind that, for reports with a 10,000-row limit:

- The report shows the first 10,000 results that are returned and includes a warning that not all results are available. For example, if a SQL query for your report returns 20,000 rows, and you don't add criteria to further filter that result set, users can view only the first 10,000 results in their report.
- You can set more restrictive limits for the number of rows a Reporting Workbench SQL query can return and the number of rows that can appear in the report viewer. However, you can't increase the 10,000-row limit.

Refer to the [Limit the Number of Records a Template Searches and Returns](#) topic for more information about the report-, template-, and system-level settings.

Follow the steps according to your version:

Specify Search and Return Limits Starting in May 2023

1. In Hyperspace, open your Cogito SQL template if it's not open already (search: Template Editor).
2. Go to the Basic Info form.
3. In the Search max field, optionally enter the number of rows the SQL query is allowed to return. This set of rows acts as the base result set for your report. By default, no limit is used.
4. In the Max results to return field, optionally enter how many rows can be included in the report.
 - If this report template is static, the report returns the number of results you specify here unless you override this setting at the report level.
 - If you're configuring a dynamic report template prior to May 2023, SQL-based report templates can't include more than 10,000 results. If you enter more than 10,000, only up to 10,000 results are returned.

Specify Search and Return Limits in February 2023 and Earlier Versions

1. In Hyperspace, open your Cogito SQL template if it's not open already (search: Template Editor).
2. Go to the Basic Information form.
3. In the Search max field, optionally enter the number of rows the SQL query is allowed to return. This set of rows acts as the base result set for your report. By default, no limit is used.
4. In the Return max field, optionally enter how many rows can be included in the report.

- If this report template is static, the report returns the number of results you specify here unless you override this setting at the report level.
- If you're configuring a dynamic report template prior to May 2023, SQL-based report templates can't include more than 10,000 results. If you enter more than 10,000, only up to 10,000 results are returned.

Customize the Message That Appears with Results in a Cogito SQL Report

From a report user's perspective, the biggest difference between a Cogito SQL report and standard Reporting Workbench report is the timeliness of results. Standard Reporting Workbench reports that use the ad hoc engine, not the SQL engine, to find and return results in Chronicles always show information that's near real-time.

Cogito SQL reports query a SQL database to find and return a result set. Because data stored in a SQL database must be extracted from Chronicles or an external data source, the results aren't always up to date with the most recent changes. However, any columns that show Chronicles data for those results are returning near real-time information.

You can configure messages and descriptions to ensure users have this context when reviewing Cogito SQL reports.

1. In Hyperspace, open your Cogito SQL template if it's not open already (search: Template Editor).
2. To modify the message that appears at the top of Cogito SQL reports, go to the Viewer form and edit the Message area text.
 - By default, the following message appears: "This report does not show real-time data. It should not be used to make decisions that require the most recent information." You can remove or modify this message to meet your needs.
 - Users typically expect Reporting Workbench reports to return real-time data, so think carefully before you remove this message. Users should know before they follow up on report data that the result set is based on data at the time of the last ETL.

Determine How Long Cogito SQL Queries Can Run Before Timing Out

Note: This section refers to SQL query timeouts. For more information about Reporting Workbench queue settings, refer to the [Establish Where Reporting Workbench Runs Reports](#) topic.

Cogito SQL reports can take a few minutes to run depending on the complexity of your Cogito SQL query and the number of results it returns. To help ensure these more performance-intensive reports don't slow down the Interconnect server or prevent other queries from running against your SQL database, you can specify a number of seconds the SQL query is allowed to run before it times out.

Override settings provide you with some flexibility for determining how long Cogito SQL queries can run. To determine when a SQL query should time out, the system looks at the following settings in order and uses the first setting configured with a value:

- Report-level setting: Seconds until query timeout (I HRX 70464) (starting in February 2022)
- Template-level setting: Query timeout (I IDJ 1104) (starting in February 2022)
- Database connection setting: Query timeout (I E0A 30961)
- System-level setting: Query runtime limit: (I EAF 17961)
- Default query timeout limit of 240 seconds

Specify Timeout Settings at the Report Level

Starting in February 2022

1. In the Analytics Catalog, open the Report Settings window for the report you want to configure (search: Analytics Catalog).
2. Go to the Override tab.
3. In the Seconds until query timeout field, enter the number of seconds that the report's SQL query is allowed to run before it times out.

Specify Timeout Settings at the Report Template Level

Starting in February 2022

1. In the Template Editor, open the template you want to configure (search: Template Editor).
2. Go to the SQL Query form.
3. In the Query timeout field, enter the number of seconds that the report's SQL query is allowed to run before it times out.

Specify Timeout Settings at the Database Connection (E0A) Level

The timeout limit you specify in a database connection (E0A) also affects any Radar SQL metric drilldown reports that use the same database connection to run their SQL queries.

1. In Hyperspace, go to the Epic-BI Database Connection Manager (search: Epic-BI Database Connection Manager).
2. Select the database connection you want to modify and click the Drill Down button.
3. In the Query timeout field, enter the number of seconds that a SQL query using this database connection is allowed to run before it times out.

Specify Timeout Settings at the System Level

The timeout limit you specify at the system level affects both Cogito SQL reports and Radar SQL metric drilldown reports that aren't overridden at the report or database connection level.

1. In Hyperspace, open Analytics System Settings (search: Analytics System Settings).
2. Go to the Radar Settings tab.
3. In the Query runtime limit field, enter the number of seconds that a SQL query is allowed to run before it times out.

Manage Dependency Checking for SQL-Based Content in Radar and Reporting Workbench

Prerequisites

This feature requires that both of the following are configured:

- The BI RESTful Web Server. Refer to the [RESTful Server Setup and Support guide](#) for more information.
- An Interconnect queue, which varies depending on your version of Epic. Refer to the [Cogito SQL](#) topic for setup instructions.
 - Configure the CSFSYNC queue and Caché listener. The CSFSYNC queue is created automatically, but you need to work with your Epic Client Systems Administrator (ECSA) to set up the Caché listener.

Additionally, the user that accesses the databases for dependency checking must have permission to execute the following objects:

- dbo.GetDmcStatus for Caboodle
- EPIC_UTIL.ESP_CR_TABLE_STATUS_CHECK for Clarity

BI dependency (HMD) records store information about the Clarity and Caboodle objects that populate your reports. Using this information, you can more easily assess the effect that data model changes will have on your reporting content. For general information about BI dependencies, refer to the [Manage BI Dependencies](#) topic.

Dependencies are calculated and stored automatically when you save SQL queries for the following Cogito reporting content:

- SQL-based Reporting Workbench templates
- Manual SQL metric drilldown reports
- Automatic SQL metric drilldown reports
- Automatic and manual SQL metrics

You can open these dependency records from the BI Dependency Editor activity. You can also click a link in the individual record editors, like the Template Editor and Metric Editor, to view dependencies. When opening a dependency in the BI Dependency Editor activity, the dependency records have the same name as the metric (IDN) record for automatic SQL metrics and drilldown reports, the metric query (IDJ) record for manual SQL metrics and drilldown reports, and the report template (HGT) record for report templates.

For SQL metric drilldown reports and Cogito SQL templates that use Clarity or Caboodle objects, the system checks the query's dependencies before running the corresponding report to ensure the latest extracted data is available. By default, reports are prevented from running if the reports' dependencies aren't ready when the system does dependency checking. A report might not be run, for example, if the ETL didn't run successfully overnight. In these cases, you can open the BI Log Viewer to view error information and determine next steps.

Dependencies aren't checked for SQL metrics or Cogito SQL reports that use a database other than Clarity or Caboodle.



Epics uses a third-party SQL parser for dependency checking. For more information about that parser, refer to [Gudu Software's website](#). Troubleshooting tools are available from the Live Demo menu on the website.

Change the Dependency Checking Mode

While we recommend you prevent reports from running when their dependencies aren't ready, you can configure a system setting to either log the errors while still running the report or skip dependency checking on these reports altogether.

1. In Hyperspace, go to Analytics System Settings > Radar Settings tab (search: Analytics System Settings).
2. Select one of the following values in the Dependency checking mode field. The default value is Don't Run.
 - Run. If the report's dependencies are not ready, the report is still run. An error is logged to the BI Log for administrators.
 - Skip checking. The reports are always run. The system doesn't check whether reports' dependencies are ready, and nothing is recorded in the BI Log.

Starting in November 2023, there are separate fields for changing the dependency checking mode in Clarity and Caboodle. If the Caboodle dependency checking mode field is empty, it uses the value in the Clarity field by default.

Manually Calculate SQL-Based Report Dependencies When Automatic Calculation Is Unsuccessful

Starting in November 2019

During each upgrade, the system calculates dependencies for any new or updated SQL-based reporting content automatically. If some dependencies aren't calculated automatically, that information appears in the BI Log Viewer for you to address.

After you resolve any issues that prevented dependencies from being calculated automatically, run a utility to recalculate all dependencies for SQL-based reports in your system. To run this utility:

1. In Analytics text, go to Analytics Utilities > Benchmarking Utilities > Recalculate Query Deps.
2. At the Recalculate dependencies on all query records? prompt, enter Yes.

Cogito SQL: Common Issues

This section contains some common issues that organizations run into when setting up Cogito SQL report templates.

A parsing error appears when using DROP TABLE IF EXISTS in a SQL Query.

Solution

In November 2021 and earlier versions, SQL queries that contain the line [DROP TABLE IF EXISTS] encounter parsing errors due to the SQL parser Epic uses. To fix this, replace that line with [DROP TABLE], which should parse without issue. Starting in February 2022, you should no longer encounter this error.

A parsing error appears when editing a SQL query in the Template Editor.

Solution

When saving a SQL query in the Template Editor, the following error might appear: "An error was encountered while parsing the query. See the BI Log Viewer for more details." This error occurs when you're editing a SQL query in a SQL-based report template in an environment that is not connected to Clarity or Caboodle, like your POC environment. If you do nothing, you can still create Cogito SQL report templates, but the error will continue to appear each time you edit the SQL query in a template.

If you don't have an existing POC stack, connect your environment to a database that doesn't contain PHI, such as TST Clarity or TST Caboodle. For information about how to connect your environment to a database, refer to the [Add a Database to Report On](#) topic.

I can't add a SQL report column to a Cogito SQL report or template.

Solution

Verify the report template is configured as a Cogito SQL template. In Cogito SQL templates, the Search engine field in the Template Editor must have a value of SQL.

Solution

Starting in May 2021, if you're trying to add a column to a Cogito SQL template or a report from it that returns Chronicles results, make sure the SQL report column is configured with a master file in the Column Editor. SQL report columns without a master file can be added only to Cogito SQL templates that show non-Chronicles results and aren't available in Chronicles-based Cogito SQL templates.

Solution

Determine whether the SQL report column is associated with a different report template already.

A SQL report column can be added to only one report template and its related reports. For example, if you add a SQL column that shows the last documented date for allergies to a report, you can add that column to other reports created from the same template. However, you can't add it to a report created from a different template. Similarly, if you add that SQL column to one custom template, you can't add it to a different one.

If you want to use a SQL report column with the same configuration in two different templates, you need to create a new column on the second template from scratch.

Solution

If you're trying to add a copied column to a report or report template, check whether the source column that you copied is associated with any report templates. If you copy a source column that's associated with a report template already, you can use your copy only on reports created from the same template.

Starting in August 2021, May 2021 with special update E9700626, and February 2021 with special update E9605337, you can check the source column that's associated with your copy by checking the Source PAF (I PAF 95) item in Record Viewer.

If you want to use a copied column on a different report template, you need to recreate the column on either that template or a report from it.

Solution

In a rare scenario, you can't add copied SQL report columns to reports created from a released Cogito SQL template. You might run into this scenario if you copied a SQL report column and added it to a report created from a released template in May 2021 without special update E9700626 or February 2021 without special update E9605337. When you install these special updates or upgrade to August 2021 or a later version, the SQL report column on any existing reports continues to work as expected, but you can't add that column to new reports created from the same template. Similarly, if you remove the column from the existing template, you can't re-add it. In this scenario, you need to recreate the column and add it to reports.

Dependency records for Cogito SQL reports aren't created automatically.

Solution

If dependency records aren't available automatically for Cogito SQL reports, it's likely that Interconnect server can't connect to the BI RESTful Web server. Verify that:

- The Interconnect server is configured to trust the RESTful server's SSL certificate.
- The RESTful server's ports are open in the Interconnect server's firewall.

The system can't successfully check the status of dependencies while running a Cogito SQL report.

Solution

This issue can occur if the user specified in the Clarity or Caboodle database connection (E0A) record to run reports doesn't have the necessary execute permissions (search: Epic-BI Database Connection Manager). Execute permissions must be enabled on the following objects:

- For Clarity database connections, the user needs permission to execute EPIC_UTIL.ESP_CR_TABLE_STATUS_CHECK.
- For Caboodle database connections, the user needs permission to execute GetCurrentEtIs and GetDmcStatus.

Cogito SQL reports time out in the report queue before the query runs.

Solution

Verify that an Interconnect instance is set up to request and receive SQL query results for Cogito SQL reports. The Interconnect instance must be set up in the same environment as the queue that runs reports. For example, if you follow Epic's recommendation to run reports on a reporting server, a separate Interconnect instance must be set up in your reporting environment. For more information, refer to the [Run Cogito SQL Reports on a Reporting Server](#) topic. To help you more quickly identify that a report wasn't run because the Interconnect setup is incomplete, the following error appears in the report trace window: "The Radar SQL Service IC queue, ICSHDACSFASYNC isn't set up."

Obsolete SQL-based columns appear as dependencies in Data Courier when migrating a copied template.

Solution

Starting in August 2023

Cogito SQL uses the Template-specific columns (I HGR 112) item to determine which columns can be used on a particular Cogito SQL template. This item is necessary because columns can't be shared across templates, with the exception of copied templates. Columns are added to this item in the following scenarios:

- When you copy a template, any columns in the original Template-specific columns item are included in the copied template.
- When you add a column to a custom template in the Template Editor, that item is added to the Template-specific columns item.

Any columns in the Template-specific columns item are included as dependencies on that template when you migrate it using Data Courier. However, this item might have columns that are no longer relevant to that template. For example, if you remove a column from the list of available columns in your template, the column is not removed from the Template-specific columns item.

To help make migrating Cogito SQL templates easier, you can clean up columns in the Template-specific columns item using a utility in text. Go to Analytics Administration in text and follow the path Analytics Utilities > Reporting Workbench Utilities > Clean Up Template-Specific Columns. Enter the template you want to remove from I HGR 112, and then select the columns that are not relevant to that template.

I can't navigate the SQL Query page using only keyboard workflows

Starting in August 2024, the SQL Query editor for Cogito SQL templates uses a third party component for editing. This transition brings several advantages, including the ability to leverage industry-standard features such as tabbing to delineate specific sections of your SQL query and syntax highlighting. However, it is important to note that within normal Epic usage, tabbing is typically used to navigate between fields. As a result, this feature can sometimes conflict with normal accessibility.

To address this, use the shortcut Ctrl+M to escape tab trapping, which will allow you to tab out of the field. If you want to re-enable tab trapping, press Ctrl+M again.

Additionally, if you require a keyboard-accessible shortcut for inserting parameter placeholders, follow these steps:

1. Open the command palette by pressing F2.
2. Search for "param" to view a list of available parameters for your query.

Extras[?]

Optimizing Cogito SQL Configuration and Report Management

Thomas Dash | Cogito Technical Services

Last published: 3/9/2022



As of the November 2019 release, Cogito SQL was introduced, providing the ability to directly query SQL databases from Reporting Workbench (RW) templates. This provides an opportunity for transitioning some of your Epic-Crystal Integrated (ECI) reports to the Reporting Workbench framework. This article provides guidelines and considerations in areas such as when to batch, queue management, and scalability.

At a high level, here are the primary takeaways that you should consider as you create and implement Cogito SQL reports into your standard workflows.

- Leverage the on-demand nature of Reporting Workbench. Batch only when necessary.
- Test Cogito SQL templates with a production-like data set to calibrate timeout settings.

- Increase the number of daemons on your existing RW Queues and actively monitor them as part of the transition. Align RW Queue daemons and Interconnect (IC) listeners.

Reporting Workbench System Analyses

Organizations tend to see a gradual increase in report traffic on their servers as they expand with additional users and report requests. As you shift Crystal content into the Reporting Workbench framework, you can expect a large volume change on your organization's servers in a short timeframe. In this section, we dive into configuration changes used to optimize throughput with the anticipated higher use of Cogito SQL as part of the Epic-Crystal transition. Key focus areas to highlight as your organization creates Cogito SQL content include the following:

- Calibrate queue, report, template timeout and priority settings
- Align number of RW queue daemons and Interconnect listeners
- Monitor RW queue performance as part of the transition and adjust as needed

Reporting Queue Management

As more reports are pulled into Cogito SQL from Crystal, there may be a significant increase of new activity on our reporting queues that did not exist before. You need to revisit your current reporting queue settings to clearly decide on how these queues will be used to service new Cogito SQL activity. A few of the topics discussed are shown below:

- Increase number of daemons on existing reporting queues
- New queue for batch template 71100 batch jobs

Increase the Number of Daemons on Existing Reporting Queues

New reports running in an organization's RW framework will require the increase of daemons available to run at any given time. Rather than creating a new queue dedicated strictly for Cogito SQL reports, it is recommended to increase the maximum daemon count on your existing reporting queues (with a max limit of 20 daemons on any given reporting queue based on Queue Template 20000). Additional daemons on a queue allow users to run more reports simultaneously on that queue, making that queue more powerful. Distributing the same daemons on a new queue opens the possibility of one queue being overrun while another lays idle. If basing queue distribution off Foundation System, we expect on-demand Cogito SQL reports to run on REPORT, long running Cogito SQL reports will run on RWLONG, and data extracts will go on RWEXTRACT.

Given that each organization is unique in size and reporting activity, there is not one set number of daemons that we can suggest to configure on your reporting queues. Nonetheless, there are a few methods your organization can use to help predict the necessary daemon increase for both on-demand and batched reports.

- On-demand: Utilize view time distributions for ECI content to estimate peaks during the day for ad hoc runs. Once you observe which times in the day are most affected by Crystal content, review and compare those times with your current RW run time distributions to match peaks of existing content with new traffic entering the system.
- Batch: Work with your DBAs to investigate typical ETL completion times using System Pulse. Most organizations start running batched content after the ETL processes with new data. Work with your Cogito TS to run the Queue Analysis tool to evaluate your current reporting queue depth activity by hour of day.

Additionally, ask your Cogito TS about the [Optimize Reporting Queue Performance and Establish Queue Best Practices Sherlock checklist](#) if you would like to revamp your reporting queues on a more holistic level.

New Queue for Batch Template 71100 Batch Jobs

While we do not expect folks to create a new queue specifically for solely running Cogito SQL reports, we do recommend you run batch jobs based on batch template 71100 in their own queue. This separation of batch allocation would prevent potential batch template 71100 backups from impacting other report submissions. Refer to [Setting Up a Batch for Long-Running Content](#) below for more information on batch template 71100.

Considerations for Reporting Queues

Before making considerable changes to your reporting queue settings, check with your ODBAs and Server Systems TS to confirm that your servers will be able to handle running at these increased levels of activity. Your organization should be able to accommodate these new levels but revisiting this with servers administrators will be a good opportunity to get them involved during this transition. Work with your Cogito TS to run the Queue Analysis tool to help evaluate which queues are overutilized and which are underutilized.

Calibrate Queue, Report, Template Timeout and Priority Settings

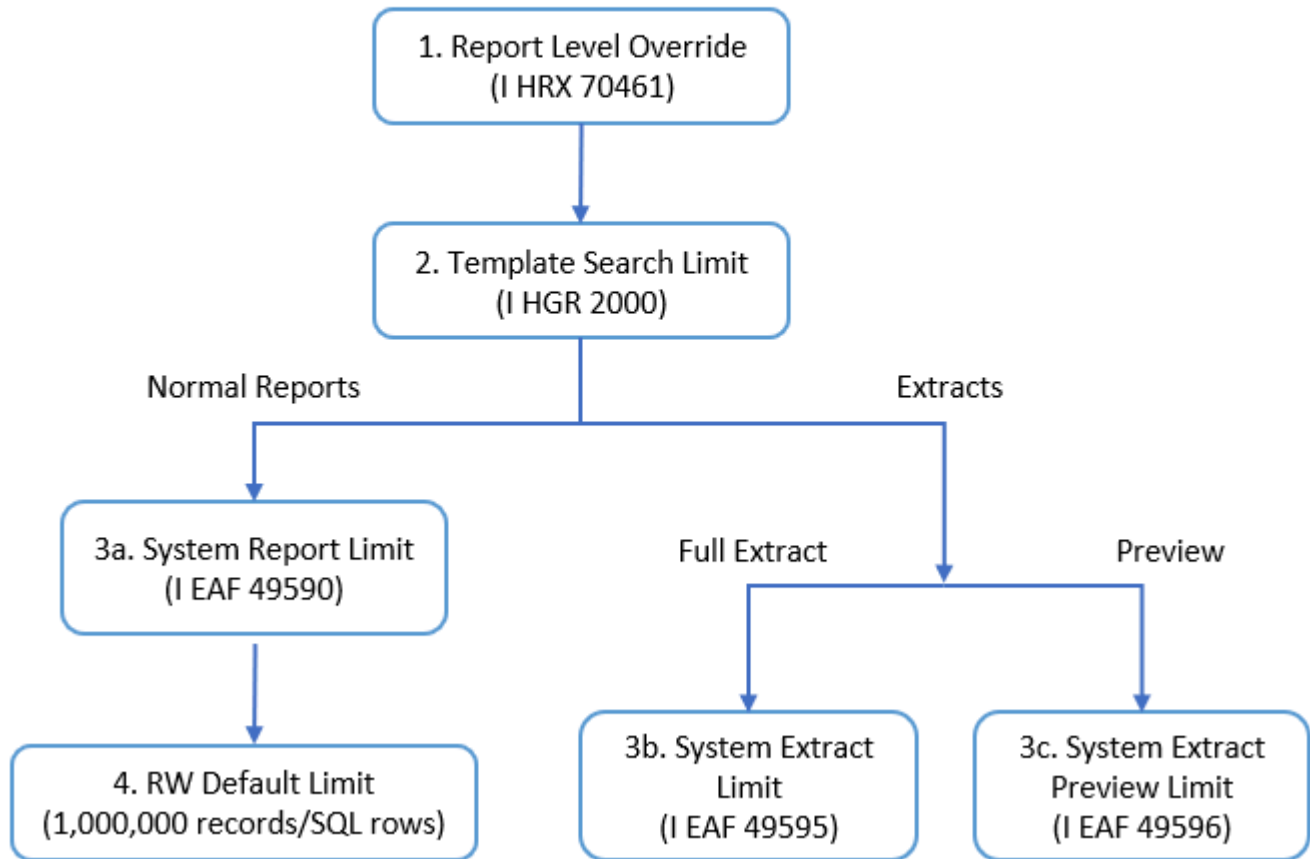
Report and Template-Level Timeout Override Settings

Reporting Workbench has standard guardrails set in place to prevent performance issues from building on our servers. One of these settings is the default four-minute timeout used to error out SQL queries running long against the Clarity/Caboodle database. Prior to February 2022, an override time limit can be set on either the External Server Configuration (E0A) record or the Facility Profile (EAF) record. However, changing this would affect all reports run across the system.

In February 2022, we added functionality for setting a report-level timeout override and a template-level timeout override. This is especially helpful for SQL Extract reports or simply larger Crystal reports that take longer to run than the four-minute timeout. Now, you can change the timeout setting for a few individual reports, leaving the majority of content to respect the existing facility level setting. Being able to now control both the row limit and the timeout for individual reports will give report builders more control over their SQL queries. Before your organization takes this upgrade, consider increasing the default EAF/E0A limit incrementally to capture more Crystal report searches being pulled into Cogito SQL.

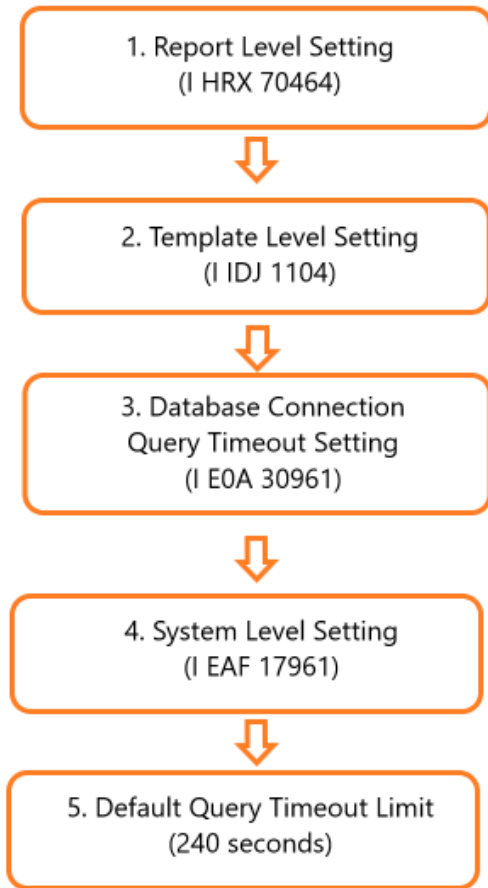
Shown below are two diagrams indicating how the record search limit and query timeout hierarchy will work once the changes in this design have been implemented. If the value in one of the items is null, then we move down to the next level to get the limit. Note the query timeout setting in I E0A 30961 is specific to the the time the report is allowed to query the SQL database.

Workbench Reports: Search Limit Hierarchy



Note: RW's Ad-hoc (non-SQL, non-Cache) search engine's max records to search hierarchy

Cogito SQL Reports: Query Timeout Hierarchy



Note: SQL-based Reporting Workbench query timeout hierarchy

Queue Reset Time Configuration

To determine the appropriate reporting queue for your report, consider the run time of your report and the reset time of the respective queue. The reset time is the longest period of time a report can run on a queue before its process is killed. The REPORT queue, having a default reset time of 1 hour, will be able to handle the majority of the report content. In cases where the report's run time is longer, use the RWLONG queue which has a default reset time of 2 hours. If reports have longer run times that cannot be made more efficient, consider increasing the reset time of RWLONG to your required threshold. As a trade-off, the reports may wait longer in the queue, potentially leading to a RWLONG queue backup given a large set of report submissions.

Review Resource Governor Configuration for Cogito SQL Workloads

[SQL Server Resource Governor](#) allows for granular control over CPU, memory, and other database resources used by certain workloads. If you don't have Resource Governor enabled on Clarity, work with your Cogito Systems TS and mention SLG 6581472. Starting in November 2022, Resource Governor default setup is automatically installed by Kuiper on your Clarity database server.

Create a workload group for Cogito-SQL workloads based on the user name you specify for the Radar connection in the Epic-BI database connection manager. Starting in May 2024, a "Cogito_SQL" workload group is automatically included in the Resource Governor default setup installed by Kuiper.

Work with your Cogito Systems TS to configure the Resource Governor settings for this workload group to set appropriate minimum and maximum CPU percentage values, maximum memory grant and relative priority amongst the other workloads on your system. Use Epic's perfmon scripts and memory grant stored procedures to monitor these workloads and incrementally tune your Resource Governor settings.

Oracle has a similar tool called Resource Manager; however, we currently do not have standard recommendations around configuring it for each organization. If you have concerns about resource contention, reach out to your Cogito Systems TS.

Considerations for Timeout Settings

Creating the capability to set longer timeout limits and write larger SQL queries is a great feature to use, but don't forget why we had those limits in the first place. Work with report writers to determine the balance between running and completing longer more complex reports and protecting system performance. As one side is prioritized, the other may be negatively impacted.

While setting an override is not expected but is the exception to the facility level override, there will be a number of templates or reports that analysts will be responsible for when evaluating the need of any timeout overrides.

Interconnect Management

Match Interconnect Listener Threads to REPORT Daemons

Recent Epic recommendations around Interconnect listeners had placed our default listener count at 2 to run Cogito SQL reports. While this was able to handle capacity for organizations' current usage of Cogito SQL, the transition of many Crystal reports to Cogito SQL may overburden these listeners past their capacity. The number of threads determines the number of Cogito SQL reports that can run at any given moment. The optimal number of threads varies based on an organization's reporting activity. From this, Epic has changed our recommendation to increase the listener thread count of the ICSHDACSFASYNC queue to match that to the max number of daemons on any individual reporting queue (usually REPORT). Additional information can be found in the [Run Cogito SQL Reports on a Reporting Server](#) topic.

Considerations for Interconnect

Before making considerable changes to your Interconnect queue settings, check with your ODBAs and Client Systems - Web & Service Servers TS to confirm that your servers will be able to handle running at these increased levels of activity. Your organization should be able to accommodate these new levels but revisiting this with server administrators is a good opportunity to get them involved during this transition.

When to Consider Batching Cogito SQL Reports

As organizations implement Cogito SQL reports in higher volumes, there will be new tools and workflows introduced that may impact how analysts and end users interact with their reporting content. Most Cogito SQL reports are intended to be used in an ad hoc manner. As reports start to become more complex or are needed on a recurring basis, a consideration to shift to pre-aggregated data comes into play. The few cases when batches should be used are as follows:

- Long-running reports
- Data extracts
- Replication of ECI report subscription model

In this section, we will dive into each of these scenarios along with any associated system, analyst, and user considerations that is needed as your Cogito team is creating Cogito SQL content. Here is a high-level summary noting key capabilities of each ad hoc and batch template option.

Functionality	Ad Hoc	*E1A 1000	*E1A 71000	*E1A 71100
Email report results	x	x		***
Export results to file	x	x	x	
Run short operational reports	x			
Run long running reports		x	x	x
Populate dashboard reports		x		x
Handle report/user dynamic criteria	x			x

* Note: E1A = Batch Scheduler Template. An "X" = "Use this"

*** Can use after-search action 71503 for this use case. See [Notify Users When Report Results from a Batch are Available](#) for more information.

Setting Up a Batch for Long-Running Content

Many reports that your users interact with complete quickly, "quickly" being anywhere from instantaneously to a few minutes depending on a unique user's expectations. When determining to run a report ad hoc or on a batch, you need to consider how long a user should expect to wait for their results once they hit the report's run button. For most users, this runtime limit is expected to be in the window of 5-10 minutes.

For anything that exceeds this limit, consider setting up a batch using batch template 1000 or 71100 to manage long-running content. Listed below are the primary points for when to use these two batch templates:

- Use batch template 1000 for reports with limited set of dynamic parameters
- Use batch template 71100 with caution for reports with a wide set of dynamic parameter values

Batch Template 1000 should be used for the majority of your content. Batch Template 71100 should be used sparingly due to increased risk of creating a large set of report runs.

Using Batch Template 1000

The RW Batch Template (R E1A 1000) can be used to run a manually curated list of reports in Reporting Workbench. Batch template 1000 will take in the RW "Report" you intend to run and requires a "Dept for Printing" and a run "User" for the batch. "Dept for Printing" and "User" are used for dynamic parameters and columns in the report. When using dynamic parameters on your report, consider which users will see applicable results. For example, if your dynamic parameter pulls in service areas from login department and service areas in your Cogito User Security, you could be limited to one service area if your run user is a background user with nothing specified in User Security. If you have additional service areas you would like to reach with this report, you will need to create a separate batch for each service area associated with the "Dept for Printing" selection to produce these results. Additional information can be found in the [1000-RW Batch Template](#) topic.

Using Batch Template 71100

Alternatively, the RW Batch Report Bursting template (R E1A 71100) is used to run reports, models, or dashboards to multiple users with context-specific criteria (such as user's login dept). Batch template 71100 will pull in reports in a "Report List", reports associated with a "Report Tag", or through a list of "Report Models". This batch template then sends reports to all users who share one of the "Report Groups" set on the batch. The batch then creates a report run (HRN record) for each unique user-summary level combination. If a solo report from the batch is based on login department and all users specified have the same login department, the batch would run the report once. If various users have different login departments or the dynamic parameter looks to user security as well, your report may then

run multiple times for each unique combination needed. Additional information can be found in the [71100-RW Batch Report Bursting Template](#) topic.

If you have a dynamic RW report that returns results for different users and then emails out those results, you can use batch template 71100. There is not a mnemonic for emailing out the results on this batch; however, you are able to assign action 71503 to a report to email the run user the results. See the [Notify Users When Report Results from a Batch are Available](#) section for more information.

If you are intending to need various user-summary level combinations when running your reports, batch template 71100 was intended to be used for that behavior. This is a powerful tool - this has the chance to multiply quickly if you are not aware of how many HRNs will be created for a 71100 batch. It is recommended to test the batch job in both a non-production environment and a copy of your production environment to verify it is creating report runs for your expected groups of users. Batch template 1000 will produce the exact number of HRNs you expect and need; however, you will need to create and maintain additional batches for other multiple summary level values. Batch 71100 counters the need for redundant batches but has the risk of unexpected report run submissions if left untested.

Considerations for E1A 1000 and E1A 71100

You might be concerned with the number of reports that can potentially be run from batch template 1000 or 71100. Prior to the February 2022 version, you can counter potential performance issues by setting a limit on the number of reports that a batch job can run. Work with your Epic representative and mention parent SLG 4612009 to run this utility. In the February 2022 version and on, a Max Batch Size parameter has been added as an option on batch template 1000 to limit the number of records that can be exported at a time. Configure the parameter to a value that works for your organization. For batch jobs created from 71100, we recommend running reports on the batch on their own queue so they do not affect other batch or ad hoc submissions.

When initially transferring Crystal content to RW, analysts will be expected to build out the required batch definition, batch job, and batch run records on your [reporting server](#). Additional build information can be found on the [Batch Scheduler Setup and Support Guide](#). Take this opportunity to have analysts with previous knowledge in batch build to train newer analysts on this workflow. Running through real-time build on a few initial records can then allow for remaining build to be shouldered by a broader team if needed.

From a high-level standpoint, use batch template 1000 as your first option; this will get you the results you need as you expect them. Pull in batch template 71100 if you recognize the need and reach of users that will need these batch results dynamically.

Setting Up a Batch for Data Extracts

There are many cases when a hospital needs to send data to an external customer, vendor, or regulatory organization. There may be specific auditing information, preferred formatting on reports, and type of file that is required by the requestor. The RW Extract framework has been consistently used to service these requests for Epic users. As organizations run these extracts on a recurring basis, batches are created from batch template 71000 to specify the required inputs.

Using Batch Template 71000

As the name states, the RW Extract Batch Template is used specifically to run RW data extracts through the RW Batch Scheduler. Required batch template mnemonics include the RW "Report" to be run, the run "User" for the batch, and the "Dept for Reporting". Both the "User" and "Dept for Reporting" mnemonics can affect dynamic parameters and columns specified in the report. More importantly, the "Report" entered must be based on an HGR with a layout of 4-Extract in order to run. Additional information can be found in the General Batch Scheduler Templates Galaxy Guide for [71000-RW Extract Batch Template](#).

Setting Up a Batch to Subscribe Users to Reports Directly in Hyperspace

Previously, if a user wanted to have a RW report scheduled to run on a recurring basis, they would have requested an administrator to set up a batch job to run that report for them. This is because the only way to batch RW reports is

through creation in text; some report writers do not have access to text, only Hyperspace.


While users are not able to create the batch themselves in Hyperspace, they are able to configure their report to run on an existing batch using report tags added in Report Settings. Organizations can establish various frequency-based reports tags for weekly, monthly, and other critical intervals. The only difference for end users used to Crystal reporting is that instead of hitting a “subscribe” button for a Crystal report, they would add the associated report tag to their report.

Two security points that are required for this report tag subscribe functionality are detailed below. The step-by-step process that follows will indicate when each security point is needed.

- RW security point 53 – Edit Public Report Tags
- RW security point 54 – Tags Editor

Before users can “subscribe” to their report, administrators will need to create the necessary HGG (Tag), E1B (Batch Definition), E1C (Batch Job), and E1D (Batch Run) records. Administrators should work with their Cogito TS to set these records up. An overview of the basic step-by-step process is shown below:


1. Create tags for Weekly, Monthly, etc. schedules in the Tags Editor (need RW security point 54 – Tags Editor):

 Tag Select

Search Recent **Create**


Name: Monthly

Internal ID: Auto-generated [Edit](#)

Contact date: 2/20/2022 

Monthly [100002]

Name: Monthly

Type: Report Tags 

Synonyms:

1	Month
2	<input type="text"/>

Related tags:

1	<input type="text"/>
---	----------------------

Tags that list this as related:

2. Create batch jobs on the appropriate schedules, using either the regular RW batch template (1000) or bursting template (71100) if needing dynamic version of the reports

Foundation System Playground

EPIC, USER Job Enter/Edit 2:57 PM CST

Job: 78 - tgd monthly reports

Single and Multiple Response Values

Mnemonic	Value
8. DOWNTIME COMPUTERS	
9. DOWNTIME REPORT NAME	
10. E-MAIL REPORT	
11. SUPPRESS EMPTY RESULTS?	
12. REPORT LIST	
13. REPORT TAG	Monthly[100002]
14. IGNORE AVAIL RESULTS?	NO

Additional Multiple Response Values

Mnemonic	Value
----------	-------

***Note:** Template 1000

Foundation System Playground

EPIC, USER Job Enter/Edit 2:57 PM CST

Job: 80 - tgd burst monthly

Single and Multiple Response Values

Mnemonic	Value
1. REPORT LIST	
2. REPORT TAG	Monthly[100002]
3. Report Models	
4. USER	
5. REPORT GROUP	
6. REPORT QUEUE	REPORT[20000]
7. INCLUDE DASHBOARD RUNS	1-Runs only for source settings

Additional Multiple Response Values

Mnemonic	Value
----------	-------

***Note:** Template 71100

- Set up batch and run for the frequency you want (in this case monthly). Work with your Cogito TS to review the Queue Analysis tool (reference SLG 6399037) to schedule your batch at an off-peak usage time.
- Add tags to reports that you want to schedule on the General tab of Report Settings (need RW security point 53 – Edit Public Report Tags):

Tags

Template Tags	Report Tags
Referrals	Monthly
Authorization and Certification	<input type="text"/>

Considerations for “Subscribe” Process

Analysts will be expected to perform the HGG (Tag), E1B (Batch Definition), E1C (Batch Job), and E1D (Batch Run) [build](#) detailed above. Users with proper security will be able to add one of these frequency-based report tags to any of their reports. While we do not expect most users to have this Edit Public Report Tags security point, we also do not expect ANY user to add their report to a batch. Your organization’s security team policies will govern which users will have the ability to “subscribe” to reports in Hyperspace.

Additionally, if a user clicks "Save As" from a report with this tag assigned, their copy will also have this report tag and will be run on the associated batch. Periodically, analysts should run a report based on report template [3011 - Report Settings Audit](#) with the "Tags on report" criterion to see which reports are included on the frequency-based batches. This step is necessary to validate that your batches are only running an up-to-date list of necessary reports.

When determining the appropriate reporting queue for one of these tagged reports, it is not recommended to assign a queue at the batch level. Instead, use the system-level queue or a template/report-level override if needed. With the number of reports that may run on one of these frequency-based batches, we would not want to bog down any one queue upon each batch run. Keep in mind that the environment (i.e. PRD vs RPT) of the selected queue must match the environment that the batch was built in. Epic recommends building content on the reporting server (RPT).

Managing Dependency Checking for Cogito SQL Reports

In order to maintain accurate content, Cogito SQL uses the BI RESTful server to calculate dependencies for tables and columns that are used in its SQL query. While running a report on a batch, the user may receive notice that the report has failed if its dependencies are not ready to be calculated. This failure occurs when an ETL is running and those dependencies are being modified. While it is not recommended, it is possible for you to configure a system to either continue running a report but produce an error message or skip dependency checking altogether. Additional information can be found in the [Manage Dependency Checking for SQL-Based Content in Radar and Reporting Workbench](#) topic.

© 2024 Epic Systems Corporation. Confidential.