

# VMware Tanzu Greenplum Command Center v6.8 Documentation

VMware Tanzu Greenplum Command Center 6.8

You can find the most up-to-date technical documentation on the VMware website at:

<https://docs.vmware.com/>

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

Copyright © 2022 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

# Contents

Tanzu™ Greenplum Command Center 6.8 Documentation	14
Tanzu Greenplum® Command Center 6.8 Release Notes	16
Supported Platforms	16
Release 6.8.3	16
Release 6.8.2	16
Release 6.8.0	17
New and Changed Features	17
Resolved Issues	18
Known Issues	18
About Tanzu™ Greenplum® Command Center	20
Greenplum Command Center Features	20
Greenplum Command Center Architecture	21
Greenplum Command Center Web Server and Web Application	21
Greenplum Database Performance Monitoring Database	22
Real-Time Query Metrics Collection	23
Command Center Workload Management	23
Tanzu Greenplum Command Center Supported Platforms	24
Supported Operating System Platforms	24
Supported Greenplum Database Versions	24
Installing Greenplum Command Center	26
Installing and Upgrading Tanzu Greenplum Command Center	26
Prerequisites	26
Selecting and Preparing an Installation Directory for Command Center	27
Install the Greenplum Command Center Software	27
Interactive Installation	28
Update the Metrics Collector Extension	31
Install With a Configuration File	32
Non-Interactive Installation with Defaults	33
Upgrade	34
Changed Location for the gpmetrics Directory Starting in Command Center 6.8	34

Installing Multiple Command Center Instances	35
Prepare the Standby Master Host	35
Next Steps	36
Securing the gpmon Database User	36
Authentication for the gpmon Role	37
Setting the gpmon Password	37
Authenticating gpmon with Kerberos	38
Setting the Greenplum Command Center Environment	39
Uninstalling Greenplum Command Center	39
Administering the Command Center Web Server	41
Starting and Stopping the Web Server	41
Configuring the Command Center Web Server	42
Restarting Command Center After Master Failure	42
Viewing and Maintaining Web Server Log Files	43
Administering Command Center Agents	44
Administering the gpperfmon Database	45
Connecting to the gpperfmon Database	45
Backing Up and Restoring the gpperfmon Database	45
Maintaining the Historical Data Tables	45
Administering Log Files, Work Files, and Performance Data	46
Logging and Log Files	46
webserver.log	46
agent.log	46
gpccws.log	46
cli.log	46
Command Center Work Files	47
Automatic Collection of Performance Data	47
Settings	48
Query Monitor Guest Access Settings	48
History Settings	48
Configure Retention Period For History Data	49
gpcc	50
Actions	50



Options	51
Description	51
Examples	51
<b>gpmetrics Schema Reference</b>	<b>52</b>
Alert Tables	53
gpcc_alert_rule	53
gpcc_alert_history	54
Greenplum Database Metrics History Tables	55
gpcc_database_history	55
gpcc_disk_history	55
gpcc_export_log	56
gpcc_pg_log_history	56
gpcc_plannode_history	57
gpcc_queries_history	59
gpcc_resgroup_history	61
gpcc_scan_history	62
gpcc_schedule	62
gpcc_system_history	63
gpcc_table_info	64
gpcc_table_info_history	66
Real-Time Monitoring Tables	68
gpcc_queries_now	68
Workload Management Tables	69
gpcc_wlm_rule	69
gpcc_wlm_log_history	70
Data Loading Tables	71
gpcc_gpss_job	71
gpcc_gpss_log	71
Other Tables	72
gpcc_department	72
gpcc_role_department	72
User-Defined Functions	72
gpcc_delete_department	73
gpcc_queries_per_hour	73
gpcc_queries_per_user	73
gpcc_queries_per_user_max_and_total_spill_size	73
gpcc_queries_per_user_max_cpu	74
gpcc_queries_per_user_max_run_time	74

gpcc_queries_per_user_max_skew	75
gpcc_queries_per_user_rows_out	75
gpcc_queries_top_10_cpu_per_resgroup	75
gpcc_queries_top_10_memory_per_resgroup	76
gpcc_recommendations	76
gpcc_system_per_hour	76
gpcc_update_department	77
<b>Configuration Files Reference</b>	<b>78</b>
<b>Command Center Console Parameters</b>	<b>78</b>
Setting Security Parameters	80
Setting TLS Cipher Suites	81
<b>gpmetrics Configuration File Reference</b>	<b>82</b>
<b>Server Configuration Parameters</b>	<b>84</b>
<b>Securing Greenplum Command Center</b>	<b>87</b>
SSL/TLS Encryption	87
Authentication Options	87
Authorization	88
Self Only	89
Basic	89
Operator Basic	89
Operator	89
Admin	89
<b>Managing Greenplum Command Center Authentication</b>	<b>89</b>
Viewing the Host-Based Authentication File	90
Editing the Host-Based Authentication File	90
Loading a Previous Version of the Host-Based Authentication File	91
<b>Enabling Authentication with Kerberos</b>	<b>91</b>
Before You Begin	92
Add Command Center Principals to the KDC Database	93
Adding Kerberos Principals	93
Set Up Keytab Files	94
Command Center Running on the Greenplum Master Host	94
Command Center Running on the Standby Master	95
Update the Greenplum Database pg_hba.conf File	95

Basic authentication	96
Trust authentication	96
Kerberos authentication	96
Enable Kerberos for Command Center	96
Authenticating With Kerberos on the Client Workstation	97
<b>Monitoring the Greenplum Database System</b>	<b>98</b>
<b>Dashboard</b>	<b>98</b>
System Information	98
Segment Health	99
Database State	99
Segment Status	100
Disk Usage Summary	100
Queries	100
CPU	100
Memory	100
Alerts	100
<b>Greenplum Database Cluster State</b>	<b>101</b>
System Information	101
Segment Health	102
Database State	102
Segment Status	102
Disk Usage Summary	102
Queries	102
CPU	102
Memory	103
Alerts	103
<b>Segment Status</b>	<b>103</b>
Segment Summary	103
Database State	104
Mirrors Acting as Primary	104
Recommended Actions	104
Total Segments	104
Segment Hosts	104
Segment Health	104
Status	104
Replication Mode	105

Preferred Roles	105
Segment Table	105
Cluster Metrics	106
Host Metrics	107
About Skew Calculations	108
Storage Status	109
Disk Usage Summary	109
GP Segments Usage History	110
GP Masters Usage History	110
Recommendations	110
Scheduling Scans	110
Bloat	111
Accuracy	115
Skew	116
Age	117
Monitoring and Managing Greenplum Database Queries	120
Query Monitor	120
Query Metrics	121
Session Information	122
Using the Query Monitor Controls for Queries	123
General Tasks	123
Pausing and Resuming the Query Monitor	124
Filtering Queries	124
Using the Query Monitor Controls for Sessions	126
Query Details	127
Query Execution Metrics	127
Query Text, Query Tags, Inner Queries, and Execution Plan	129
Query Text	129
Query Tags	129
Query Plan and Progress	130
Display Slice Metrics	131
Inner Queries	131
Textual Plan	131
Accessed Tables	132

<b>History</b>	<b>132</b>
Advanced Search	134
Queries Table	136
<b>Query History Details</b>	<b>138</b>
Query History Metrics	138
Query Text and Execution Plan	140
Query Text	140
Query Plan and Progress	140
Display Slice Metrics	141
Textual Plan	142
Accessed Tables	143
<b>Alerts</b>	<b>143</b>
Configuring Alert Rules	146
Cluster Level	146
Master Level	146
Segment Level	147
Query Level	147
Configuring Alert Email	148
Configuring email With Command Center	148
Creating a Send Alert Script	149
SNMP Alert Script Example	150
Prerequisites	150
Setting up SNMP Notifications	151
Testing the SNMP Alert Script	152
<b>Managing Users</b>	<b>153</b>
Display Resource Usage by Role Per Month	154
Display Resource Usage for Specific Roles	154
Display Resource Usage for Departments	154
Display Resource Usage for Roles with No Department	155
Display Resource Usage for Roles Holding a Specific Permission	155
Display Resource Usage for All Permission Levels	155
Create a Department Name	155
Assign a Department Name to a Specific Role	155
Change a Role's Permissions	156
Export Resource Usage of Specific Roles	156
Rename a Department with the gpcc_update_department User-Defined Function	156

Delete a Department Name from the gpperfmon Database with the gpcc_delete_department User-Defined Function	156
Data Loading	158
About Data Loading with Command Center	158
Configuring GPSS Server Access	158
Creating and Managing Data Loading Jobs	160
Prerequisites for Data Loading Feature	160
Creating Jobs	160
Creating a Job with the Default Creation Workflow	160
Creating a Job by Uploading a YAML File	162
Managing Jobs	162
Permissions Required for Job Management	163
About Job IDs	163
Viewing Data Loading Logs	163
Managing Greenplum Database Workloads	165
About Greenplum Command Center Workload Management	165
About Query Assignment Rules and Workload Rules	166
Workload Rules	166
Resource Groups	167
Defining Resource Groups and Resource Attributes	167
Assigning Roles to Resource Groups	168
Monitoring Resource Groups	169
About Resource Group Selection	169
Workload Management	170
Defining Workload Management Rules	170
Defining Workload Management Rules for Queries	170
Defining Workload Management Rules for Idle Sessions	173
Defining and Setting Query Tags	174
Monitoring Workload Rules	175
Importing Resource Queues to Resource Groups	175
Step One: Enable Resource Groups in Greenplum Database	175
Step Two: Preview and Configure Resource Group Imports	176
Step Three: Enable Command Center Workload Management	178

Troubleshooting Enabling Resource Groups	178
Greenplum Database Table Browser	182
Table List	184
Table Details	185
Usage Section	186
Recent Queries Section	186
Table Contents Tab	187
DDL Tab	187
Greenplum Command Center Plugins	189
Query Monitor Help Topics	190
CPU	190
CPU Skew	190
What is CPU Skew?	190
Data Skew	191
Computational Skew	191
Locks	191
Memory	193
Resource Group Memory Configuration Parameters	193
Greenplum Database Memory Configuration Parameters	194
Query Plan Execution	194
Slices and Gangs	195
Operators	195
Scan Operators	195
Join Operators	197
Sort and Aggregate Operators	197
Motion Operators	197
DML Operators	197
Spill Files	198
Investigating Spill File Usage	198
Eliminating or Reducing Spill Files	198
Limiting Spill Files with Server Configuration Parameters	199

Alert Help Topics	200
Segment failure	200
What is this alert?	200
What to do	200
Average memory (segment hosts) exceeds [%] for [min]	201
What is this alert?	201
What to do?	201
Memory (master) exceeds [%] for [min]	201
What is this alert?	201
What to do	201
Total disk space exceeds [%]	201
What is this alert?	202
What to do	202
Query is blocked for [min]	202
What is this alert?	202
Alert details	202
What to do	202
Number of connections exceeds [n]	203
What is this alert?	203
What to do	203
Look for idle or stale connections and terminate them	203
CPU (master) exceeds [%] for [min]	203
What is this alert?	203
What to do	204
See also	204
Out of memory error	204
What is this alert?	204
What to do	204
Query runtime exceeds [min]	205
What is this alert?	205
Alert details	205
What to do	205



Average CPU (segment hosts) exceeds [%] for [min]	205
What is this alert?	205
What to do	205
Spill files for a query exceeds [GB]	205
What is this alert?	206
What to do	206
For more information	206

# Tanzu™ Greenplum Command Center 6.8 Documentation

[Greenplum Command Center 6.8 Release Notes](#)

[Supported Platforms](#)

[About Greenplum Command Center](#)

## **Installing Greenplum Command Center**

- [Installing and Upgrading Greenplum Command Center](#)
- [Securing the gpmon Database Role](#)
- [Setting the Greenplum Command Center Environment](#)

## **Administration**

- [Administering the Command Center Web Server](#)
- [Administering Command Center Agents](#)
- [Administering the gpmetrics Query History](#)
- [gpcc Command Reference](#)
- [Configuration File Reference](#)
- [Securing Greenplum Command Center](#)

## **Using Greenplum Command Center**

- [Monitoring the Greenplum Database System](#)
  - ✦ [Overall Cluster State](#)
  - ✦ [Segment Status](#)
  - ✦ [Cluster Metrics](#)
  - ✦ [Host Metrics](#)
  - ✦ [Storage Status](#)
  - ✦ [Table Browser](#)
  - ✦ [Recommendations](#)
- [Monitoring and Managing Greenplum Database Queries](#)
  - ✦ [Query Monitor](#)
  - ✦ [Query Details](#)
  - ✦ [Query History](#)

- ✦ [Query History Details](#)
- [Managing Alerts](#)
- [Managing Users](#)
- [Data Loading](#)
  - ✦ [About Data Loading](#)
  - ✦ [Configuring GPSS Server Access](#)
  - ✦ [Creating and Managing Data Loading Jobs](#)
  - ✦ [Viewing Data Loading Logs](#)
- [Managing Workloads](#)
  - ✦ [About Workloads](#)
  - ✦ [Managing Resource Groups](#)
  - ✦ [Defining Workload Management Rules](#)
  - ✦ [Importing Resource Queues to Resource Groups](#)
  - ✦ [Troubleshooting Enabling Resource Groups](#)
- [Greenplum Database Table Browser](#)

## **Command Center Help Topics**

- [CPU](#)
- [CPU Skew](#)
- [Locks](#)
- [Memory](#)
- [Query Optimization](#)
- [Spill Files](#)

# Tanzu Greenplum® Command Center 6.8 Release Notes

This document contains release information about Tanzu Greenplum Command Center 6.8 releases. Greenplum Command Center 6.8 provides management and monitoring functionality for Tanzu Greenplum Database 6.

This topic contains release notes for Greenplum Command Center 6.8.

## Supported Platforms

Greenplum Command Center 6.8 is compatible with the following platforms.

- Tanzu Greenplum Database 6.x
- Red Hat Enterprise Linux 6.x<sup>1</sup>, 7.x, and 8.x
- CentOS 6.x<sup>1</sup> and 7.x
- Ubuntu 18.04

See [Tanzu Greenplum Command Center Supported Platforms](#) for the most current compatibility information.

## Release 6.8.3

Release Date: 2022-12-8

VMware Tanzu Greenplum Command Center 6.8.3 is a maintenance release that resolves these issues:

- [32486] Resolves an issue where stale queries were remaining in the Query Monitor after they had completed.
- [183496099] Resolves an issue where Command Center erroneously reported an error about incompatible versions when users upgraded the metrics collector extension by following the instructions in the `update-extension.txt` file installed with Command Center 6.8.2.
- [183880537] Resolves an issue where the Command Center process was using excessive amounts of memory and CPU.

Release Date: 2022-10-03

## Release 6.8.2

Release Date: 2022-10-03

VMware Tanzu Greenplum Command Center 6.8.2 is a maintenance release that resolves these issues:

- [32445] Resolves a crash that could occur when Greenplum Command Center collected its own performance data.
- [32433] Resolves a problem that could cause Greenplum Command Center to terminate the wrong process with signal -9.
- [183359122] Greenplum Command Center now prevents its agent from running a different version of the software. This is indicated by a log message similar to `AgentManager: GPCC Version mismatch, Agent <agent>, webserver <webserver>, will be stopped`

## Release 6.8.0

Release Date: 2022-08-05

VMware Tanzu Greenplum Command Center 6.8.0 is a minor release that includes new and changed features and resolves several issues.

## New and Changed Features

- The default location of the `gpmetrics` directory is now `$HOME` -- the home directory of the user who starts up Command Center -- rather than `$MASTER_DATA_DIRECTORY`.

If you are upgrading from Command Center 6.7.x or earlier and a directory called `gpmetrics` **does not** already exist under `$HOME`, Command Center will migrate the `gpcc.conf` and `gpcc-alert.yaml` files from `$MASTER_DATA_DIRECTORY/gpmetrics` directory to `$HOME/gpmetrics`.

If a directory called `gpmetrics` **does** already exist under `$HOME`, you must copy the `gpcc.conf` and `gpcc-alert.yaml` files over manually before starting Command Center 6.8.0 for the first time.

To specify a default location different than `$HOME` for the `gpmetrics` directory, configure the `gpmetrics_home` parameter in the `app.conf` file. For more information, see the [Command Center Console Parameters](#) topic.

For more information, see [Install the Greenplum Command Center Software](#).

- A user can now configure Command Center to purge history data that exceeds the specified retention period. For more information, see [the History Settings](#) topic.
- A user can now create Workload Management rules for logging queries that match a particular condition. For more information, see the [Workload Management](#) documentation page.
- A user can now filter queries by department name, using the Advanced Search tool in the **Query Monitor** dashboard. For more information, see the [Query Monitor](#) documentation page.
- A user can now search query history by department name, using the Advanced Search tool in the **History** dashboard. For more information, see the [History](#) documentation page.
- A user can now see the timestamp of each inner query.

- A user can now view the size of a table on the table detail page in the **Table Browser**.
- The **realtime Query Details** page now displays a list of the tables the query has accessed. For more information, see the [Query Details](#) documentation page.
- The **Query History details** page now displays a list of the tables the query has accessed. For more information, see the [Query History Details](#) documentation page.
- The `gpmetrics` schema has the following changes:
  - ✦ The `gpmetrics.gpcc_queries_history` table now has a column called `access_tables_info`, which stores the table OIDs of the tables the query accessed.
  - ✦ The `gpmetrics.gpcc_table_info` and `gpmetrics.gpcc_table_info_history` tables now have a column called `last_size_ts`, which stores the time when the current table's size column was last updated.

For more information, see the [gpmetrics Schema Reference](#).

- This Command Center release introduces a new `gpmetrics` configuration parameter `--disable_size_check`. When set to `true`, the Command Center table browser will not collect table size information, in order to prevent performance issues when there are a large number of tables. For more information, see the [gpmetrics Configuration File Reference](#).
- This Command Center release introduces a new console parameter `--XSRFSecure` to protect the cookie Command Center uses. For more information, see the [Command Center Console Parameters](#) topic.

## Resolved Issues

[32297] Resolved an issue where using "@" or "/" in the user field of the Command Center authentication screen resulted in an error.

[181943431] Resolved an issue where users could not use the cursor to edit the S3 file location they entered when creating a data loading job.

## Known Issues

The following are known issues in the current Greenplum Command Center release.

- [181749314] If a user creates a data loading job with the same data source and target table as an existing job, the existing job will be updated even if it was created by a different user.
- [32049] Command Center does not monitor FETCH queries.
- [31723] Submitting a large query string with `psql -c` or libraries causes Command Center performance issues.
- [179572065] When the "Query is blocked for xxx minutes" alert is triggered, the query ID shown on the notification feed may be incorrect.
- [178295227"] Resource group assignment by a query tags rule has no effect if the query tags are defined by `PGOPTIONS` with single quotes.
- [31038] When Daylight Saving Time occurs, Command Center requires a restart to repopulate certain data metrics.

- [30972] When using the `gppersistentrebuild` utility, the **Last Accessed** column in the Table Browser view might be refreshed, although no queries accessed the tables.
- [174667383] When you define a workload management rule that uses the Planner Cost condition, the input field transforms your entry to a double datatype value. This can limit the ability to accurately define large planner costs that may be necessary for the GPORCA optimizer. For example, if you enter the GPORCA maximum cost value as 1457494496834852608, the actual value is converted to 1457494496834852600 for the rule. The value shown in the completed rule definition is the value that the rule enforces, and it may not be the exact value that you entered. Increase the cost value as necessary to cover the cost you want to set as the maximum.
- [174395009] If you recreate the `gp_wlm` extension from outside of Command Center after you have already created workload management rules using the Command Center interface, the rules engine may not run for a period of roughly 1 hour. This behavior occurs because Command Center checks for the availability of the extension every hour. Any changes you make outside of Command Center in this situation will not be visible until Command Center checks for the extension, or until you login to Command Center and access the **Workload> Workload Mgmt** page.
- [167285013] If you upgrade your Command Center installation using the `gpccinstall -u` option and you also specify an SSH binary using the `-ssh-path <path>` option, the customized SSH path will not be used during the installation and the `ssh_path` parameter will not be set in the `app.conf` file.
- When viewing the **Table Browser** view, the calculated size for a partitioned table is incorrect if some child partitions are in different schemas than the root table.
- If there are a large number of tables in a schema in a database, sorting the partition table list by size can take a long time to display.

# About Tanzu™ Greenplum® Command Center

Tanzu Greenplum Command Center is a management tool for the Tanzu Greenplum Database Big Data Platform. This topic introduces key concepts about Greenplum Command Center and its components.

## Greenplum Command Center Features

Greenplum Command Center monitors system performance metrics, analyzes cluster health, and enables database administrators to perform management tasks in a Greenplum Database environment.

Greenplum Command Center provides a browser-native HTML5 graphical console for viewing Greenplum Database system metrics and performing certain database administrative tasks. The Command Center application provides the following functionality:

- Interactive overview of realtime system metrics. Drill down to see details for individual cluster hosts and segments.
- Detailed realtime statistics for the cluster and by server.
- Query Monitor view lists queries executing, waiting to execute, and blocked by locks held by other queries.
- Query Detail view shows query metrics, query text, and the execution plan for the query.
- Workload Management view allows administrators to:
  - ◊ Create and manage workloads to manage concurrency and allocate CPU and memory resources.
  - ◊ Change default resource groups for Greenplum Database roles.
  - ◊ Create assignment rules to assign transactions to resource groups.
- Four permission levels allow users to view or cancel their own or others' queries, and to view or manage administrative information.
- Cluster Metrics view shows synchronized charts of historical system metrics.
- History view lists completed queries and system metrics plotted over a selected time period.
- Permissions view to see or manage Command Center permission levels.
- Authentication view to see or edit the `pg_hba.conf` host-based authentication configuration file.
- Segment Status view with summaries and details by segment.



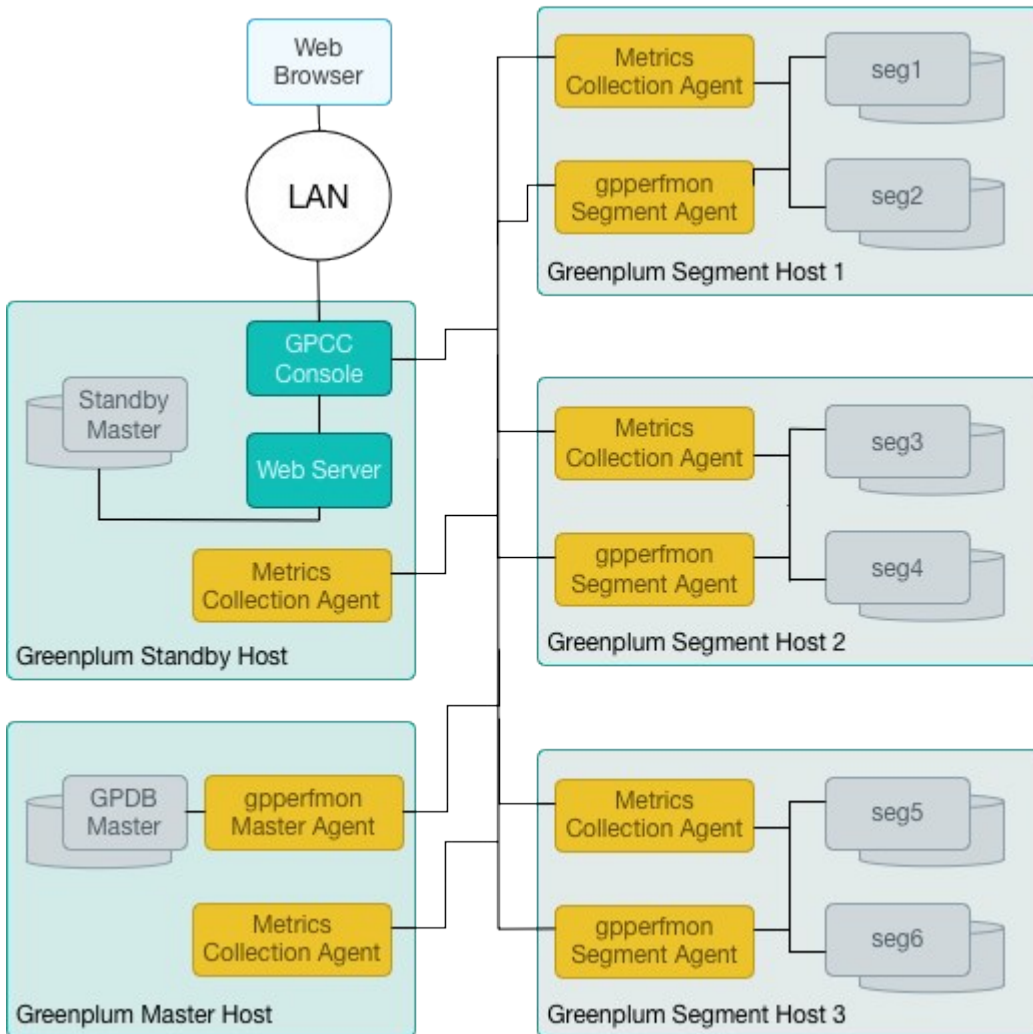
- Storage Status view with summaries and details by segment data directory.



**Note:** Command Center versions 6.4/4.12.0 and later do not run on the Microsoft IE browser.

## Greenplum Command Center Architecture

The following figure illustrates the Greenplum Command Center architecture.



## Greenplum Command Center Web Server and Web Application

The Greenplum Command Center web server and backend application can run on the master or standby master host—the master host is recommended. The web server, `gpccws`, is a custom HTTP server designed for Command Center. The web application is an HTML5 and Go language application.

The Command Center web server authenticates users with the Greenplum Database authentication system. Administrators can edit the Greenplum Database host-based authentication file, `pg_hba.conf`, in the Command Center Console. Command Center can also be configured to authenticate users in a Kerberos environment.

Command Center defines four user authorization levels to manage users' access to the Query Monitor, and to administrative information and operations. User authorization is managed in the Administrative area of the Command Center user interface.

Greenplum Command Center displays information derived from several sources:

- Greenplum Database performance monitoring database (gpperfmon)
- Operating system process accounting
- Greenplum Database system catalog tables
- Real-time query metrics collection extension
- Workload management extension

Greenplum Database is instrumented to enable capturing performance metrics and tracking query execution. The performance monitoring database and the query metrics collection extension deploy agents—processes running on each host to collect metrics. The gpperfmon agents forward collected data to an agent on the Greenplum Database master. The real-time query metrics agents submit collected data directly to the Command Center rpc port. The agents also collect data from the host operating system so that query performance can be correlated with CPU and memory utilization and disk space can be monitored in Command Center.

## Greenplum Database Performance Monitoring Database

The gpperfmon performance monitoring database stores current and historical query status and system information collected from agents running on the master and segment hosts. Greenplum Command Center uses gpperfmon for historical data only; it uses the real-time query metrics to monitor active and queued queries. Greenplum Database sends UDP packets at various points during query execution. The `gpsmon` process on each segment host collects the data. Periodically, every 15 seconds by default, a `gpmmmon` agent on the master host signals the `gpsmon` process to forward the collected data. The agent on the master host receives the data and adds it to the gpperfmon database.

The Command Center database consists of three sets of tables:

- *now* tables store data on current system metrics such as active queries
- *history* tables store data on historical metrics
- *tail* tables are for data in transition. Tail tables are for internal use only and should not be queried by users.

The now and tail data are stored as text files on the master host file system, and the Command Center database accesses them via external tables. The history tables are regular database tables stored within the gpperfmon database.

You can run SQL queries on the data stored in the gpperfmon database. Greenplum Command Center runs queries on the database for information presented in the Command Center Console. The *Greenplum Database Reference Guide* contains references for the tables in the gpperfmon database.

Greenplum Database provides a management utility, `gpperfmon_install`, to create the gpperfmon database and enable the gpperfmon agents on the master and segment hosts. Creating the gpperfmon database is a prerequisite for installing Greenplum Command Center. See the

*Greenplum Database Utility Guide* for details of running the `gpperfmon_install` management utility.

## Real-Time Query Metrics Collection

The data collected by real-time query metrics collection is more detailed and more current than statistics recorded in the `gpperfmon` database. Command Center users can observe queries as they execute and, with sufficient permissions, cancel problem queries to allow other queries to complete.

The Greenplum Database query metrics extension and the metrics collection agent work together to collect real-time metrics and update the Command Center application.

Greenplum Database calls the query metrics extension when a query is first submitted, when a query's status changes, and when a node in the query execution plan initializes, starts, or finishes. The query metrics extension sends metrics to the metrics collection agent running on each segment host. The extension also collects information about the locks queries hold so that you can see which queries hold locks that block other queries. The agent posts the metrics to the Greenplum Command Center rpc port.

The `metrics_collection` extension is included with Tanzu Greenplum Database. The extension is enabled by setting the `gp_enable_query_metrics` server configuration parameter to on and restarting the Greenplum Database cluster. The metrics collection agent is installed on each host when you install Greenplum Command Center. The Command Center application monitors the agent and restarts it if needed.

## Command Center Workload Management

Workloads set concurrency, memory, and CPU resource limits for database transactions they manage. A Greenplum Command Center workload corresponds to a Greenplum Database resource group, but adds additional capabilities that are not available with resource groups.

Command Center allows administrators greater flexibility in assigning transactions to workloads. Every Greenplum Database role is assigned to a single resource group and, by default, transactions are managed by the role's resource group. With Command Center workload management, administrators can define criteria to assign transactions to workloads based on attributes other than the role submitting the transaction. Currently, assignment criteria can evaluate query tags and roles in combination with query tags.

A *query tag* is a key-value pair defined in the `gpcc.query_tags` parameter of a database session. The parameter has the format `<tag1>=<value1>;<tag2>=<value2>`, where tags and values are user-defined values. For example, if you want to run ETL operations in a workload named "etl", you could define a tag named "xact-type" and set it to "etl": `xact-type=etl`. The `gpcc.query_tags` parameter can be set as a connection parameter on Greenplum Database clients that allow it, or with a `SET` command inside the session after the connection has been established, for example `SET gpcc.query_tags='xact-type=etl'`.

The `gp_wlm` extension in Tanzu Greenplum Database provides support for Command Center workloads. Initially, Greenplum Database uses resource queues to manage resources. Using Command Center workloads requires enabling resource groups in Greenplum Database. Resource groups are based on the Linux control groups (cgroups) service, which must first be enabled in the operating system.

# Tanzu Greenplum Command Center Supported Platforms

For information about Greenplum Command Center end of life, see the [Lifecycle Support Policy](#).

## Supported Operating System Platforms

Greenplum Command Center supports the same operating systems Greenplum Database supports.

**Notes:**

If you use resource groups and workload management on Red Hat or CentOS 6.x, upgrade your kernel to 2.6.32-696 or higher to benefit from improvements in the Linux cgroups service.

Greenplum Command Center workload management features are not supported in SUSE Enterprise Linux 11.

## Supported Greenplum Database Versions

Some versions of Greenplum Command Center can be installed on some newer versions of Greenplum Database, but new features may not be supported and some functions may not display properly in the Command Center interface. For best results, use the most recent supported Command Center version for your Greenplum Database version.

This page lists the versions of Greenplum Database supported with each version of Command Center.

Greenplum Command Center	Greenplum Database
2.5.1	4.3.x
3.3.1	4.3.x - 5.2.0
3.3.3	4.3.x - 5.x
4.0.0	5.7.x
4.1.0	5.7.x, 5.8.x
4.1.1	5.7.x, 5.8.x
4.2.0	5.9.0
4.3.0	5.10.x

Greenplum Command Center	Greenplum Database
4.3.1	5.10.x
4.4.0	5.11.0 - 5.11.3
4.4.1	5.11.0 - 5.11.3
4.4.2	5.12.x - 5.13.x
4.5.0	5.14.x - 5.16.x
4.6.0	5.17.0 - 5.18.x
4.7.0	5.19.0 - 5.23.0
4.8.0	5.19.0 - 5.23.0
4.9.0	5.19.0 - 5.27.0
4.10.0 - 4.13.0	5.19.0 - 5.29.1
4.14.0	5.29.2 - 5.29.5
4.15.0	5.29.6
6.0.0	6.0.0 - 6.2.x
6.1.0	6.0.0 - 6.7.x
6.2.0	6.0.0 - 6.10.x
6.3.0	6.0.0 - 6.11.x
6.3.1	6.0.0 - 6.14.x
6.4.0	6.0.0 - 6.16.x
6.5.0	6.0.0 - 6.18.x
6.6.0	6.0.0 - 6.19.x
6.7.0	6.0.0 - 6.21.x

# Installing Greenplum Command Center

- [Installing and Upgrading Greenplum Command Center](#)
- [Securing the gpmon Database User](#)
- [Setting the Command Center Environment](#)
- [Uninstalling Greenplum Command Center](#)

## Installing and Upgrading Tanzu Greenplum Command Center

The Tanzu Greenplum Command Center installation utility installs the Command Center files on all hosts in the Greenplum Database cluster.



**Note:** Run the Greenplum Command Center installer on the Greenplum Database master host. The installer installs the Command Center software on every host in your Greenplum Database cluster. It retrieves the list of hosts in your Greenplum Database cluster from the `gp_segment_configuration` system table.

After you have run the installer you can start Greenplum Command Center on the master host (recommended) or on the standby master host.

## Prerequisites

Before installing or upgrading Greenplum Command Center, ensure the following requirements are met:

- Greenplum Database must be installed and running. See the Tanzu Greenplum Command Center release notes for compatible Greenplum Database versions.
- The Greenplum Database `MASTER_DATA_DIRECTORY` environment variable must be set.
- The directory where Greenplum Command Center will be installed, `/usr/local/` by default, must be writable by the `gpadmin` user on all Greenplum Database hosts. See [Selecting and Preparing an Installation Directory for Command Center](#).
- Port 28080 (default) must be open to TCP connections from Web clients to the HTTP server on the master and standby master hosts. Greenplum Command Center web browser clients connect to this port to access the Command Center Console. Browser connections use HTTP/HTTPS and WebSocket (WS)/Secure WebSocket (WSS) protocols. A different port number can be specified when Command Center is installed. To access the Command Center web server through a proxy, the proxy must have WebSocket support.
- Port 8899 must be open on all hosts in the Greenplum Database cluster for TCP

connections. This is an RPC port, used by the metrics collection agents on the segment hosts to send metrics to the backend.

- Command Center requires the Apache Portable Runtime Utility library. The library is no longer included in the Greenplum Database lib directory, but it is installed as a dependency if you install the Greenplum Database distribution with `yum` or `apt`. Run the command `yum install apr-util` or `apt install libapr1` if you need to install the `apr-util` library.
- If you choose to have Command Center server support SSL/TLS encryption for browser connections -- the default behavior -- you need a combined SSL certificate file containing the server certificate and private key. See [SSL/TLS Encryption](#) for more information.
- Reserve the Command Center webserver port by setting the `net.ipv4.ip_local_reserved_ports` parameter in the `/etc/sysctl.conf` file to the Command Center's web server port, which is 28080 by default.

## Selecting and Preparing an Installation Directory for Command Center

The Command Center installation directory (default `/usr/local`) must exist and be writable by the `gpadmin` user on every host in the Greenplum Database cluster. The Command Center installer creates a directory named `greenplum-cc-<version>` in this directory on every host. When Command Center installation is complete the `greenplum-cc-<version>` directory and all of its contents must be owned by the `gpadmin` user. If Command Center creates the `greenplum-cc-<version>` directory, it also creates a `greenplum-cc` symbolic link to it, providing the `gpadmin` user has permission to create the link. You can use the `greenplum-cc` link in your shell startup script to source the `gpcc_path.sh` file.

In a standard Linux system, the `/usr/local` directory is owned by root and is only writable by root. If you choose the default installation directory or another directory where `gpadmin` does not have write permission, the root user can create the Command Center directory and the symbolic link in the installation directory on each host and make `gpadmin` the owner. Then the `gpadmin` user can run the installer without error.

If the `gpadmin` user has sudo access, you can use the `gpssh` utility to create the directory and set the owner to `gpadmin` on all hosts with these commands:

```
$ source /usr/local/greenplum-db-<version>/greenplum_path.sh
$ gpssh -f <hostfile> 'sudo mkdir -p /usr/local/greenplum-cc-6.3.0; sudo chown -R gpadmin:greenplum /usr/local/greenplum-cc-6.3.0'
$ gpssh -f <hostfile> 'sudo ln -s /usr/local/greenplum-cc-6.3.0 /usr/local/greenplum-cc'
```

The `<hostfile>` text file contains a list of all Greenplum host names, including the master, standby master, and segment hosts.

See the *Tanzu Greenplum Database Installation Guide* for information about setting up passwordless SSH.

## Install the Greenplum Command Center Software

Run the Greenplum Command Center installer on the Greenplum Database master host as the

gpadmin user. The installer copies the software to all other hosts in the cluster.

1. Download the Greenplum Command Center distribution file for your Greenplum Database version from [Tanzu Network](#) and copy it to the gpadmin user's home directory on the master host.
2. Extract the installer from the zip file.

```
$ unzip greenplum-cc-web-gp6-<version>-<platform>.zip
```

Extracting the installer creates a `greenplum-web-<version>` directory containing the `gpccinstall-<version>` installation utility.

There are four ways to run the Greenplum Command Center installer:

- **Interactive** – the installer prompts you for the installation parameters.
- **Scripted** – you run the installer with a configuration file containing installation parameters.
- **Upgrade** – the installer uses the installation parameters from the current Command Center installation.
- **Auto** – the installer uses default installation parameters.



**Note:** The Command Center installer uses HTTPS as the default protocol for the Command Center web server. If you choose HTTP instead, Command Center issues a warning message. The only exception is if you install with the `-auto` option, for which the default protocol is HTTP.

## Interactive Installation

1. Source the `greenplum_path.sh` script in the Greenplum Database installation directory to ensure the `GPHOME` environment variable is set.

```
$ source /usr/local/greenplum-db/greenplum_path.sh
```

2. Run the Greenplum Command Center installer.

```
$ cd greenplum-cc-<version>
$ ./gpccinstall-<version>
```

You can add the following options to the `gpccinstall` command-line.

- ✦ The `-w` option instructs the installer to prompt for the gpmon database user's password.
  - If the gpmon user does not yet exist, the installer creates it using the password you enter at the prompt. The password is not saved anywhere, so be sure to remember the password you enter.
  - If the gpmon user already exists, the installer uses the password you enter at the prompt for the gpmon user. It ignores any password set in the `PGPASSWORD` environment variable, or in the `.pgpass` file in the `gpadmin` user's



home directory (or in the file specified in the `PGPASSFILE` environment variable).

- If the `gpmon` user already exists, but you do not specify the `-w` option, the installer uses the password set in the `PGPASSWORD` environment variable or in the `.pgpass` file.
- If the `gpmon` user does not yet exist and you do not specify `-w` (or if you specify the `-w` option but enter no password at the prompt) the installer creates the `gpmon` user with a default password. See the `.pgpass` file to find the password.
- ♦ The `-ssh_path` option allows you to specify the full path to a custom SSH program. If you do not supply this option, the installer uses the `ssh` command on the path.  
Example:

```
$ ./gpccinstall-<version> --ssh_path /usr/local/bin/ssh -w
```

- ♦ The `-ipv6` option installs a version of Command Center with support for running on IPv6 networks. To enable IPv6, you must set the `HTTPAddr` configuration parameter in the `$GPCC_HOME/conf/app.conf` file to the IPv6 address of the host that runs Command Center.
- ♦ The `-krbenable` option includes prompts to configure Command Center Kerberos authentication. The Kerberos prompts are omitted from the installation dialog by default.

### 3. Where would you like to install Greenplum Command Center? (Default=/usr/local)

Press Enter to accept the default or enter the desired path. The directory must exist on all hosts in the Greenplum Database cluster and must be writable by the `gpadmin` user. See [Selecting and Preparing an Installation Directory for Command Center](#) for more information.

### 4. What would you like to name this installation of Greenplum Command Center? (Default=gpcc)

Enter a name to display on Command Center web pages to identify this Greenplum Command Center installation.

### 5. What port would you like the gpcc webserver to use? (Default=28080)

The default Command Center listen port is 28080. Press Enter to accept the default or enter another port number.

### 6. (Optional). These prompts appear only if you specify `-krbenable` on the `gpccinstall` command line.

#### Enter webserver name for this instance: (Default=<gpcc-host>)

The Kerberos keytab file must contain a principal for the Command Center web server. The web server principal name has the the format `HTTP/<gpcc-host>@<realm>`, where `<gpcc-host>` is the host name clients use in URLs when connecting to the Command Center web server.

**Enter the name of the Kerberos service: (Default=postgres)**

The default Kerberos service name for Greenplum Database is `postgres`.

**Choose Kerberos mode (1.normal/2.strict/3.gpmon\_only): (Default=1)**

Greenplum Command Center supports three different Kerberos authentication schemes.

**1 - normal mode** (default) – The Command Center Kerberos keytab file contains the Command Center principal and may contain principals for Command Center users. If the principal in the client's connection request is in the Command Center's keytab file, Command Center uses the client's principal for database connections. Otherwise, Command Center uses the gpmon user for database connections.

**2 - strict mode** – Command Center has a Kerberos keytab file containing the Command Center service principal and a principal for every Command Center user. If the principal in the client's connection request is in the keytab file, the web server grants the client access and the web server connects to Greenplum Database using the client's principal name. If the principal is not in the keytab file, the connection request fails.

**3 - gpmon\_only mode** – Command Center uses the gpmon database role for all Greenplum Database connections. No client principals are needed in the Command Center's keytab file.

**Enter path to the keytab file**

Enter the path to the keytab file containing the Kerberos principal for the Command Center web server and, optionally, Command Center user principals.

**What is the URL of the Command Center web server?**

The Kerberos keytab file must contain a principal for the Command Center web server. The principal name is in the format `HTTP/<gpcc-host>@<realm>`, where `<gpcc-host>` is the host name clients use in URLs when connecting to the Command Center web server.

**7. Would you like to enable SSL?**

Enter `y` if you want to enable SSL/TLS (HTTPS) encryption for client connections to the Command Center web server. The installation utility prompts for the location of the combined SSL certificate file.

**Enter the full path of the certificate file, including file name**

Enter the path to the combined SSL certificate file installed on the Command Center host. This file contains a certificate and a private key for the web server. The file must be readable by the gpadmin user. See [SSL/TLS Encryption](#) for information about creating this file and installing it on your server.

**8. Choose a display language (Default=English)**

1. English
2. Chinese
3. Korean
4. Russian
5. Japanese

Enter a number to choose a language for the Command Center user interface.

The installer saves a log of the installation session in the current directory in a file named `gpccinstall.<timestamp>.log`.

## Update the Metrics Collector Extension

If you are installing Command Center into a Greenplum Database version released before the Command Center version you are installing, the installer prompts you to update the `metrics_collector` extension. It writes a README file, `update-extension.txt`, in the new Command Center installation directory, which lists the steps to install the new extension.

To access new Command Center features you must do the following:

1. Remove the current `metrics_collector` extension.
2. Install the new extension.
3. Restart Greenplum Database.

If a new Greenplum Database `metrics_collector` extension is required, the installer displays the following message:

```
*****
*
* INSTALLATION IS ALMOST COMPLETED
*
* The last step is to update the metrics_collector extension, which needs to
* restart the Greenplum Database cluster. Please proceed to $GPCC_HOME and
* follow the instructions in this file:
*
*                               update-extension.txt
*
* Some new features may not be available before the update is done.
*
* To use GPCC with the old metrics_collector extension, source gpcc_path.sh
* and run 'gpcc start'.
*
*****
```

If you are unable to restart Greenplum Database at this time, you can source the `gpcc_path.sh` file in the newly installed Command Center directory and then restart Command Center. New Command Center features may not be available until you update the metrics collector extension and restart Greenplum Database.

**IMPORTANT:** When updating GPDB to a newer version, DO NOT use `gppkg --migrate` to migrate the previous `metrics_collector`. If you have to use `gppkg -- migrate` for some other packages, first run `gppkg -r` to remove `metrics_collector` from the old GPDB installation.

Follow these steps to upgrade the `metrics_collector` extension.

1. Source the `gpcc_path.sh` file in the new Command Center installation directory.

```
$ source <install-dir>/greenplum-cc/gpcc_path.sh
```

2. Change to the new Command Center home directory.

```
$ cd $GPCC_HOME
```

3. View and follow the instructions in the `update-extension.txt` file.

```
$ cat update-extension.txt
```

```
Please update metrics_collector extension with the following steps.
Skip if you just reinstalled GPCC without any changes to GPDB binary.
For more information please check the documentation.

#####
#                                     #
#   I M P O R T A N T   N O T I C E   #
#                                     #
#####

When upgrading GPDB to a newer version,
DO NOT use "gppkg --migrate" to migrate the previous metrics_collector.

If you have to use "gppkg -- migrate" for some other packages,
first run "gppkg -r" to remove metrics_collector from the old GPDB
installation.

Before you proceed with updating the metrics_collector extension,
be sure to back up these original files, in case you decide to roll back:

$GPHOME/lib/postgresql/metrics_collector.so
$GPHOME/share/postgresql/extension/metrics_collector*
$GPHOME/share/postgresql/extension/gp_wlm*

1. cd /usr/local/greenplum-cc-6.3.0/gppkg
2. Stop GPCC: gpcc stop
3. Drop extension: psql gpperfmon -c 'drop extension metrics_collector'
4. Use "gppkg -q --all" to list packages installed.
5. Remove the old version of the metrics_collector: "gppkg -r <name>-<version>"
6. Install the new version: gppkg -i MetricsCollector-6.3.0_gp_6.7.0-<OS>-<ARCH>.gppkg
7. Restart GPDB
8. Restart GPCC: gpcc start
```

## Install With a Configuration File

You can provide a configuration file to the Greenplum Command Center installer to perform a non-interactive Command Center installation.

```
$ cd greenplum-cc-<version>
$ ./gpccinstall-<version> -c <config-file>
```

The following table contains the names of the parameters corresponding to the interactive installation prompts and their default values. Define parameters in a configuration file for any parameters that have no default value or to override default values.

Installer Prompt	Default	Parameter
Where would you like to install Greenplum Command Center?	/usr/local	path

Installer Prompt	Default	Parameter
What would you like to name this installation of Greenplum Command Center?	gpcc	display_name
What port would you like the gpcc webserver to use?	28080	web_port
Would you like to enable Kerberos?	false	enable_kerberos
Choose Kerberos mode (1.normal/2.strict/3.gpmon_only):	1	krb_mode
Please provide the path to the keytab file:		keytab
What is the name of the GPDB Kerberos service?	postgres	krb_service_name
What is the URL of the Command Center web server?	gpcc	webserver_url
Would you like to enable SSL?	true	enable_ssl
Please provide the file path for the SSL certificate:	/etc/ssl/certs/cert.pem	ssl_cert_file
Please choose a display language (1.English/2.Chinese/3.Korean/4.Russian/5.Japanese)	1	language

If the `enable_kerberos` parameter is true, the `keytab`, `webserver_url`, `krb_mode`, and `krb_service_name` parameters must also be set.

If the `enable_ssl` parameter is true, the `ssl_cert_file` parameter is required.

The following installation configuration file example sets all parameters to their default values.

```
path = /usr/local
# Set the display_name param to the string to display in the GPCC UI.
# The default is "gpcc"
# display_name = gpcc

master_port = 5432
web_port = 28080
rpc_port = 8899
enable_ssl = false
# Uncomment and set the ssl_cert_file if you set enable_ssl to true.
# ssl_cert_file = /etc/certs/mycert
enable_kerberos = false
# Uncomment and set the following parameters if you set enable_kerberos to true.
# webserver_url = <webserver_service_url>
# krb_mode = 1
# keytab = <path_to_keytab>
# krb_service_name = postgres
# User interface language: 1=English, 2=Chinese, 3=Korean, 4=Russian, 5=Japanese
language = 1
```

## Non-Interactive Installation with Defaults

The non-interactive installation is useful when installing Command Center in a cloud environment.

1. Source the `greenplum_path.sh` script in the Greenplum Database installation directory to

ensure the `GPHOME` environment variable is set.

```
$ source /usr/local/greenplum-db/greenplum_path.sh
```

2. Run the Greenplum Command Center installer with the `-auto` option.

```
$ cd greenplum-cc-<version>
$ ./gpccinstall-<version> -auto
```



**Note:** When you run the installer with the `-auto` option, the Command Center installer uses HTTP as the default protocol for the Command Center web server. To override this, you must use the `-c` option to pass in a configuration file that has `enable_ssl` set to true and `ssl_cert_file` set to the cert file path.

## Upgrade

Before you upgrade Command Center, you must have installed the new Command Center software. See [Install the Greenplum Command Center Software](#).

Running `gpccinstall` with the `-u` option installs a Greenplum Command Center release using the configuration parameters from the current Command Center installation. You can install a new Command Center release, or reinstall the current release. This option is also useful after you have added new hosts to the Greenplum Database cluster or replaced failed hosts.

The configuration parameters are read from the `$GPCC_HOME/conf/app.conf` file.

1. Source the `greenplum_path.sh` script in the Greenplum Database installation directory to ensure the `GPHOME` environment variable is set.

```
$ source /usr/local/greenplum-db/greenplum_path.sh
```

2. Source the `gpcc_path.sh` script in the Greenplum Command Center installation directory.

```
$ source /usr/local/greenplum-cc-<version>/gpcc_path.sh
```

3. Run the Greenplum Command Center installer with the `-u` option.

```
$ cd greenplum-cc-<version>
$ ./gpccinstall-<version> -u
```

After the upgrade has completed, the Command Center installer may prompt you to update the Greenplum Database metrics collector extension. See [Update the Metrics Collector Extension](#) for steps to complete this task.

## Changed Location for the `gpmetrics` Directory Starting in Command Center 6.8

In Command Center 6.8, the `gpmetrics` directory default location changed from `$MASTER_DATA_DIRECTORY` to `$HOME`.

If you are upgrading from Command Center 6.7.x or earlier and a directory called `gpmetrics` **does not** already exist under `$HOME`, Command Center will migrate the `gpcc.conf` and `gpcc-alert.yaml` files from `$MASTER_DATA_DIRECTORY/gpmetrics` directory to `$HOME/gpmetrics`.

If a directory called `gpmetrics` **does** already exist under `$HOME`, you must copy the `gpcc.conf` and `gpcc-alert.yaml` files over manually before starting Command Center 6.8.0 or later for the first time.

**NOTE:** To specify a default location different than `$HOME` for the `gpmetrics` directory, configure the `gpmetrics_home` parameter in the `$GPCC_HOME/conf/app.conf` file. For more information, see the [Command Center Console Parameters](#) topic.

## Installing Multiple Command Center Instances

You can install a Command Center instance for each Greenplum Database system installed on your cluster. This feature requires Greenplum Database 6.8 or above.

Follow these steps when installing an additional Command Center instance.

1. Source the `greenplum_path.sh` file in the target Greenplum Database system before you start the Command Center installer.
2. Run the latest Greenplum Command Center installer for the target Greenplum Database system.
  - ✦ Choose an installation path that is different from any existing Command Center instances.
  - ✦ Choose a name for the Command Center installation that is different from any existing Command Center instances.
  - ✦ Choose a Command Center web server port number that is different from any existing Command Center instances.
3. When the installation has finished source the `gpcc_path.sh` file in the new Command Center installation directory.
4. Edit the `$GPCC_HOME/conf/app.conf` file.
  - ✦ Change the `rpc_port` parameter from the default value, 8899, to a different, available port number.
  - ✦ Change the `ws_perf_port` parameter from the default value, 6162, to a different, available port number.
  - ✦ Change the `agent_perf_port` parameter from the default value, 6163, to a different, available port number.

Before you start or stop a Command Center instance, be sure to source the `greenplum_path.sh` file in the Greenplum Database installation directory and the `gpcc_path.sh` file in the corresponding Command Center installation directory. Then you can run the `gpcc start`, `gpcc stop`, or other `gpcc` commands.

## Prepare the Standby Master Host

After the Command Center software is installed, you start the Command Center web server and

backend on the master host or the standby master host. Running Command Center on the master host is recommended, but preparing the standby host ensures the standby can run Command Center if the master fails.

1. Copy the `.pgpass` file from the master host to the standby master host. Run these commands from the master:

```
$ ssh gpadmin@<standby_host>
$ scp gpadmin@<master_host>:~/.pgpass ~
$ chmod 600 ~/.pgpass
```



**Note:** There are two alternative methods you can use to supply the gpmon user's password so that you do not have to put the `.pgpass` file on the host. Command Center only requires the gpmon password when you run `gpcc start`, `gpcc stop`, or `gpcc status`.

1. Set the `PGPASSWORD` environment variable before you run `gpcc` commands. Example:

```
$ PGPASSWORD=changeme gpcc status
```

2. Add the `-W` option to `gpcc` commands to have the command prompt you for the password. Example:

```
$ gpcc start -W
```

2. If the Greenplum Command Center web server is to support TLS/SSL, a server certificate `.pem` file must be obtained and installed on the standby host in a location readable by the `gpadmin` user. The default path is `/etc/ssl/certs/cert.pem`.
3. If Greenplum Command Center is to support Kerberos authentication, Greenplum Database must have Kerberos authentication set up and the required principals and keytabs must be installed on the standby host. See [Enabling Authentication with Kerberos](#) for Kerberos setup instructions. You can install Command Center without Kerberos authentication initially and then enable Kerberos later by running the `gpcc krbenable` command.

For more details on starting Command Center on the standby master host, see the section [Restarting Command Center After Master Failure](#).

## Next Steps

- [Securing the gpmon Database User](#)
- [Setting the Greenplum Command Center Environment](#)
- [Starting and Stopping Greenplum Command Center](#)
- [Connecting to Greenplum Command Center](#)

## Securing the gpmon Database User



The Greenplum Database gpmon role is a superuser role used to manage the gpperfmon database and to retrieve other information from Greenplum databases used in the Command Center interface. The Command Center installer creates the gpperfmon database and the gpmon database role if they do not already exist.

## Authentication for the gpmon Role

If the `$MASTER_DATA_DIRECTORY/pg_hba.conf` authentication file contains no entries for the gpmon role, the installer adds these entries:

local	gpperfmon	gpmon		md5
host	all	gpmon	127.0.0.1/28	md5
host	all	gpmon	:::1/128	md5
host	all	gpmon	samenet	md5

If the `pg_hba.conf` file already has one or more gpmon entries, the installer will add the `samenet` entry if it is not already present. If `samenet` does not work with your network configuration, you must manually edit the `pg_hba.conf` file to allow gpmon access from all hosts in the cluster.

The Command Center Table Browser requires the `all` entry in the database field to allow gpmon to retrieve table metadata and statistics from each database.

If you use a method other than `md5` to authenticate the gpmon role, such as LDAP or Kerberos, edit the `pg_hba.conf` file manually to enable gpmon connections from all hosts and access to all databases.

If any of these entries are missing from `pg_hba.conf` Command Center may fail to start, or some features may not work properly.

Administrators can view and update the `pg_hba.conf` file in Command Center on the **Admin> Authentication** page. If you update the `pg_hba.conf` outside of Command Center, be sure to use the `gpstop -u` command to have Greenplum Database reload the file.

```
$ gpstop -u
```

## Setting the gpmon Password

If you run the `gpccinstall` command with the `-w` option, it prompts you to enter a password for the gpmon role. If you do not supply the `-w` option, `gpccinstall` uses a default password. It adds the password to the `.pgpass` file in the gpadmin user's home directory. The entry in the `.pgpass` file is similar to the following:

```
*:5432:gpperfmon:gpmon:changeme
```

See [The Password File](#) in the PostgreSQL documentation for details about the `.pgpass` file.

The `.pgpass` file is required on the Greenplum Database master host to start the gpperfmon data collection agents. If you run Greenplum Command Center on the standby master host, you can copy the `.pgpass` file to that host, or you can run the Command Center `gpcc` management utility with the `-w` option to request password entry each time you start or stop Command Center or request status.

To change the gpmon password, follow these steps:

1. Log in to Greenplum Database as a superuser and change the gpmon password with the `ALTER ROLE` command:

```
# ALTER ROLE gpmon WITH ENCRYPTED PASSWORD 'new_password';
```

2. On the Greenplum master host, update the password in the `.pgpass` file in the gadmin home directory (`~/.pgpass`). Replace the existing password in the line or lines for gpmon with the new password.

```
*:5432:gpperfmon:gpmon:new_password
```

3. Ensure that the `.pgpass` file is owned by gadmin and RW-accessible by gadmin only.

```
$ chown gadmin:gadmin ~/.pgpass
$ chmod 600 ~/.pgpass
```

4. Restart Greenplum Command Center with the `gpcc` utility.

```
$ gpcc stop
$ gpcc start
```



**Note:** Be sure to also update the `.pgpass` file on the standby master host.

## Authenticating gpmon with Kerberos

If you authenticate Greenplum Database and Command Center users with Kerberos, you can also authenticate the gpmon user with Kerberos.

To prepare for installing Command Center with Kerberos authentication, follow these steps:

1. Create the gpperfmon database using the Greenplum Database `gpperfmon-install` management utility.
2. On the KDC, create a keytab file containing the Kerberos principal for the gpmon user, just as you would for any Kerberos-authenticated client. Install the file on the Greenplum master and standby hosts.
3. Update the entries for gpmon in the `$MASTER_DATA_DIRECTORY/pg_hba.conf` file to use the `gss` authentication method.

```
host all gpmon 0.0.0.0/0 gss include_realm=0 krb_realm=GPDB.EXAMPLE.COM
```

Note that `local` entries in `pg_hba.conf` cannot be authenticated with Kerberos. If there is a `local` entry for the gpmon user, it will use the `.pgpass` file to authenticate with the database. See [The pg\\_hba.conf file](#) in the PostgreSQL documentation for complete `pg_hba.conf` file documentation.

4. Log in to the master host as gadmin and authenticate the gpmon user.

```
$ kinit gpmon
```

5. Install Greenplum Command Center to set up the Kerberos-enabled Command Center.

## Setting the Greenplum Command Center Environment

To enable the `gpadmin` user to execute Command Center utilities such as `gpcc` at the command line, source the `gpcc_path.sh` file in the Greenplum Command Center installation directory. For example:

```
$ source /usr/local/greenplum-cc-<version>/gpcc_path.sh
```

The `gpcc_path.sh` script sets the `GPCC_HOME` environment variable to the Command Center installation directory and adds the `$GPCC_HOME/bin` directory to the path.

To automatically source the `gpcc_path.sh` each time you log in, add the above source command to your start-up script, for example `~/.bashrc` or `~/.bash_profile`.

## Uninstalling Greenplum Command Center

To uninstall Greenplum Command Center, you must stop both the Command Center Console and disable the data collection agents. Optionally, you may also remove any data associated with Greenplum Command Center by removing your Command Center Console installation and the `gpperfmon` database.

1. Stop Command Center Console if it is currently running. For example:

```
$ gpcc --stop
```

2. Remove the Command Center installation directory from all hosts. For example:

```
$ rm /usr/local/greenplum-cc ## removes the symbolic link
$ rm -rf /usr/local/greenplum-cc-version
```

3. Disable the data collection agents.

1. Log in to the master host as the Greenplum administrative user (`gpadmin`):

```
$ su - gpadmin
```

2. Disable the data collection agents by setting the `gp_enable_gpperfmon` server configuration parameter off:

```
$ gpconfig -c gp_enable_gpperfmon -v off
```

3. Remove or comment out the `gpmon` entries in `pg_hba.conf`. For example:

```
#local      gpperfmon      gpmon      md5
#host       gpperfmon      gpmon      0.0.0.0/0    md5
```

4. Drop the Command Center superuser role from the database. For example:

```
$ psql template1 -c 'DROP ROLE gpmon;'
```

5. Restart Greenplum Database:

```
$ gpstop -r
```

6. Clean up any uncommitted Command Center data and log files that reside on the master file system:

```
$ rm -rf $MASTER_DATA_DIRECTORY/gpperfmon/data/*  
$ rm -rf $MASTER_DATA_DIRECTORY/gpperfmon/logs/*
```

7. If you do not want to keep your historical Command Center data, drop the gpperfmon database:

```
$ dropdb gpperfmon
```

# Administering the Command Center Web Server

The gpccws web server binary and web application files are installed in the `bin` directory of your Greenplum Command Center installation.

## Starting and Stopping the Web Server

Starting the Command Center Web Server runs the gpccws web server, starts the metrics collection agents on the segment servers, and starts a listener on the Command Center rpc port.

You can run the `gpcc` command as the gpadmin user on the master host (recommended) or on the standby host. Starting Command Center on the standby host is not recommended because it can cause heavy network traffic between the Command Center agent on the master host and the backend on the standby host.

To ensure the `gpcc` command is on your path, source the `gpcc_path.sh` file in the Command Center installation directory or add it to the startup script for your command shell. See [Setting the Greenplum Command Center Environment](#) for instructions. The `MASTER_DATA_DIRECTORY` environment variable must be set to the location of the Greenplum Database master data directory.



**Note:** The `gpcc` command uses the gpmon role to connect to Greenplum Database. It looks for the gpmon password in the `PGPASSWORD` environment variable or in the `.pgpass` file in the gpadmin user's home directory. You can instead append the `-W` flag to the `gpcc` commands below to have `gpcc` prompt you to enter the password.

### To start Greenplum Command Center

Log on to the master host or the standby host.

To log on to the standby from the master host:

```
$ ssh <standby-host>
```

Source the Command Center environmental script.

```
$ source /usr/local/greenplum-cc-<version>/gpcc_path.sh
```

Start the Command Center web server and the metrics collection agents.

```
$ gpcc start
Starting the gpcc agents and webserver...
2019/05/07 01:51:03 Agent successfully started on 5/5 hosts
2019/05/07 01:51:03 View Greenplum Command Center at http://mdw:28090
```

## To stop Greenplum Command Center

```
$ gpcc stop
2019/05/07 01:51:55 GPCC webserver and metrics collection agents have been stopped. Use gpcc start to start them again
```

## To check the Greenplum Command Center status

```
$ gpcc status
2019/05/07 01:50:13 GPCC webserver: running
2019/05/07 01:50:14 GPCC agents: 5/5 agents running
```

See the [gpcc](#) reference page for full syntax for the [gpcc](#) command.

## Configuring the Command Center Web Server

The web server configuration file is stored in `$GPCC_HOME/conf/app.conf`. The parameters in this configuration file are set when you install Greenplum Command Center. The installer copies the Command Center installation directory, including this configuration file, to every Greenplum Database host.

See the *Web Server Parameters* section of [Configuration File Reference](#) for a description of the parameters in this file.

You can see a summary of the current configuration using the `gpcc --settings` command.

```
$ gpcc --settings
Install path:          /usr/local
Display Name:         gpcc
GPCC port:            28080
Kerberos:             enabled
Kerberos webserver name: mdw
Kerberos gpdb name:   gpadmin
Kerberos mode:        gpmon_only
Kerberos path:        /home/gpadmin/gpcc-kerberos.keytab
SSL:                  disabled
```

If you modify the file on one host you should copy it to every other host. Be sure to restart the web server after you change the configuration. Rather than modifying the configuration directly, you can just stop Command Center and re-run the `gpccinstall-<version>` installation command. This ensures the configuration is consistent on all hosts.

You can use the `gpcc krbenable` command to add Kerberos authentication to the Command Center configuration. See [Enabling Authentication with Kerberos](#) for details about setting up Kerberos on the Command Center host. The `gpcc krbenable` command prompts for the Kerberos principal names and artifacts and updates the configuration.

The `gpcc krbdisable` command removes Kerberos parameters from the Command Center configuration.

## Restarting Command Center After Master Failure

The Greenplum Database standby master is a *warm* standby—if the primary master fails, the administrator must activate the standby master by running the `gpactivatestandby` management utility.

After Greenplum Database has restarted on the standby master, you can start Command Center on the standby master.

Follow these steps to activate Greenplum Command Center on the standby master host.

1. On the standby master, edit the `app.conf` file in the `$GPCC_HOME` directory.
  - ♦ Change the `master_host` and `master_port` parameters to the standby master host name and port number.
  - ♦ Change the `httpport` and `httpsport` parameters to the Command Center web server port number.
2. Edit the `pg_hba.conf` file on the standby master to allow the `gpmon` user to access all databases and then run `gpstop -u` on the standby host to reload the configuration files.
3. Copy the `$HOME/gpmetrics` directory from the old master host to the standby master host.
4. Copy the `~gpadmin/.pgpass` file from the master to the standby master and make sure the port number is correct in the entry for the `gpmon` user.
5. Restart Command Center on the standby master.

```
$ source <command-center-install-dir>/gpcc_path.sh
$ gpcc start
```

If you perform the initial four steps ahead of time and keep the standby master configuration files up to date when configurations change on the primary master, you can just restart Command Center on the standby master host after Greenplum Database has restarted on the standby master.

## Viewing and Maintaining Web Server Log Files

Web server access and error log messages are written to `$GPCC_HOME/logs/gpccws.log`.

If you experience errors viewing the Greenplum Command Center Console, refer to this file for more information.

To prevent the web server log from growing to excessive size, you can set up log file rotation using `logrotate` or `cronolog`.

# Administering Command Center Agents

The Greenplum Command Center metrics collection agent, `ccagent`, runs on segment hosts and receives real-time metrics emitted by the metrics collection database extension. Each segment host has one `ccagent` process. The metrics collection extension connects to `ccagent` using Unix Domain Sockets (UDS) to transfer metrics from Greenplum Database. Starting Greenplum Command Center with the `gpcc start` command starts the Command Center agent on each segment host. Stopping Command Center with `gpcc stop` ends these processes. The Command Center backend monitors these agents and restarts them when necessary.

If you expand the Greenplum Database cluster with new segment hosts or replace segment hosts, you must reinstall Command Center so that the agent is installed onto the new hosts.

See [Managing Log Files and Data Files](#) for information about managing Command Center agent log files.

**Note:** For information about the `gpperfmon` service agents `gpmmmon` and `gpsmon`, see [The gpperfmon Database](#).



# Administering the gpperfmon Database

Data collected by gpmmmon and gpsmon agents is stored in a dedicated database called gpperfmon. This database requires the typical database maintenance tasks, such as clean up of old historical data and periodic `ANALYZE`.

See the [gpperfmon Database Reference](#) section for a reference of the tables and views in the gpperfmon database.

## Connecting to the gpperfmon Database

Database administrators can connect directly to the gpperfmon database using any Greenplum Database-compatible client program (such as `psql`). For example:

```
$ psql -d gpperfmon -h master_host -p 5432 -U gpadmin
```

## Backing Up and Restoring the gpperfmon Database

The history tables of the gpperfmon database can be backed up and restored using the Greenplum Database backup and restore utilities. See the *Greenplum Database Utility Guide* for more information.

## Maintaining the Historical Data Tables

All of the `*_history` tables stored in the gpperfmon database are partitioned into monthly partitions. A January 2010 partition is created at installation time as a template partition. It can be deleted once some current partitions are created. The Command Center agents automatically create new partitions in two month increments as needed. Administrators must periodically drop partitions for the months that are no longer needed in order to maintain the size of the database.

See the *Greenplum Database Administrator Guide* for more information on dropping partitions of a partitioned table.

# Administering Log Files, Work Files, and Performance Data

This topic describes Command Center logging and log files, work files, and automatic collection of performance data for troubleshooting performance issues with VMware Support.

## Logging and Log Files

The Command Center components—the backend server, agents, HTTP server, and the `gpcc` command-line utility—write messages to log files in the `$GPCC_HOME/logs` directory.

Log files rotate once per day or whenever the component starts or restarts. The current log file is saved to a file with a timestamp in its name and a new log file is started.

The log files can be useful for monitoring and troubleshooting, but you can delete them when you determine they are no longer useful. Remove older log files regularly to conserve disk space.

You can use a utility such as the Linux `logrotate` utility to manage archived log and data files.

### webserver.log

The Command Center backend web server writes to the `webserver.log` log file on the Command Center host in the `$GPCC_HOME/logs` directory on the host running Command Center. This is usually the master host, but could also be the standby master host. Messages in this log file describe start-up events, rpc request/response events with agents running on the Greenplum Database segment hosts, and data loading from external tables to the gpmetrics tables in the gpperfmon database.

### agent.log

A Command Center agent running on each Greenplum Database host (including the segment and master hosts) logs messages to the `agent.log` log file in the `$GPCC_HOME/logs` directory of the host machine. Agents log messages about connections to the Command Center rpc server, periodic data collection events, and metrics transmissions to the rpc port on the backend.

### gpccws.log

The Command Center HTTP server writes messages to the `gpccws.log` file on the Command Center host in the `$GPCC_HOME/logs` directory. The messages in this log file include server start up messages and WebSocket connection events with Command Center clients.

### cli.log

The `gpcc` command-line utility writes messages to the `cli.log` file in the `$GPCC_HOME/logs` directory. Whenever the `gpcc` utility executes, it writes its output to the log file as well as the standard output.

The log file is useful for auditing Command Center start and stop events. The `cli.log` file is not rotated and continues to grow until you archive or delete it manually.

## Command Center Work Files

Command Center saves temporary data into files in the `$GPCC_HOME/ccdata` directory. These files include saved query text files (`q*.txt`), plan node files (`gpccexec*.txt`), and table size and statistics information files (`gpcc_size_info.dat` and `gpcc_stat_info.dat`). Each night, Command Center archives files in the `$GPCC_HOME/ccdata` directory that are more than two weeks old. The archive files have names in the format `archive_YYYYMMDD_YYYYMMDD.tar.gz`, where the dates are the beginning and end dates of the week included in the archive. The archived files are no longer needed by Command Center but may be useful for troubleshooting; you can remove them manually if you do not want to save them.

## Automatic Collection of Performance Data

Command Center automatically collects performance data for the purpose of troubleshooting performance issues, in `*.prof` files in `$GPCC_HOME/logs`. Command Center implements Golang pprof runtime profiling described [here](#).

This feature collects two types of benchmarking profile data: heap memory and CPU. Command Center saves the files hourly and overwrites the data every 24 hours.

Using the properties in the `[pprof]` section of the `gpmetrics` configuration file, you can control:

- whether or not data collection is enabled (it is by default)
- the duration of the profiling interval
- the time duration of CPU profiling in each profiling interval
- the time-to-live (TTL) of profiling logs
- whether Command Center saves the heap memory profile, the CPU profile, or both.

For more details on the `pprof`-related configuration parameters, see the [gpmetrics Configuration File Reference](#).

# Settings

The **Admin> Settings** view enables Command Center administrators to configure settings for Command Center features.

## Query Monitor Guest Access Settings

Turn on **Allow guests to view Query Monitor** to allow users access to the **Query Monitor** without logging in. When enabled, the **Allow guests to view Query Monitor** feature allows anyone with access to the Greenplum Command Center web server to click **View Query Monitor** on the Command Center sign-in screen and see the **Query Monitor** page without logging in. These anonymous users cannot cancel queries and cannot access any other Command Center features.

When this feature is off, the **View Query Monitor** link does not appear on the sign-in screen and anonymous users cannot see the **Query Monitor** page.

Command Center users with Admin permission can toggle the feature on and off with a mouse click. Users with Operator or Operator Basic permission see a message reporting if the feature is on or off.

Settings

Configure settings for various GPCC features

Current Time  
2021-11-10 05:42:07

Query Monitor Guest Access

Provides access to Query Monitor without signin

☒ Allow guests to view Query Monitor

## History Settings

Turn on **Enable GPCC history data collection** to save query history, host metrics, and disk usage metrics to tables in the gpccmon database gpmetrics schema. This option defaults to on for a Greenplum Database 6 installation, but the Greenplum Database metrics collector extension is inactive until Command Center is started.

Enter a number of seconds to set the minimum runtime for a query to be saved in history. The default is to save all queries. Set this threshold to prevent Command Center from filling history with trivial queries.

Settings
Configure settings for various GPCC features
Current Time  
2022-07-20 05:18:48

**Query Monitor Guest Access**  
Provides access to Query Monitor without signin  
☒ Allow guests to view Query Monitor

**History Settings**
EDIT

You can now turn on GPCC agents to collect and store historical usage data in the gpcc\_\*\_history tables. This history collection may be turned on in addition to gpperfmon. [Read more about gpcc\\_\\*\\_history information here.](#)

History Data Collection ⓘ <b>Enabled</b>	Skip Query History Collection ⓘ <b>Not Set</b>
History Data Retention Period <b>Keep it Forever</b>	History Data Purge Time <b>Not Applicable</b>

When you enable GPCC history data collection:

- Command Center saves query and metrics history in the gpmetrics schema tables in the gpperfmon database. Only queries that execute for at least the number of seconds you specify are saved. Query plan node history is only saved for queries that run for at least 10 seconds, or the number of seconds you specify, if greater than 10.

See [gpmetrics Schema Reference](#) for information about the gpmetrics history tables.

When you disable GPCC history data collection:

- Command Center query history views use history data collected by the Greenplum Database gpperfmon agents.



**Note:** For best performance, set **Enable GPCC history data collection** to on and disable gpperfmon data collection. To disable the gpperfmon agents, set the Greenplum Database `gp_enable_gpperfmon` configuration parameter to off and restart Greenplum Database:

```
$ gpconfig -c gp_enable_gpperfmon -v 'off'
$ gpstop -ar
```

## Configure Retention Period For History Data

When History data collection is enabled, you may specify how long Command Center stores historical usage data in the `gpcc_*_history` tables. The default is to keep the data forever.

Using the **History Data Retention Period** dropdown menu, you may choose to have Command Center keep the data for 3 months, 6 months, 12 months, 24 months, or forever. In addition, for values other than Forever, you may configure the time of day at which Command Center purges the data, in the **History Data Purge Time** dropdown menu.

Once you have selected values in each of these two dropdown menus, click Save to save your selection.

# gpcc

Manages the Greenplum Command Center web service and metrics collection agents.

```
gpcc <action> [-W]

gpcc [--version | -v ]

gpcc [--help | -h]

gpcc [--settings]
```

## Actions

### start

Starts the Command Center web service and metrics collection agents. Add the `-w` flag to force a prompt for the gpmon user password.

### stop

Stops the Command Center web service and metrics collection agents. Add the `-w` flag to force a prompt for the gpmon user password.

### status

Displays the status, either `Running` or `Stopped`, of the web server and metrics collection agents. Add the `-w` flag to force a prompt for the gpmon user password.

### krbenable

Enables Kerberos authentication for Command Center.

Use the `gpcc krbenable` command to set up Kerberos authentication for Command Center users if Command Center was initially installed without enabling Kerberos. When you run `gpcc krbenable`, `gpcc` prompts for:

- the web server name
- the name of the Greenplum Database Kerberos service
- the Command Center Kerberos authentication mode
- the path to the keytab file on the Command Center host.

Before you run `gpcc krbenable`, see [Enabling Authentication with Kerberos](#) to check prerequisites and for help preparing the Command Center host to allow Kerberos authentication.

### krbdisable

Disables Kerberos authentication for Command Center.

## help

Displays syntax and help text for the `gpcc` command.

## Options

### `--settings`

Displays the current values of the Command Center configuration parameters. See [Command Center Console Parameters](#) for a list of the configuration parameters.

### `--version` or `-v`

Displays the Greenplum Command Center version.

### `-W <password>`

The optional `-w` option specifies the password for the gpmon user. The `gpcc` command normally gets the password from the `$PGPASSWD` environment variable or the `.pgpass` file in the gpadmin user's home directory. If the password is not available with either of these methods, the `-w` option must be included to specify the password whenever you run `gpcc`.

## Description

The Greenplum Database `MASTER_DATA_DIRECTORY` environment variable must be set when you run the `gpcc` command. This environment variable is usually set in the gpadmin user's shell initialization script (`/home/gpadmin/.bashrc`, for example). If `MASTER_DATA_DIRECTORY` is not set when you run `gpcc start`, `gpcc` prints an error message and exists.

Once started, the Command Center backend monitors the metrics agents with a heartbeat. If a failed agent is detected, the backend spawns a new agent process.

## Examples

Start Command Center and the metrics agents, prompting for the gpmon password.

```
$ gpcc start -W
Password for GPDB user gpmon:
Starting the gpcc agents and webserver...
2018/03/22 17:51:51 Agent successfully started on 7/8 hosts
2018/03/22 17:51:51 View Greenplum Command Center at http://smdw:28080
```

# gpmetrics Schema Reference

Greenplum Command Center creates the `gpmetrics` schema in the Greenplum Database `gpperfmon` to save alert rules and logs, and historical metrics collected by the Greenplum Database metrics collection system. The `gpmetrics` schema contains the following tables and user-defined functions:

## Tables

- `gpcc_alert_rule` — saves alert rules configured on the Command Center **Admin> Alerts** page.
- `gpcc_alert_history` — records an event when an alert rule is triggered.
- `gpcc_database_history` — saves summary query activity information.
- `gpcc_department` — saves department information.
- `gpcc_disk_history` — saves disk usage statistics for each Greenplum Database host file system.
- `gpcc_export_log` — saves the log notifications from every "EXPORT ALL" user action.
- `gpcc_gpss_job` — saves information about data loading jobs.
- `gpcc_gpss_log` — saves logging-related information about data loading jobs.
- `gpcc_pg_log_history` — saves history from the Greenplum Database `pg_log` log file.
- `gpcc_plannode_history` — saves plan node execution statistics for completed queries.
- `gpcc_queries_history` — saves execution metrics for completed Greenplum Database queries.
- `gpcc_queries_now` — saves real-time query metrics data.
- `gpcc_role_department` — saves information about roles within a department.
- `gpcc_resgroup_history` — saves the history of the resource consumption of each resource group on each segment.
- `gpcc_scan_history` — saves history for Recommendations scans.
- `gpcc_schedule` — saves schedule for Recommendations scans.
- `gpcc_system_history` — saves system metrics sampled from Greenplum Database segments hosts.
- `gpcc_table_info` — saves current statistics and size information for tables.
- `gpcc_table_info_history` — saves daily statistics and size information for tables.
- `gpcc_wlm_rule` — saves workload management rules.



- `gpcc_wlm_log_history` — saves the log history of workload management rule actions.

## User-Defined Functions

- `gpcc_delete_department` — deletes a department.
- `gpcc_queries_per_hour` — returns a variety of details about query activity per hour.
- `gpcc_queries_per_user` — returns, for each user, the number of queries whose runtime is longer than the input interval, per hour, in the specified time range.
- `gpcc_queries_per_user_max_and_total_spill_size` — returns, for each user, the total `spill_size` and maximum `spill_size` per query per hour.
- `gpcc_queries_per_user_max_cpu` — returns, for each user, the query with the maximum segment and master cpu usage per hour, along with details about the query.
- `gpcc_queries_per_user_max_run_time` — returns, for each user, the longest running query per hour, along with details about the query.
- `gpcc_queries_per_user_max_skew` — returns, for each user, the query with the maximum amount of processing skew in the system (`skew_cpu`) per hour, along with details about the query.
- `gpcc_queries_per_user_rows_out` — returns, for each user, the query with the maximum `rows_out` per hour, along with details about the query.
- `gpcc_queries_top_10_cpu_per_resgroup` — returns by resource group, the top ten queries utilizing the most CPU.
- `gpcc_queries_top_10_memory_per_resgroup` — returns by resource group, the top ten queries utilizing the most memory.
- `gpcc_recommendations` — returns a list of tables in need of a vacuum-related action such as `VACUUM`, `VACUUM FULL`, `ANALYZE`, and `VACUUM FREEZE`.
- `gpcc_system_per_hour` — returns a variety of system information.
- `gpcc_update_department` — renames a department.

If you set the schema search path to include the `gpmetrics` schema, you do not have to qualify table and user-defined function names with the `gpmetrics` schema name. To set the default search path for the `gpccperfmon` database enter this SQL command.

```
=# ALTER DATABASE gpccperfmon SET search_path TO public,gpmetrics;
```

You must exit the current session and start a new session for the new search path to take effect.

## Alert Tables

Command Center uses the `gpcc_alert_rule` and `gpcc_alert_log` tables to store the alert rules you set up in the user interface and to log messages when the alert rules are triggered.

### `gpcc_alert_rule`

**Note:** Deprecated in Command Center 6.4/4.12.

The `gpcc_alert_rule` table records the alert rules configured in the Command Center user interface. It has the columns shown in the following table.

Column	Type	Description
<code>rule_id</code>	integer	Unique id for the rule.
<code>rule_type</code>	integer	Reserved for future use.
<code>rule_description</code>	character varying(512)	Text of the rule.
<code>rule_config</code>	json	JSON string containing parameters for user-specified values.
<code>ctime</code>	timestamp(0) without time zone	Time the rule was created.
<code>etime</code>	timestamp(0) without time zone	Time the rule became inactive, or null if rule is active.

The `gpcc_alert_rule` table keeps a history of alert rule configurations. When a rule becomes active, a new row is inserted and the `ctime` timestamp column is set to the current time; the `etime` timestamp is null, indicating that the rule is still active. When a rule is either disabled or superceded by a new rule, the `etime` timestamp column is set to the current time. Thus, the set of currently active rules is all rows where the `etime` column is null. A row that has timestamps in both `ctime` and `etime` columns is an historical record of the period of time during which the rule was active.

The `rule_id` column, a unique integer, is the distribution key for the table and is used to identify a single alert rule configuration. This column can be joined with the `rule_id` column in the `gpcc_alert_log` table to identify the rule that triggered each recorded alert event.

The `rule_description` column contains a string that describes the event that matches the rule. It is the text displayed in the Command Center UI for the rule, with user-specified values inserted.

The `rule_config` column contains a JSON string with parameters for the values entered for each of the rule's fields in the Command Center UI.

## gpcc\_alert\_history

The `gpcc_alert_history` table (was `gpcc_alert_log` before Command Center 6.4) has the following columns:

Column	Type	Description
<code>id</code>	integer	Unique ID for the alert.
<code>rule_id</code>	integer	The ID of the rule that triggered this alert.
<code>transaction_time</code>	timestamp(0) without time zone	Time the alert was raised.
<code>content</code>	json	Contains parameters specifying values that triggered the alert.
<code>config</code>	json	Stores alert history from emails or notifications.

Where:

- The `id` column, a unique integer, is the distribution key for the table.
- The `transaction_time` column is set to the current time when a row is created.
- The `rule_id` column can be joined with the `rule_id` column in the `gpcc_alert_rule` table to

access details of the rule that triggered the alert.

- The `content` column contains a JSON string with parameters specifying details about the event that triggered the alert. The JSON parameters vary with the type of the alert.
- The `config` column stores the alert history received from the notification center or from emails.

The `gpcc_alert_history` table is an append-only, column-oriented table, partitioned by month on the `transaction_time` column. Command Center creates new partitions as needed and removes partitions over 12 months old.

A row is added to the `gpcc_alert_history` table whenever an alert rule is matched.

## Greenplum Database Metrics History Tables

The `gpmetrics` query history saves information collected by the Greenplum Database metrics collection system and forwarded to Greenplum Command Center.

The distribution key for each table is a `ctime` timestamp column, which is the time when the row is added to the database. The tables are partitioned by year and month. Greenplum Command Center creates new partitions automatically as needed.

The history tables use append-optimized, column-oriented storage.

Command Center only saves queries with runtimes greater than the value of the `min_query_time` configuration parameter, found in the `$HOME/gpmetrics/gpcc.conf` configuration file on the host executing Command Center. The default, 0, saves all queries in the history table. This parameter can be configured on the Command Center **Admin> Settings** page.

### gpcc\_database\_history

The `gpcc_database_history` table saves summary query activity metrics collected by the Greenplum Database metrics collector. This data can be used to review the Greenplum Database query load over time.

Column	Type	Description
<code>ctime</code>	timestamp(0) without time zone	Time the record was created.
<code>queries_total</code>	integer	Total number of queries running and queued to run.
<code>queries_running</code>	integer	Number of queries currently running.
<code>queries_queued</code>	integer	Number of queries queued, but not yet running.
<code>queries_blocked</code>	integer	The number of queries started, but blocked by other transactions.
<code>queries_finished</code>	integer	The number of queries that completed since the previous sampling interval.

### gpcc\_disk\_history

The `gpcc_disk_history` table saves historical disk usage statistics for each Greenplum Database segment host file system.

Column	Type	Description
ctime	timestamp(0) without time zone	Time the row was created.
hostname	character varying(64)	Name of the segment host.
filesystem	text	Path to the segment's data directory.
total_bytes	bigint	Total size of the file system storage in bytes.
bytes_used	bigint	Number of storage bytes in
bytes_available	bigint	Number of storage bytes available.

## gpcc\_export\_log

The `gpcc_export_log` table saves the log notifications from every "EXPORT ALL" user action. Whenever the user exports search results from the History or the Table Browser pages, this table gets updated.

Column	Type	Modifiers
id	integer	not null default nextval('gpcc_export_log_id_seq'::regclass)
ctime	timestamp(0) with time zone	not null default now()
source	character varying(64)	not null
filename	character varying(64)	not null
role	name	not null
req_params	json	not null
etime	timestamp(0) with time zone	
status	character varying(64)	
fail_msg	text	

## gpcc\_pg\_log\_history

The `gpcc_pg_log_history` table stores `pg_log` errors and warnings from the Greenplum Server log files.

Column	Type	Description
logtime	timestamp without time zone	Timestamp for this log.
loguser	text	Name of the role executing the query.
logdatabase	text	The database accessed.
logpid	text	Process id.
logthread	text	Thread number.

Column	Type	Description
loghost	text	Host name or IP address of the host.
logport	text	Port number.
logsessiontime	timestamp without time zone	Session timestamp.
logtransaction	integer	Transaction id.
logsession	text	Session id.
logcmdcount	text	Command count.
logsegment	text	Segment number.
logslice	text	Slice number.
logdistxact	text	Distributed transaction id.
loglocalxact	text	Location transaction id.
logsubxact	text	Subtransaction id.
logseverity	text	Log severity.
logstate	text	SQL State code associated with this log message.
logmessage	text	Log or error message text.
logdetail	text	Detail message text associated with an error message.
loghint	text	Hint message text associated with an error message.
logquery	text	Internally-generated query text.
logquerypos	integer	Index into the internally-generated query text.
logcontext	text	Context in which this message gets generated.
logdebug	text	Query string with full detail for debugging.
logcursorpos	integer	Cursor index into the query string.
logfunction	text	Function in which this message is generated.
logfile	text	Log file in which this message is generated.
logline	integer	Line in the log file in which this message is generated.
logstack	text	Full text of the stack trace associated with this message.

## gpcc\_plannode\_history

The `gpcc_plannode_history` table saves detailed metrics for each operation (node) in a completed query plan. Each row contains metrics for one operation that executed on one Greenplum Database segment. This information allows reconstructing the plan and execution metrics for a completed query.

Plan node history is only saved for queries that execute for 10 seconds or more.

Column	Type	Description
--------	------	-------------

ctime	timestamp(0) without time zone	Time the row was created.
tmid	integer	A time identifier for the query. All records associated with a query will have the same <code>tmid</code> .
ssid	integer	Session id for the database connection. All records associated with the query will have the same <code>ssid</code> .
ccnt	integer	Command number within the session. All records associated with the query will have the same <code>ccnt</code> .
segid	integer	Id ( <code>dbid</code> ) of the segment for this plan node.
procid	integer	The postgres process ID for this plan node.
sliceid	integer	<del>DEPRECATED. Id of the slice the operation belongs to. Operations that belong to the same slice execute in parallel.</del>
nodeid	integer	The query plan node ID for this operation.
parent_nodeid	integer	The parent query plan node ID from the query plan.
node_type	character varying(64)	Name of the operation type.
tinit	timestamp(6) without time zone	Time the operation was initialized.
tstart	timestamp(6) without time zone	Time the operation started.
tfinish	timestamp(6) without time zone	Time the operation finished.
status	character varying(16)	Status of the operation: <code>initialize</code> , <code>executing</code> , or <code>finished</code> .
planrows	double precision	The number of output rows estimated for the operation.
planwidth	integer	Width of output rows estimated for the operation.
start_cost	double precision	Number of page reads expected to produce first output row.
total_cost	double precision	Number of page reads expected to produce all output rows.
tuple_count	bigint	
ntuples	bigint	
nloops	bigint	
first_tuple	timestamp(6) without time zone	Time the operation produced the first output row.
rel_oid	oid	Object ID of the output rows produced by the operation.
relation_name	character varying(64)	Name of the table this operation processed, if any.

Column	Type	Description
index_name	character varying(64)	Name of the index used by this operation, if any.
alias_name	character varying(64)	Alias for the relation declared in the SQL command.
node_sequence	integer	Node sequence
condition	text	Condition expression used for a filter or join operation.

## gpcc\_queries\_history

The `gpcc_queries_history` table saves metrics for completed queries.

Column	Type	Description
access_tables_info	integer	The table OID of the query accessed.
ctime	timestamp(0) without time zone	Time the row was created.
tmid	integer	A time identifier for the query. All records associated with the query will have the same <code>tmid</code> .
ssid	integer	Session id for the database connection. All records associated with the query will have the same <code>ssid</code> .
ccnt	integer	Command number within this session. All records associated with the query will have the same <code>ccnt</code> .
username	character varying(64)	Role name that issued the query.
db	character varying(64)	Name of the database queried.
cost	double precision	Estimated cost to execute query, computed by the legacy planner or GPORCA.
tsubmit	timestamp(6) without time zone	Time the query was submitted.
tstart	timestamp(6) without time zone	Time the query was started.

Column	Type	Description
tfinish	timestamp(6) without time zone	Time the query finished.
status	character varying(16)	Status of the query – <code>abort</code> , <code>error</code> , or <code>done</code> .
rows_output	bigint	Number of rows returned by the query.
error_message	text	Error message, if the query failed.
plan_name	character varying(16)	<code>PLANNER</code> if query plan was generated by the legacy planner; <code>OPTIMIZER</code> if the plan was generated by GPORCA, the Tanzu query optimizer.
query_hash	character varying(64)	Hash code generated from the text of the query.
query_text	text	Complete text of the query. Some queries may be reformatted before storing in the history database.
application_name	character varying(64)	Name of the client application that established the database connection.
rsname	character varying(64)	If the <code>gp_resource_manager</code> configuration parameter is <code>queue</code> , the name of the resource queue managing the query.
rsgname	character varying(64)	If the <code>gp_resource_manager</code> configuration parameter is <code>group</code> , the name of the resource group managing the query.
cpu_master	bigint	Total CPU usage for this query on the Greenplum Database master instance.
cpu_seconds	bigint	Total CPU usage for this query across all segments, measured in seconds. This is the sum of the CPU usage values taken from all active primary segments in the database array.
cpu_master_percent	double precision	Average CPU percent usage on the master host during execution of this query.
cpu_segment_percent	double precision	Average CPU percent usage on the segment hosts during the execution of this query.
skew_cpu	double precision	Displays the amount of processing skew in the system for this query. Processing/CPU skew occurs when one segment performs a disproportionate amount of processing for a query. The skew is calculated from total CPU seconds used on all segments during the execution of the query.



Column	Type	Description
skew_rows	double precision	Displays the amount of row skew in the system. Row skew occurs when one segment produces a disproportionate number of rows for a query.
memory	bigint	Total size of memory, in kilobytes, used by all segments to execute this query.
disk_read_bytes	bigint	Number of bytes read from disk.
disk_writes	bigint	Number of bytes written to disk.
spill_size	bigint	Total size, in bytes, of spill files used by all segments to execute this query.
rqpriority	character varying(16)	Priority setting for the resource queue managing this query. Blank if resource group management is enabled.
query_tag	text	A key-value pair describing a query.
slices_metrics	JSON	The cpu/memory/disk metrics for a slice.
peak_memory	bigint	Maximum memory usage across all segments during the execution of the query, measured in KB.
node_sliceid	json	A map of node ID to slice ID.

## gpcc\_resgroup\_history

The `gpcc_resgroup_history` table saves the history of the resource consumption of each resource group on each segment.

Column	Type	Description
ctime	timestamp(0) without time zone	Time the row was created.
rsgname	name	Resource group name.
segid	integer	Database ID of the segment for this resource group info.
groupid	oid	Object ID of the resource group.
concurrency_limit	integer	The concurrency ( <code>CONCURRENCY</code> ) value specified for the resource group.
cpu_rate_limit	integer	The CPU limit ( <code>CPU_RATE_LIMIT</code> ) value specified for the resource group, or -1.
num_running	integer	The number of transactions currently executing in the resource group.
num_queueing	integer	The number of currently queued transactions for the resource group.

Column	Type	Description
cpu_usage_percent	double precision	The sum of the percentages of CPU cores that are used by the resource group.
mem_used_mb	integer	Memory used, in MB.
mem_available_mb	integer	Available memory, in MB.
mem_quota_used_mb	integer	Memory quota used, in MB.
mem_quota_available_mb	integer	Available memory quota, in MB.
mem_quota_granted_mb	integer	Granted memory quota, in MB.
mem_quota_proposed_mb	integer	Proposed memory quota, in MB.
mem_shared_used_mb	integer	Shared memory used, in MB.
mem_shared_available_mb	integer	Available shared memory, in MB.
mem_shared_granted_mb	integer	Granted shared memory, in MB.
mem_shared_proposed_mb	integer	Proposed shared memory, in MB.

## gpcc\_scan\_history

Column	Type	Description
ctime	timestamp(0) without time zone	Time the row was created.
scan_id	integer	Scan id for this scan. See <a href="#">schedule_id</a> column in the <a href="#">gpcc_schedule</a> .
status	integer	Scan completion status. 0=Running, 1=Success, 2=Failed, 3=Cancelled, 4=Expired.
tables_scanned	bigint	The number of tables that were scanned.
start_dt	timestamp(0) without time zone	Time the scan began.
end_dt	timestamp(0) without time zone	Time the scan ended.

## gpcc\_schedule

The [gpcc\\_schedule](#) table contains details for scheduled events, such as the table scan for Recommendations. Each row describes a schedule in a JSON value. The [ctime](#) and [etime](#) columns together specify the period of time during which the schedule is enabled.

Column	Type	Description
schedule_id	integer	Id number for this schedule.
description	character varying(512)	Description of schedule.
schedule_config	json	JSON string with schedule details.
ctime	timestamp(0) without time zone	Time the schedule was added and enabled.
etime	timestamp(0) without time zone	Time the schedule was disabled.

## gpcc\_system\_history

The `gpcc_system_history` table saves historical system metrics for each Greenplum Database host, including the master, standby master, and segment hosts. The metrics include information about memory, CPU, disk, and network utilization.

Column	Type	Description
cpu_iowait	double precision	The percentage of CPU used waiting on IO requests.
ctime	timestamp(0) without time zone	Time the row was created.
hostname	character varying(64)	Segment or master hostname associated with these system metrics.
mem_total	bigint	Total system memory in Bytes for this host.
mem_used	bigint	System memory used, in Bytes, for this host.
mem_actual_used	bigint	Actual memory used, in Bytes, for this host (not including the memory reserved for cache and buffers).
mem_actual_free	bigint	Free actual memory, in Bytes, for this host (not including the memory reserved for cache and buffers).
swap_total	bigint	Total swap space in Bytes for this host.
swap_used	bigint	Swap space used, in Bytes, for this host.
swap_page_in	bigint	Number of swap pages in.
swap_page_out	bigint	Number of swap pages out.
cpu_user	double precision	Percentage of time CPU processes execute in user mode.
cpu_sys	double precision	Percentage of time CPU processes execute in system (kernel) mode.
cpu_idle	double precision	Percentage idle CPU.
load0	double precision	CPU one-minute load average.
load1	double precision	CPU five-minute load average.
load2	double precision	CPU fifteen-minute load average.
quantum	seconds	Interval between metrics collections.
disk_ro_rate	bigint	Disk read operations per second.

Column	Type	Description
disk_wo_rate	bigint	Disk write operations per second.
disk_rb_rate	bigint	Bytes per second for disk read operations.
disk_wb_rate	bigint	Bytes per second for disk write operations.
net_rp_rate	bigint	Packets per second on the system network for read operations.
net_wp_rate	bigint	Packets per second on the system network for write operations.
net_rb_rate	bigint	Bytes per second on the system network for read operations.
net_wb_rate	bigint	Bytes per second on the system network for write operations.

## gpcc\_table\_info

The `gpcc_table_info` table stores a current snapshot of statistics for tables. There is one row for each table and partition.

Column	Type	Description
ctime	timestamp without time zone	Time the row was created.
dbid	oid	Object ID of the database.
relid	oid	Object ID of the table.
paroid	oid	Object ID of a partition.
reltablespace	oid	Object ID of the table's tablespace.
seq_scan	bigint	Number of sequential scans initiated on this table.
idx_scan	bigint	Number of index scans initiated on this table.
n_tup_ins	bigint	Number of rows inserted.
n_tup_del	bigint	Number of rows deleted.
n_tup_upd	bigint	Number of rows updated (includes hot updated rows).
n_tup_hot_upd	bigint	Number of rows HOT updated (no separate index update required).
last_seq_scan	timestamp with time zone	Time of the last sequential scan on this table. An <code>ANALYZE</code> on a table can cause a sequential scan and update the count in the <code>seq_scan</code> column. However, the time of that scan is not saved in this column.

Column	Type	Description
last_idx_scan	timestamp with time zone	Time of the last index scan on this table.
last_ins	timestamp with time zone	Time of the last insert on this table.
last_del	timestamp with time zone	Time of the last delete on this table.
last_updated	timestamp with time zone	Time of the last update on this table.
last_analyze	timestamp with time zone	Time of the last analyze on this table.
last_vacuum	timestamp with time zone	Time of the last vacuum of this table (excludes vacuum full).
last_vacuum_full	timestamp with time zone	Time of the last vacuum full or last vacuum full freeze of this table.
size	bigint	Combined size of all of this table's files.
row_count	real	Number of rows in the table.
children	integer	Number of partitions, including middle-level and child partitions.
schema	name	Name of the schema this table belongs to.
table_name	name	Name of the table.
owner	name	Name of the database role that owns this table.
relstorage	character(1)	Storage mode of this table. a=append-optimized, c=column-oriented, h =heap, v = virtual, x= external table.
relkind	character(1)	The type of object: r = heap or append-optimized table.
bloat	real	Calculated bloat for the table when last scanned.
scan_size	bigint	Size of table when last scanned.
unused	real	Unused.
skew	real	Calculated skew for the table when last scanned.
last_scan_rowcount	real	Number of rows in the table when last scanned.

Column	Type	Description
last_scan_ts	timestamp with time zone	Time the table was last scanned for Recommendations.
accuracy	real	Accuracy ratio calculated from query history when the table was last scanned.
last_accuracy_ts	timestamp with time zone	Time the accuracy ratio was last set.
distributed_by	text	Distribution policy for the table.
age_percent	float	Tracks information for calculating when a table should have vacuum freeze run on it. It stores what percentage of the <code>xid_warn_limit</code> GUC the table's age is. For example, if a table's age is 70% of <code>xid_warn_limit</code> , its <code>age_percent</code> is stored as 0.7.
last_vacuum_freeze	timestamp with time zone	Records the last time that one of vacuum full, vacuum freeze or vacuum full freeze was performed on the table.
last_size_ts	timestamp with time zone	Time when the current table's size column was last updated.

## gpcc\_table\_info\_history

The `gpcc_table_info_history` table stores a daily snapshot of statistics about tables. Command Center saves statistics from the `gpcc_table_info` table just before midnight each night. There is one row for each table per day.

Column	Type	Description
ctime	timestamp (0) without time zone	Time the record was created.
dbid	oid	Object ID of the database.
relid	oid	Object ID of the table.
paroid	oid	Object ID of a partition.
reltablespace	oid	Object ID of the table's tablespace.
seq_scan	bigint	Number of sequential scans initiated on this table.
idx_scan	bigint	Number of index scans initiated on this table.
n_tup_ins	bigint	Number of rows inserted.

Column	Type	Description
n_tup_del	bigint	Number of rows deleted.
n_tup_upd	bigint	Number of rows updated (includes hot updated rows).
n_tup_hot_upd	bigint	Number of rows HOT updated (no separate index update required).
last_seq_scan	timestamp with time zone	Time of the last sequential scan on this table.
last_idx_scan	timestamp with time zone	Time of the last index scan on this table.
last_ins	timestamp with time zone	Time of the last insert on this table.
last_del	timestamp with time zone	Time of the last delete on this table.
last_upd	timestamp with time zone	Time of the last update on this table.
last_analyze	timestamp with time zone	Time of the last analyze on this table.
last_vacuum	timestamp with time zone	Time of the last vacuum of this table (excludes vacuum full).
last_vacuum_full	timestamp with time zone	Time of the last vacuum full of this table.
size	bigint	Combined size of all of this table's files.
row_count	real	Number of rows in the table.
children	integer	Number of partitions, including middle-level and child partitions.
schema	name	Name of the schema this table belongs to.
tablename	name	Name of the table.
owner	name	Name of the database role that owns this table.
relstorage	character(1)	Storage mode of this table. a=append-optimized, c=column-oriented, h = heap, v = virtual, x= external table.

Column	Type	Description
relkind	character(1)	The type of object: r = heap or append-optimized table.
bloat	real	Calculated bloat for the table when last scanned.
scan_size	bigint	Size of the table when last scanned.
unused	real	
skew	real	Calculated skew for the table when last scanned.
last_scan_rowcount	real	Number of rows in the table when last scanned.
last_scan_ts	timestamp with time zone	Time the table was last scanned for Recommendations.
accuracy	real	Accuracy ratio calculated from query history when the table was last scanned.
last_accuracy_ts	timestamp with time zone	Time the accuracy ratio was last set.
distributed_by	text	Distribution policy for the table.
age_percent	float	Tracks information for calculating when a table should have <code>VACUUM FREEZE</code> run on it. It stores what percentage of the <code>xid_warn_limit</code> GUC the table's age is. For example, if a table's age is 70% of <code>xid_warn_limit</code> , its <code>age_percent</code> is stored as 0.7
last_vacuum_freeze	timestamp with time zone	Records the last time that one of <code>VACUUM FULL</code> , <code>VACUUM FREEZE</code> or <code>VACUUM FULL FREEZE</code> was performed on the table.
last_size_ts	timestamp with time zone	Time when the current table's size column was last updated.

## Real-Time Monitoring Tables

### gpcc\_queries\_now

The `gpcc_queries_now` table saves real-time query metrics data.

Column	Type	Description
tmid	integer	A part of the query ID.
ssid	integer	Session ID of the query.
ccnt	integer	A part of the query ID.
username	character varying(64)	User name.



Column	Type	Description
db	character varying(64)	Database.
status	character varying(16)	Query status.
rsgname	character varying(64)	Resource group name.
rsqname	character varying(64)	Resource queue name.
rsqpriority	character varying(16)	Resource queue priority.
tsubmit	timestamp(0) without time zone	Submit time.
tstart	timestamp(0) without time zone	Start time.
cpu_time	double precision	CPU time accumulated (unit: seconds).
cpu_master	double precision	CPU time on master.
cpu_segment	double precision	CPU time on all segments.
cpu_master_percent	double precision	CPU percent on master.
cpu_segment_percent	double precision	CPU percent on all segments.
spill_size	bigint	Spill file size (bytes)
memory	bigint	Memory (KB)
disk_read_rate	double precision	Current disk read rate (KB/s).
disk_write_rate	double precision	Current disk write rate (KB/s).
disk_read_bytes	bigint	Current disk read bytes (bytes).
disk_write_bytes	bigint	Current disk write bytes (bytes).
skew_cpu	double precision	CPU skew.
plan_gen	character varying(16)	Plan generator for the query.
cost	double precision	Query cost.
query_text	text	Query text.

## Workload Management Tables

### gpcc\_wlm\_rule

The `gpcc_wlm_rule` table stores the definition of workload management rules (both assignment rules and workload rules).

Column	Type	Description
serial_number	integer	Serial number of the rule.
rule_id	integer	Unique ID of the rule shown in the Command Center interface.
rsgname	name	Resource group name used to assign rule.

Column	Type	Description
role	name	Database role name used to assign rule.
query_tag	text	Query tag string used to assign rule.
dest_rsg	name	Destination resource group name for moving a query.
cpu_time	integer	Maximum CPU time consumed by the query.
running_time	integer	Maximum running time for the query.
disk_io_mb	integer	Maximum total disk I/O used by the query.
planner_cost	float8	Maximum Postgres Planner cost for the query.
orca_cost	float8	Maximum GPORCA optimizer cost for the query.
slice_num	integer	Maximum number of slices assigned for executing the query.
action	integer	Action to performed when conditions are met.
active	boolean	Whether the rule is active or inactive.
ctime	timestamp without time zone	Time the rule was created.
etime	timestamp without time zone	Time that the rule was deleted.
idle session	json	JSON string containing parameters for idle session kill rules.
spill_file_mb	int	Spill file size, in MB.
cpuskew_percent	int	If the CPU skew percent of a query is equal to or higher than <code>cpuskew_percent</code> during <code>cpuskew_duration_sec</code> seconds, Command Center executes the rule.
cpuskew_duration_sec	int	If the CPU skew percent of a query is equal to or higher than <code>cpuskew_percent</code> during <code>cpuskew_duration_sec</code> seconds, Command Center executes the rule.

## gpcc\_wlm\_log\_history

The `gpcc_wlm_log_history` table stores the log history of workload rule actions that are triggered. Note that log history for assignment rules is not collected.

Column	Type	Description
ctime	timestamp without time zone	Time the log entry was created.
tstart	timestamp without time zone	Time the operation started.
tfinish	timestamp without time zone	Time the operation finished.

Column	Type	Description
rule_serial_number	integer	Serial number of the rule.
rule_id	integer	Unique ID of the rule shown in the Command Center interface.
tmid	integer	A time identifier for the query. All records associated with a query will have the same <code>tmid</code> .
ssid	integer	Session id for the database connection. All records associated with the query will have the same <code>ssid</code> .
ccnt	integer	Command number within the session. All records associated with the query will have the same <code>ccnt</code> .
action	integer	Workload rule action that was initiated.
rsgname	name	Resource group name used to assign rule.
role	name	Database role name used to assign rule.
status	text	Indication of whether the action succeeded or failed.
fail_msg	text	Message associated with a failed action.

## Data Loading Tables

Command Center uses the `gpcc_gpss_job` and `gpcc_gpss_log` tables to store information related to data loading.

### gpcc\_gpss\_job

Column	Type	Description
job_id	varchar(32)	The job ID.
job_name	text	The job name.
job_type	character varying(20)	The job type. Currently, there is just one job type: s3.
creator	character varying(64)	The creator of the job.
yaml_content	text	YAML description of the job.
is_active	boolean	Indicates whether or not the job has been removed. Set to <code>true</code> when the job is submitted or resubmitted. <code>false</code> means the job has been removed.

### gpcc\_gpss\_log

Column	Type	Description
--------	------	-------------

ctime	timestamp without time zone.	The time when Command Center received the log.
gpss_time	timestamp(0)	Greenplum Streaming Server time returned from Greenplum Streaming Server gRPC call.
job_id	varchar(20)	The id of the data loading job.
job_name_at_the_time	text	The name of the job at the time Command Center received the log message.
log_type	gpmetrics.gpss_log_type	Specifies whether the log is received from GPSS gRPC streaming calls (marked as 'event') or from GPCC API calls (marked as 'operation').
ctx	integer	Context of event(job status changed to: UNSPECIFIED=0/STOPPED=1/RUNNING=2/ERROR=3/REMOVED=4/SUCCESS=5/SUBMITTED=6), or context of operation(start=0/stop=1/remove=2/submit=3)
operation_status	gpmetrics.gpss_operation_status	The status of operations ('success'/'fail' for log_type=operation, 'not_applicable' for other types).
msg	text	Any messages returned from Greenplum Streaming Server.
log_id	integer	The id of the log. Increments automatically. When sorting logs, Command Center uses this value to break ties between logs with the same <code>ctime</code> , <code>log_type</code> , and <code>job_name_at_the_time</code> .

## Other Tables

Command Center uses the `gpcc_department` and `gpcc_role_department` tables to store information about departments and roles.

### gpcc\_department

Column	Type	Description
dept_id	integer	Department ID.
dept_name	character varying(64)	Department name. Limited to 64 characters, can only contain [A-Z][a-z][0-9][_].
dept_status	smallint	1 or 0, where 1 indicates active and 0 indicates deleted.

### gpcc\_role\_department

Column	Type	Description
rolname	name	Role name.
dept_id	integer	Department ID.

## User-Defined Functions

The `gpmetrics` schema includes a number of user-defined functions that provide users with information about queries and about the hosts on which the queries run. You can find the source code for all of these functions in the `$GPCC_HOME/sqls/` directory.

## gpcc\_delete\_department

The `gpcc_delete_department` user-defined function deletes a department name from the `gpperfmon` database. It takes one parameter: the name of the department you want to delete. You may only delete department names that have no roles associated with them.

## gpcc\_queries\_per\_hour

The `gpcc_queries_per_hour` user-defined function returns the average and maximum number of total queries, running queries, queued queries, blocked queries, and finished queries per hour.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_hour();
select * from gpmetrics.gpcc_queries_per_hour('2021-06-07');
select * from gpmetrics.gpcc_queries_per_hour('2021-06-09 00:00:00', '2021-06-09 13:20:05');
```

The first query returns results for the interval between 00:00:00 on the current date and the current time.

The second query returns results for the interval between 00:00:00 on '2021-06-07' and the current time.

The third query returns results for the interval between 00:00:00 on 2021-06-09 and 13:20:05 on 2021-06-09.

## gpcc\_queries\_per\_user

The `gpcc_queries_per_user` user-defined function returns, for each user, the number of queries whose runtime is longer than the input interval, per hour, in the specified time range. The function takes an interval, a start time, and an end time. The default start time is 00:00:00 today. The default end time is the current time. The interval can not be omitted. It accepts PostgreSQL's `INTERVAL` value.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_user('5min', '2018-05-01', '2018-06-01');
select * from gpmetrics.gpcc_queries_per_user('5min');
select * from gpmetrics.gpcc_queries_per_user('5min', '2018-05-01');
```

The first query returns, by user, all queries running longer than 5 minutes for the interval between 00:00:00 on 05/01/2018 and 23:59:59 on 06/01/2018.

The second query returns, by user, all queries running longer than 5 minutes in the interval between 00:00:00 today and the current time.

The third query returns, by user, all queries longer than 5 minutes in the interval between 00:00:00 on 2018-05-01 and the current time.

## gpcc\_queries\_per\_user\_max\_and\_total\_spill\_size

The `gpcc_queries_per_user_max_and_total_spill_size` user-defined function returns, for each user, the total `spill_size` and maximum `spill_size` per query per hour. The function takes a start time and an end time. The default start time is 00:00:00 today. The default end time is the current time.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_user_max_and_total_spill_size();
select * from gpmetrics.gpcc_queries_per_user_max_and_total_spill_size('2018-05-20','2018-05-21');
select * from gpmetrics.gpcc_queries_per_user_max_and_total_spill_size('2018-05-20');
```

The first query returns, by user, the total `spill_size` and maximum `spill_size` per query per hour for the interval between 00:00:00 on the current date and the current time.

The second query returns, by user, the total `spill_size` and maximum `spill_size` per query per hour for the interval between 00:00:00 on 05/20/2018 and 23:59:59 on 05/21/2018.

The third query returns, by user, the total `spill_size` and maximum `spill_size` per query per hour for the interval between 00:00:00 on '2018-05-20' and the current time.

## gpcc\_queries\_per\_user\_max\_cpu

The `gpcc_queries_per_user_max_cpu` user-defined function returns, for each user, the query with the maximum segment and master cpu usage per hour, along with details about the query. The function takes a start time and an end time. The default start time is 00:00:00 today. The default end time is the current time.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_user_max_cpu();
select * from gpmetrics.gpcc_queries_per_user_max_cpu('2018-05-20','2018-05-21');
select * from gpmetrics.gpcc_queries_per_user_max_cpu('2018-05-20');
```

The first query returns, by user, the query with the maximum segment and master cpu usage per hour for the interval between 00:00:00 on the current date and the current time.

The second query returns, by user, the query with the maximum segment and master cpu usage per hour for the interval between 00:00:00 on 05/20/2018 and 23:59:59 on 05/21/2018.

The third query returns, by user, the query with the maximum segment and master cpu usage per hour for the interval between 00:00:00 on '2018-05-20' and the current time.

## gpcc\_queries\_per\_user\_max\_run\_time

The `gpcc_queries_per_user_max_run_time` user-defined function returns, for each user, the longest running query per hour, along with details about the query. The function takes a start time and an end time. The default start time is 00:00:00 today. The default end time is the current time.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_user_max_run_time();
select * from gpmetrics.gpcc_queries_per_user_max_run_time('2018-05-20','2018-05-21');
```

```
select * from gpmetrics.gpcc_queries_per_user_max_run_time('2018-05-20');
```

The first query returns, by user, the longest running query per hour for the interval between 00:00:00 on the current date and the current time.

The second query returns, by user, the longest running query per hour for the interval between 00:00:00 on 05/20/2018 and 23:59:59 on 05/21/2018.

The third query returns, by user, the longest running query per hour for the interval between 00:00:00 on '2018-05-20' and the current time.

## gpcc\_queries\_per\_user\_max\_skew

The `gpcc_queries_per_user_max_skew` user-defined function returns, for each user, the query with the maximum amount of processing skew in the system (`skew_cpu`) per hour, along with details about the query. The function takes a start time and an end time. The default start time is 00:00:00 today. The default end time is the current time.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_user_max_skew();
select * from gpmetrics.gpcc_queries_per_user_max_skew('2018-05-20', '2018-05-21');
select * from gpmetrics.gpcc_queries_per_user_max_skew('2018-05-20');
```

The first query returns, by user, the query with the maximum amount of processing skew in the system for the interval between 00:00:00 on the current date and the current time.

The second query returns, by user, the query with the maximum amount of processing skew in the system for the interval between 00:00:00 on 05/20/2018 and 23:59:59 on 05/21/2018.

The third query returns, by user, the query with the maximum amount of processing skew in the system for the interval between 00:00:00 on '2018-05-20' and the current time.

## gpcc\_queries\_per\_user\_rows\_out

The `gpcc_queries_per_user_rows_out` user-defined function returns, for each user, the query with the maximum `rows_out` per hour, along with details about the query. The function takes a start time and an end time. The default start time is 00:00:00 today. The default end time is the current time.

Here are some examples:

```
select * from gpmetrics.gpcc_queries_per_user_rows_out();
select * from gpmetrics.gpcc_queries_per_user_rows_out('2018-05-20', '2018-05-21');
select * from gpmetrics.gpcc_queries_per_user_rows_out('2018-05-20');
```

The first query returns, by user, the query with the maximum `rows_out` per hour for the interval between 00:00:00 on the current date and the current time.

The second query returns, by user, the query with the maximum `rows_out` per hour for the interval between 00:00:00 05/20/2018 and 23:59:59 05/21/2018.

The third query returns, by user, the query with the maximum `rows_out` per hour for the interval between 00:00:00 on '2018-05-20' and the current time.

## gpcc\_queries\_top\_10\_cpu\_per\_resgroup

The `gpcc_queries_top_10_cpu_per_resgroup` user-defined function retrieves, by resource group, the top 10 queries utilizing the most CPU over the last 1 hour, 1 day, 1 week, and 1 month.

Here are some examples:

```
select * from gpcc_queries_top_10_cpu_per_resgroup();
select * from gpcc_queries_top_10_cpu_per_resgroup('1 day');
select * from gpcc_queries_top_10_cpu_per_resgroup('1 week');
select * from gpcc_queries_top_10_cpu_per_resgroup('1 month');
```

## gpcc\_queries\_top\_10\_memory\_per\_resgroup

The `gpcc_queries_top_10_memory_per_resgroup` user-defined function retrieves, by resource group, the top 10 queries utilizing the most memory over the last 1 hour, 1 day, 1 week, and 1 month.

Here are some examples:

```
select * from gpcc_queries_top_10_memory_per_resgroup();
select * from gpcc_queries_top_10_memory_per_resgroup('1 day');
select * from gpcc_queries_top_10_memory_per_resgroup('1 week');
select * from gpcc_queries_top_10_memory_per_resgroup('1 month');
```

## gpcc\_recommendations

The `gpcc_recommendations` user-defined function returns a list of tables in need of a vacuum-related action such as `VACUUM`, `VACUUM FULL`, `ANALYZE`, `SKEW`, and `VACUUM FREEZE`. You may call this function directly from the command line, from within an SQL program, or using a `psql` utility.

Here are some examples:

```
gpmetrics.gpcc_recommendations('vacuum-full');
gpmetrics.gpcc_recommendations('vacuum-freeze');
gpmetrics.gpcc_recommendations('analyze');
gpmetrics.gpcc_recommendations('skew');
```

If you set custom values for the `accuracy_threshold`, `bloat_unused_threshold`, or `bloat_dead_threshold` properties in the `gpcc.conf` file, you should define any values you've set via the `PGOPTIONS` option when calling this user defined function, as in this example: `PGOPTIONS="-c accuracy.threshold=0.2 -c bloat.vacuum.full.threshold=0.3 -c bloat.threshold=0.6" psql -d gpperfmon -c "select * from gpmetrics.gpcc_recommendations('vacuum');`

## gpcc\_system\_per\_hour

The `gpcc_system_per_hour` user-defined function returns a variety of system information, including: the average `cpu_user` and `cpu_sys` for all hosts, aggregated by hour; the average `load0`, `load1`, and `load2` per hour for all hosts, aggregated by hour; and the average disk write and disk read bytes for all hosts, aggregated by hour. The function takes a start time and end time. The default start time is 00:00:00 today. The default end time is the current time.

This example returns the system information summarized in the previous paragraph for the interval between 00:00:00 08/30/2021 and 23:59:59 08/31/2021.



```
gpperfmon=# select * from gpmetrics.gpcc_system_per_hour('2021-08-30','2021-08-31');
```

## gpcc\_update\_department

The `gpcc_update_department` user-defined function renames a department name. It takes two parameters: `from_name` -- which specifies the old name -- and `to_name` -- which specifies the new name. Once it has run, all the roles associated with the old name will now be associated with the new name.

# Configuration Files Reference

Configuration parameters for Greenplum Command Center are saved in the following locations.

- [Command Center Console Parameters](#) - This file stores configuration parameters for the Command Center web application and web server.
- [gpmetrics Configuration File](#) - This file stores configuration parameters for Command Center interface options and alert emails.
- [Command Center Setup Configuration File](#) - Used as input to the Command Center installer, this file customizes a non-interactive Command Center installation. See [Install With a Configuration File](#) for information about this file.
- [Metrics Collector Server Configuration Parameters](#) - The metrics collector extension declares custom server configuration variables that configure query metrics collection in Greenplum Database. See [Server Configuration Parameters](#) for information about these parameters.

Configuration parameters for the optional Greenplum Database gpperfmon service and collection agents are stored in the following locations.

- Greenplum Command Center gpperfmon service agents, [gpmmmon](#) and [gpsmon](#) - Parameters that configure the gpperfmon [gpmmmon](#) and [gpsmon](#) agents are in the `$MASTER_DATA_DIRECTORY/gpperfmon/conf/gpperfmon.conf` file, on the Greenplum Database master host.
- Greenplum Database configuration file - Server configuration parameters in the Greenplum Database `postgresql.conf` file configure the gpperfmon query history collection system.

See [The gpperfmon Database](#) in the *Greenplum Database Reference Guide* for information about the parameters in these configuration files.

## Command Center Console Parameters

The Command Center Console configuration file is on the Command Center host at `$GPCC_HOME/conf/app.conf`. Some parameters in this file are set by the Command Center installer.

You can add security settings in `app.conf` to suit your environment. See [Security Parameters](#).

After editing this file, reload the configuration by restarting the Command Center Console.

```
$ gpcc --start
```

```
appname = gpccws
```

The web server binary file. Do not change.

```
httpport = <port>
```

The web server port when `EnableHTTP` is true. The default is 28080.

```
httpsport = <port>
```

The web server port when `EnableHTTPS` is true. The default is 28080.

```
rpcport = <port>
```

The port on which the Command Center backend receives data from metrics collector agents. The default is 8899.

```
listentcp4 = [true | false]
```

When `true`, the address type is tcp4. The default is `true`.

```
runmode = [prod | dev | test]
```

The application mode, which can be `dev`, `prod` or `test`. The default, `prod`, is the recommended setting. In `dev` and `test` modes Command Center prints more verbose log messages. These are different logs than the logs affected by the `log_level` parameter.

```
session = [true | false]
```

Use sessions to manage user experience. The default is `true`. Sessions are stored in memory.

```
enablexsrf = [true | false]
```

Enable CSRF protection.

```
xsrfkey = <token_string>
```

The CSRF token.

```
xsrfexpire = <seconds>
```

CSRF expire time. The default is 2592000 seconds.

```
rendertype = json
```

The render type of the web server. Do not change.

```
printallsqls = [true | false]
```

Print all backend gpperfmon SQL to the web server console. The default is `false`.

```
log_level
```

The level of messages to log: `Debug`, `Info`, or `Error`. The default is `Info`. The values are not case-sensitive.

```
master_host = <hostname>
```

The Greenplum Database host name. The default is `localhost`.

```
master_port = <port>
```

The Greenplum Database master port. The default is 5432.

```
path = /usr/local
```

Path to the directory where Greenplum Command Center is installed.

```
display_name = <display_name>
```

The display name for the console.

```
enable_kerberos = [true | false]
```

True if Kerberos authentication is enabled for Command Center. The default is `false`.

```
enable_history = [true | false]
```

True if history data collection is enabled for Command Center. The default is `true`. This parameter is managed in Command Center by setting **Enable GPCC history data collection** on or off on the **Admin> Settings** page.

```
HTTSPCertFile = </path/to/cert.pem> HTTPSKeyFile = </path/to/cert.pem>
```

Set both of these properties to the full path to a .pem file containing the certificate and private key for the Command Center web server.

```
EnableHTTPS = [true | false]
```

Enable listening on the secure SSL port. True if SSL is enabled. Only one of `EnableHTTPS` or `EnableHTTP` can be true.

```
EnableHTTP = [true | false]
```

Enable listening on the HTTP port. True if SSL is not enabled. Only one of `EnableHTTP` or `EnableHTTPS` can be true.

```
HTTPAddr = <ipaddress>
```

The IPv6 address of the host that runs Command Center. It is only necessary to set this parameter if Command Center is running in an IPv6 environment.

```
stats_check_interval = <seconds>
```

How often the statistics in the Command Center Table Browser are refreshed. The default is `300`. New tables and changed values such as file size and last access time may not be seen until `stats_check_interval` seconds have elapsed.

```
ws_perf_port = <port>
```

Port to access the Command Center web server Go profiling data. (See [pprof](#) for more information.) The default is `6162`. Choose another port if there is a port conflict or if you are setting up another Command Center instance on the same host.

```
agent_perf_port = <port>
```

Port to use to access agent Go profiling data. The default is `6163`. Choose another port if there is a port conflict on segment hosts, or if you are setting up another Command Center instance on the same cluster.

```
gpmetrics_home = <dirname>
```

The location of the `gpmetrics` directory. If you don't set this the default is `$HOME`.

```
XSRFSecure = true
```

The Command Center installer writes this parameter to `$GPCC_HOME/conf/app.conf` if a user configures Command Center to use the HTTPS protocol. This parameter defaults to `false`.

**NOTE:** Do not change this value unless you are experiencing problems.

## Setting Security Parameters

You may customize the following security headers:

```
"Cache-Control",           // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/Cache-Control
"Content-Security-Policy",  // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/Content-Security-Policy
"Permissions-Policy",      // See https://www.w3.org/TR/permissions-policy-1/
"Referrer-Policy",        // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/Referrer-Policy
"Strict-Transport-Security", // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/Strict-Transport-Security
"X-Content-Type-Options",  // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/X-Content-Type-Options
"X-Frame-Options",        // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/X-Frame-Options
"X-XSS-Protection",       // See https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/X-XSS-Protection
```

The following headers are configured by default and are set to these values:

```
"Cache-Control":           "no-store",
"Referrer-Policy":         "same-origin",
"Strict-Transport-Security": "max-age=31536000",
"X-Content-Type-Options":  "nosniff",
"X-Frame-Options":         "DENY",
"X-XSS-Protection":       "1; mode=block",
```

Where:

- **"Cache-Control": "no-store"** indicates that the response may not be stored in any cache.
- **"Referrer-Policy": "same-origin"** indicates a referrer will be sent for same-site origins, but cross-origin requests will send no referrer information.
- **"Strict-Transport-Security": "max-age=31536000"** indicates the time, in seconds, that the browser should remember that a site is only to be accessed using HTTPS.
- **"X-Content-Type-Options": "nosniff"** blocks or allows requests depending on type.
- **"X-Frame-Options": "DENY"** indicates that the page cannot be displayed in a frame, regardless of the site attempting to do so.
- **"X-XSS-Protection": "1; mode=block"** enables XSS filtering and the browser will block page rendering if it detects an attack.

To customize any of these headers, enter your values in the `app.conf` file and restart Command Center. For example, to customize `Content-Security-Policy`, `Permissions-Policy`, and `X-Frame-Options` use a `app.conf` entry similar to:

```
[security_headers]
Content-Security-Policy = default-src 'self' http://example.com;
Permissions-Policy = fullscreen=(), geolocation=()
X-Frame-Options = DENY
```

## Setting TLS Cipher Suites

By default, Command Center supports the following cipher suites:

- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

To use cipher suites other than the default four, add them to `$GPCC_HOME/conf/app.conf` in a section labeled `[tls_cipher_suites]`, as in the following example:

```
[tls_cipher_suites]
Enable_TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 = true
Enable_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 = true
Enable_TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 = true
Enable_TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 = true
Enable_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 = true
Enable_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA = true
Enable_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 = true
Enable_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA = true
Enable_TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA = true
Enable_TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA = true
Enable_TLS_RSA_WITH_AES_128_GCM_SHA256 = true
Enable_TLS_RSA_WITH_AES_256_GCM_SHA384 = true
Enable_TLS_RSA_WITH_AES_128_CBC_SHA256 = true
Enable_TLS_RSA_WITH_AES_128_CBC_SHA = true
Enable_TLS_RSA_WITH_AES_256_CBC_SHA = true
Enable_TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305 = true
Enable_TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305 = true
```

**Warning:** Cipher suites that are not among the four default cipher suites may have potential security risks and are not recommended.

When there are one or more entries under `[tls_cipher_suites]` in `app.conf`, Command Center will not use any default cipher suites, unless they are also declared in the `[tls_cipher_suites]` section.

## gpmetrics Configuration File Reference

Greenplum Command Center uses the `gpcc.conf` configuration file to save configuration information entered in the Command Center user interface.

You should not normally edit the `gpcc.conf` file directly. Instead, modify configuration information in the Command Center user interface. The threshold properties for the **Recommendations** page reports are an exception and must be added manually. If you edit the file, you must restart Command Center for the new values to take effect. The `gpcc.conf` file is created in the `$HOME/gpmetrics/` directory on the Greenplum Database master or standby host where you start Command Center. The file is an INI-format configuration file, containing properties defined as `key = value` entries, one property per line. Properties are organized within sections, with the exception of a handful that have no associated section.

Property	Description	Section
----------	-------------	---------

min_query_time	Queries shorter than number of seconds specified will not be captured in Query History.	[history]
wlm_short_query_threshold	Specifies the minimum number of seconds that a query must run before Command Center applies the action of a matching workload management rule. Use this parameter to prevent Command Center from applying rule actions to short-running queries. The default is 45 seconds.	[wlm]
wlm_query_cool_down_time	Specifies the wait time, in seconds, for retrying a failed workload management rule (canceling a query or moving a query to another resource group). Command Center makes 2 retry attempts for failed rules, after waiting the configured amount of time. The default is 15 seconds.	[wlm]
allow_anonymous	If <b>true</b> , Command Center users can access the Query Monitor view without logging into Command Center. You can change this setting on the Command Center <b>Admin&gt; Permissions</b> page.	N/A
resource_queue_import_status	Command Center uses this property to determine whether to offer to import Greenplum Database resource queues to resource groups when you access the <b>Admin&gt; Workload Mgmt</b> view. The default is <b>false</b> .	N/A
emailFrom	The email address to set on the "From:" line of alert emails. The default is <b>noreply-gpcc-alerts@pivotal.io</b> .  Note: Set the email and smtp properties on the Command Center <b>Admin&gt; Alerts</b> page.	[alert]
emailTo	A comma-delimited list of email addresses to send alert emails.	[alert]
smtpUserName	The account name to use when authenticating with the SMTP server.	[alert]
smtpServer	The address and port of the SMTP server to use for alert emails.	[alert]
smtpPassword	The password used to authenticate the SMTP user with the SMTP server, base 64-encoded.	[alert]
age_ratio	Specifies the age a table must reach to prompt Command Center to issue a <b>VACUUM FREEZE</b> recommendation in the Age Rank report on the Recommendations page. You must specify a value between 0 and 1. The default is 0.7. The age is calculated using two Greenplum Database GUCs: <b>xid_warn_limit</b> and <b>xid_stop_limit</b> . For example, 0.7 means a table is regarded as old enough and in need of <b>VACUUM FREEZE</b> when its age is above $70\% * (2^{31} - 1 - \text{xid\_warn\_limit} - \text{xid\_stop\_limit})$ .	[recommendations]
recommendations_visited	Indicates whether the Recommendations page has been visited.	N/A
bloat_threshold	Specifies the minimum percentage of bloat (space occupied by dead tuples) required to include a table with a <b>VACUUM</b> recommendation in the <b>Bloat Rank</b> report on the <b>Recommendations</b> page. The default is <b>0.1</b> (10%).	[recommendations]
bloat_vacuum_full_threshold	Specifies the minimum percentage of bloat (combined dead tuples and unused space) required to include a table with a <b>VACUUM FULL</b> recommendation in the <b>Bloat Rank</b> report on the <b>Recommendations</b> page. The default is <b>0.5</b> (50%).	[recommendations]
accuracy_threshold	Specifies the minimum inaccuracy percentage required to include a table in the <b>Accuracy% Rank</b> table on the <b>Recommendations</b> page. The default is <b>0.1</b> (10%).	[recommendations]

Property	Description	Section
skew_threshold	Specifies the minimum skew percentage required to include a table in the <b>Skew% Rank</b> report on the <b>Recommendations</b> page. The default is <code>0.1</code> (10%).	[recommendations]
skew_tuple_per_segment	Specifies a minimum number of tuples per segment to include a table in the <b>Skew% Rank</b> report on the <b>Recommendations</b> page. This parameter helps to avoid reporting tables that have high skew due to small numbers of tuples. The default is <code>128</code> , which means a table with fewer tuples than 128 times the number of segments will be excluded from the report.	[recommendations]
stats_check_interval	Specifies how often to refresh the statistics in the Command Center Table Browser. The default is <code>300</code> . New tables and changed values such as the file size and the last access time may not be seen until <code>stats_check_interval</code> seconds have elapsed. To disable table statistics collection, set this property to <code>-1</code> .	[table_browser]
disable_size_check	When set to <code>true</code> , the table browser will not collect table size information, in order to achieve performance gains when the number of tables is very large. The default is <code>false</code> .	[table_browser]
gpssServer	Specifies the hostname or IP address of the GPSS server listening for client requests from Command Center.	[gpss]
gpssPort	Specifies the port number on which the GPSS server listens for client requests from Command Center. The default is 5000.	[gpss]
gpssUsername	Specifies the username for authenticating to the GPSS server.	[gpss]
gpssAuthentication	Specifies the authentication method for authenticating to the GPSS server. Accepted values are <code>without-tls</code> , <code>tls</code> , and <code>tls-skip-ca-verify</code> .	[gpss]
gpssCaFile	Specifies the file system path to the Certificate Authority file. This file should be accessible by Command Center on the host where it runs. This entry can be empty if the CA is already in the system pool on that host.	[gpss]
enable	Specifies whether Command Center automatically collects performance data and saves it in the <code>\$GPCC_HOME/logs/pprof</code> file. The default is <code>true</code> .	[pprof]
interval	Specifies how long Command Center will collect data. It takes a value of datatype <i>time duration</i> ; a <i>time duration</i> string is a possibly signed sequence of decimal numbers, each with optional fraction and a unit suffix, such as "30ms", "1.5h" or "2h45m". Valid time units are "s", "m", "h". The default is 1h.	[pprof]
cpu_interval	Specifies how long Command Center will conduct CPU profiling. It takes a value of datatype <i>time duration</i> ; a <i>time duration</i> string is a possibly signed sequence of decimal numbers, each with optional fraction and a unit suffix, such as "30ms", "1.5h" or "2h45m". Valid time units are "s", "m", "h". The default is 10s.	[pprof]
ttl	Specifies time-to-live for profile logs. It takes a value of datatype <i>time duration</i> ; a <i>time duration</i> string is a possibly signed sequence of decimal numbers, each with optional fraction and a unit suffix, such as "30ms", "1.5h" or "2h45m". Valid time units are "s", "m", "h". The default is 24h	[pprof]
profiles	Specifies a comma-delimited list of benchmarking profile names. Valid values are <code>heap</code> -- which means Command Center collects heap benchmark data, <code>cpu</code> -- which means Command Center collects CPU benchmark data, and <code>heap,cpu</code> -- which means Command Center collects both. The default is <code>heap</code> .	[pprof]

## Server Configuration Parameters



The metrics collector extension in Greenplum Database defines custom server configuration parameters to configure metrics collection options. If the metrics collector extension is installed and the value of the `shared_preload_libraries` configuration parameter includes `metrics_collector` when Greenplum Database starts, these parameters can be viewed and set using the Greenplum Database `gpconfig` utility.

`gpcc.enable_send_query_info`

Enables collection of query metrics in Greenplum Database. When `on`, the metrics collector sends query metrics to the Greenplum Command Center agent.

**Type:** `boolean`

**Default:** `on`

`gpcc.enable_query_profiling`

When `off` the metrics collector does not collect queries executed by the `gpmon` user in the `gpperfmon` database or plan node history for queries that run in less than ten seconds (or `min_query_time`, if it is set and greater than 10 seconds). If enabled in a session, the metrics collector collects those queries in the session.

**Type:** `boolean`

**Default:** `off`

`gpcc.query_metrics_port`

The port number to send query metrics. The metrics collector sends query metrics to the Greenplum Command Center agent process on this port.

**Type:** `integer`

**Default:** `9898`

`gp_external_enable_exec`

This parameter is enabled by default and must remain enabled. It allows the use of external tables that execute OS commands or scripts on the segment hosts. The Command Center agents use this type of external tables to collect current system metrics from the segments.

`gp_enable_query_metrics`

When `on`, enables query metrics collection. The default is `off`. After setting this configuration parameter, Greenplum Database must be restarted for the change to take effect.

`gpcc.enable_save_full_query_text`

When `on`, Command Center saves to the `gpmetrics.gpcc_queries_history` table the full query text for queries whose text is longer than 100 KB. When `off`, Command Center saves only 5KB of query text for queries whose text is longer than 100 KB. It is turned off by default.

**NOTE:** The full query text is always saved for queries whose text is shorter than 100 KB, regardless of what `gpcc.enable_save_full_query_text` is set to.

`gpcc.enable_send_lock_info` Turn this parameter off to reduce Command Center's memory consumption by stopping Command Center from collecting lock information for queries. When it is off, the Query Monitor will not report a query's lock information or whether the query is blocked by another query. **NOTE:** This GUC was introduced in maintenance release 6.7.1.

You must have superuser permission to change this parameter. In addition, you must restart Greenplum Database in order for the change to take effect.

It is turned on by default.

`gpcc.enable_send_instrument`

When off, Command Center will not display some query details -- such as the visual query plan tree -- and some statistics. This will reduce Command Center's memory consumption. It is turned on by default.

`gpcc.enable_send_inner_query` When turned on, users can see the statements within functions that the query is calling when they click the **Inner Queries** tab in the query monitor's **Query Detail** view. It is turned off by default.

`gp_instrument_shmem_size` The amount of shared memory, in kilobytes, allocated for query metrics. The default is 5120 and the maximum is 131072. At startup, if `gp_enable_query_metrics` is set to on, Greenplum Database allocates space in shared memory to save query metrics. This memory is organized as a header and a list of slots. The number of slots needed depends on the number of concurrent queries and the number of execution plan nodes per query. The default value, 5120, is based on a Greenplum Database system that executes a maximum of about 250 concurrent queries with 120 nodes per query. If the `gp_enable_query_metrics` configuration parameter is off, or if the slots are exhausted, the metrics are maintained in local memory instead of in shared memory.

`shared_preload_libraries` A comma-separated list of shared libraries that are to be preloaded when Greenplum Database starts. The workload management and query metrics extension libraries must be included in this configuration parameter to use Greenplum Command Center.

`track_activities` Enables the collection of information on the currently executing command of each session, along with the time when that command began execution. The default value is true. Only superusers can change this setting.

# Securing Greenplum Command Center

Greenplum Command Center Console can be secured by encrypting network traffic between the web server and users' browsers, authenticating Command Center users, and managing users' permissions to access Command Center features.

## SSL/TLS Encryption

Greenplum Command Center supports SSL/TLS encryption to secure connections between browsers and the Command Center web server. Command Center supports TLS 1.2 protocol and higher. When enabled, Command Center uses the Secure WebSockets API, enabling long-lived, full-duplex connections, in addition to encryption.

To enable SSL/TLS encryption, you need a combined certificate/key file for the Command Center web server in place when you install Command Center. The file contains a private key and a server certificate.

You can request a key pair and certificate from your organization's internal certificate authority or from an external certificate authority. You can also create a self-signed certificate with a cryptography suite such as OpenSSL. If you create a self-signed certificate, however, clients will have to override a security warning when they first connect to the Command Center web server.

To create the combined certificate/key file, create a text file, for example `server.pem`, and copy the entire body of private key and certificate into it. Make sure to include the beginning and end tags:

```
-----BEGIN RSA PRIVATE KEY-----
< private key >
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
< certificate >
-----END CERTIFICATE-----
```

You can concatenate additional certificates to the end of the file if a certificate chain is required to authenticate the server certificate.

Place the file on the server where Command Center will execute, for example in the `/etc/ssl/certs` directory of the Greenplum Database master host. When you install Command Center the installer prompts you to enter the full path to this file. See [Command Center Console Parameters](#) for details.

## Authentication Options

Users logging in to Greenplum Command Center are authenticated with the Greenplum Database host-based authentication system. Users can enter credentials as a user name and password or, if Kerberos authentication is configured, by authenticating with Kerberos on their workstation before browsing to the Command Center web server.

**Note:** Greenplum Command Center does not accept logins from the gpadmin user, or from users configured with trust authentication in the `pg_hba.conf` file.

Database users must first be added to the Greenplum Database by using commands such as `CREATE ROLE` or `CREATE USER`. The `LOGIN` privilege is required. This example creates a login user with an encrypted password:

```
CREATE ROLE cc_user WITH LOGIN ENCRYPTED PASSWORD 'changeme';
```

The `pg_hba.conf` configuration file determines how authentication will proceed. This file contains a list of entries that are compared to attributes of the user's connection request, including the type of connection, network location of the originating host, database name, and login user name. When a match is found, the authentication method specified in the entry is applied.

The `pg_hba.conf` file can be viewed by Operators and edited by Admins in the Command Center console on the [Admin>Authentication](#) page.

The `md5` and `password` authentication methods authenticate the user name and password with the Greenplum Database `pg_roles` system table. The `md5` method requires the password to be MD5-encoded when sent over the network, so it is preferred over the `password` method, which sends the password in clear text.

The `ldap` authentication method authenticates the user name and password with an LDAP server. The LDAP server and parameters are specified in the options field of the `pg_hba.conf` entry. See the PostgreSQL [LDAP authentication](#) documentation for the format of the LDAP options.

The `gss` authentication method is used for Kerberos authentication. To use Kerberos with Command Center, Kerberos authentication must be enabled for the Greenplum Database system and Command Center must also be configured. Users authenticate with the Kerberos KDC on their workstations (using `kinit`, for example) before connecting to the Command Center web server. The role name in Command Center is the user's Kerberos principal name.

For details about setting up Kerberos authentication, see [Enabling Kerberos Authentication with Greenplum Command Center](#).

See the PostgreSQL [Authentication methods](#) documentation for additional details of the authentication options.

## Authorization



**Note:** The functionality described in this section has not been fully implemented in Greenplum Command Center 4.0.0. Only Admin and Self Only permission levels are available.

Command Center manages permission levels using Greenplum Database roles and groups. The Basic, Operator Basic, and Operator permission levels correspond to the `gpcc_basic`, `gpcc_operator_basic`, and `gpcc_operator` group roles in the database. The Admin permission level is conferred to roles that have the `SUPERUSER` privilege. A user who has not been added to any of the groups and does not have `SUPERUSER` privilege has the most restrictive permission level, Self Only.

Greenplum Database superusers can manage permission levels on the Command Center [User](#)

**Management** page. Superusers can also directly assign users roles in the database by using the `ALTER USER`, `ALTER GROUP`, and related commands to add or remove users from groups and add or remove the `SUPERUSER` privilege. If a role is configured for more than one permission level, Command Center uses the highest permission level.

Command Center users have the following capabilities, according to their permission levels:

## Self Only

Users can view metrics and view and cancel their own queries.

Any Greenplum Database user successfully authenticated through the Greenplum Database authentication system can access Greenplum Command Center with Self Only permission. Higher permission levels are required to view and cancel other's queries and to access the System and Admin Control Center screens.

## Basic

Allows users to view metrics, view all queries, and cancel their own queries.

Users with Basic permission are members of the Greenplum Database `gpcc_basic` group.

## Operator Basic

Allows users to view metrics, view their own and others' queries, cancel their own queries, and view the System and Admin screens.

Users with Operator Read-only permission are members of the Greenplum Database `gpcc_operator_basic` group.

## Operator

Allows users to view their own and others' queries, cancel their own and other's queries, and view the System and Admin screens.

Users with Operator permission are members of the Greenplum Database `gpcc_operator` group.

## Admin

Allows users access to all views and capabilities in the Command Center.

Greenplum Database users with the `SUPERUSER` privilege in Greenplum Database have Superuser permissions in Command Center.

## Managing Greenplum Command Center Authentication

The **Admin> Authentication** screen allows users with Operator Basic, Operator, and Admin permission to view the Greenplum Database host-based authentication file, `pg_hba.conf`.

**Host-Based Authentication** Controls user access for all GPDB activity on this server

For guidance on managing `pg_hba.conf` consult the postgres documentation

pg\_hba.conf This version 2018-09-25, 12:10 LOAD VERSION... ABANDON CHANGES SAVE CONFIG AND UPDATE GPDB

local	all	gpadmin		ident
host	all	gpadmin	127.0.0.1/28	trust
host	all	gpadmin	192.168.1.144/32	trust
host	all	gpadmin	::1/128	trust
host	all	gpadmin	2605:a601:4199:bb00:a00:27ff:...	trust
host	all	gpadmin	fe80::a00:27ff:fe6c:43b4/128	trust
local	replication	gpadmin		ident
host	replication	gpadmin	same-net	trust
local	gpperfmon	gpmon		md5
host	all	gpmon	127.0.0.1/28	md5
host	sales	nickd	nickd-pc	md5
host	all	gpmon	::1/128	md5

Type Database User Address (CIDR/Hostname) Method Comment

host all gpmon 192.168.1.144/28 md5

DONE Cancel

Users with Admin permission can add, remove, change, and move entries in the file. The Command Center UI validates entries to ensure correct syntax. Previous versions of the file are archived so that you can restore an earlier version or audit changes.

See [Authentication Options](#) for an overview of user authentication options for Greenplum Database and Greenplum Command Server.

See [Configuring Client Authentication](#) in the *Greenplum Database Administrator Guide* for a detailed description of the contents of the `pg_hba.conf` file.


## Viewing the Host-Based Authentication File





Choose **Admin>Authentication** to display the content of the Greenplum Database `pg_hba.conf` file.

The `pg_hba.conf` file contains a list of entries that specify the characteristics of database connection requests and authentication methods. When Greenplum Database receives a connection request from a client, it compares the request to each entry in the `pg_hba.conf` entry in turn until a match is found. The request is authenticated using the specified authentication method and, if successful, the connection is accepted.

## Editing the Host-Based Authentication File

Command Center users with the *Admin* permission can edit the `pg_hba.conf` file. Note that any changes you make are lost if you move to another screen before you save them.

- To change an existing entry, click anywhere on the entry. Edit the fields and click **Save** to save your changes, or **Cancel** to revert changes.
- To move an entry up or down in the list, click on the  symbol, drag the line to the desired location, and release.
- To add a new entry to the end of the file, click **Add New Entry** at the bottom of the screen. Edit the fields and click **Save** to save your changes, or **Cancel** to abandon the new entry.

- To add a new entry after an existing entry, highlight the existing entry and click . Edit the fields and click **Save** to save your changes, or **Cancel** to abandon the new entry.
- To copy an entry, select the entry and click . A copy of the selected entry is added below the selected entry and displayed for editing. Edit the fields and click **Save** to save your changes, or **Cancel** to abandon the copy.
- To add a comment to the file, add an entry by clicking **Add New Entry** or  and then choose # from the **Type** list.
- To toggle an entry between active and inactive, select the line and click the **active/inactive** toggle control to the right. This action adds or removes a comment character (#) at the beginning of the entry.
- To remove an entry, highlight the line and click . The entry is displayed with strikethrough text. You can restore the entry by highlighting it and clicking **undelete**. The entry is permanently removed when you click **Save config and update GPDB**.
- To finish editing, click **Save config and update GPDB**. Then click **Save and Update** to save your changes or click **Cancel** to return with your edits intact.

When you select **Save and Update**, the `pg_hba.conf` file is saved and refreshed in Greenplum Database. Note that existing client connections are unaffected.

## Loading a Previous Version of the Host-Based Authentication File

When you save a new version of the `pg_hba.conf` file, a copy is saved in the Greenplum Database `$MASTER_DATA_DIRECTORY/pg_hba_archive` directory as `pg_hba.conf-<timestamp>`.

To view an archived version of the `pg_hba.conf` file, click **Load versions...** and click the timestamp for the version to display.

To revert to a previous version of the file, load the previous version and then click **Save config and update GPDB**. The configuration is refreshed in Greenplum Database and saved as a new version in the archive directory.

## Enabling Authentication with Kerberos

If you have enabled Kerberos authentication for Greenplum Database, you can set up Greenplum Command Center to accept connections from Kerberos-authenticated users.

Greenplum Database and Command Center include support for the Generic Security Service Applications Program Interface (GSS-API) standard. A related standard, Simple and Protected GSS-API Negotiation Mechanism (SPNEGO), describes the protocol GSS-API clients and servers use to agree on the method of authentication.

With a SPNEGO-compliant web application such as Command Center, the client and server agree on the authentication method on the client's initial HTTP request. If Kerberos authentication is not supported on both ends of the connection the server falls back to basic authentication, and displays a login form requesting a user name and password. If a user has authenticated on the workstation with Kerberos and has a valid ticket granting ticket, the web browser offers the user's credential to the Command Center web server. A Kerberos-enabled Command Center web server is configured to

handle the authenticated user's connection request in one of three modes, called strict, normal, or gpmon-only.

#### Strict

Command Center has a Kerberos keytab file containing the Command Center service principal and a principal for every Command Center user. If the principal in the client's connection request is in the keytab file, the web server grants the client access and the web server connects to Greenplum Database using the client's principal name. If the principal is not in the keytab file, the connection request fails.

#### Normal

The Command Center Kerberos keytab file contains the Command Center principal and may contain principals for Command Center users. If the principal in the client's connection request is in Command Center's keytab file, it uses the client's principal for database connections. Otherwise, Command Center uses the gpmon user for database connections.

#### gpmon-only

Command Center uses the gpmon database role for all Greenplum Database connections. No client principals are required in the Command Center's keytab file. This option can be used, for example, if Command Center users authenticate with Active Directory and you do not want to maintain client principals in the keytab file.

If you have set up Kerberos authentication for Greenplum Database, most of the configuration required to enable Command Center Kerberos authentication has been done. The Command Center Kerberos configuration builds upon the Greenplum Database Kerberos setup.

Kerberos authentication can be enabled by responding to prompts when you install Command Center, or you can use the `gpcc --krbenable` command to enable Kerberos after Command Center has been installed.

## Before You Begin

Kerberos authentication must be enabled for Greenplum Database. See [Using Kerberos Authentication](#) for instructions. Make sure the following prerequisites are met before you continue:

- The `krb5-workstation` package and associated libraries (`libkrb5*`) must be installed on the Greenplum master host and each client workstation.
- The date and time on the Greenplum master host and all client workstations must be synchronized with the KDC.
- The `krb5.conf` configuration file must be the same on the KDC host, the Greenplum Database master host, and client workstations.
- The KDC database must have a service principal for Greenplum Database. The default service name for Greenplum Database is `postgres/<master-host>@<realm>`. You can choose a service name other than `postgres`, but it must match the value of the `krb_srvname` parameter in the `$MASTER_DATA_DIRECTORY/postgresql.conf` file.
- A keytab file with the Greenplum Database principal must be installed on the Greenplum master host and identified by the `krb_server_keyfile` parameter in the `$MASTER_DATA_DIRECTORY/postgresql.conf` file.



- Each client workstation can have a keytab file containing their Kerberos principal, `<username>@<realm>`.

## Add Command Center Principals to the KDC Database

Before you configure Command Center for Kerberos authentication, you must create the required Kerberos principals. All of the principals used with Command Center are created in the Greenplum Database Kerberos realm. Command Center users can use the same Kerberos principal to authenticate with Command Center and Greenplum Database.

### Command Center Service Principal

A service principal is needed for the Command Center web server. This principal has the format `HTTP/<host>@<realm>`. For example, if users access Command Center at the URL `http://mdw.example.com:28080`, the `<host>` part of the service key is `mdw.example.com` and the `<realm>` part is the Greenplum Database Kerberos realm, for example `GPDB.KRB`.

Note that Kerberos authentication only works if Command Center users enter the host in the same format specified in the Kerberos service principal. If the principal specifies the FQDN, for example, using the host's IP address in the browser URL will not work; the web server will fall back to basic authentication.

### Greenplum Database gpmon User

Command Center uses the gpmon Greenplum role to access the gpperfmon database, which contains data presented in the Command Center UI.

You can choose to authenticate the gpmon user with Kerberos or with basic authentication. To use Kerberos, you must create a principal for the gpmon user.

If you choose to use basic authentication you do not need a Kerberos principal for the gpmon user. The gpmon user will authenticate with Greenplum Database using the password saved in the `.pgpass` file in the gpadmin user's home directory on the host running Command Center. If you run Command Center on a host other than the Greenplum Database master host, you must copy the `.pgpass` file from the master host to the Command Center host. See [Changing the gpmon Password](#) for instructions to manage the gpmon password.

### Command Center Users

Add Kerberos principals for any Command Center users who do not already have principals in the KDC for Greenplum Database.

## Adding Kerberos Principals

To add the required principals, perform the following steps as root on the KDC server.

1. Add a principal for the Command Center web service. Be sure to specify the `<gpcc-host>` in the same format that users should enter the host in their browsers.

```
# kadmin.local -q "addprinc -randkey HTTP/<gpcc-host>@<realm>"
```

2. If you want the gpmon database user to use Kerberos authentication, add a gpmon principal.

```
# kadmin.local -q "addprinc gpmon@<realm>"
```

3. Add principals for any new Command Center users.

```
# kadmin.local -q "addprinc cc_user1@<realm>"
```

Repeat for each new Command Center user.

## Set Up Keytab Files

After you have created all of the Kerberos principals needed, you create and distribute keytab files. Keytab files contain Kerberos principals and encrypted keys based on the principals' Kerberos passwords. Keytab files are needed for the Greenplum Database master and standby hosts and the Command Center host.

You can also create a keytab file for each Greenplum Database or Command Center user containing just the user's principal. This keytab file is installed on the user's workstation to enable the user to authenticate to Kerberos. Note that all keytab files must contain the most recent versions of the principals' keys.

## Command Center Running on the Greenplum Master Host

If the Greenplum Command Center web server is running on the Greenplum Database master host, Command Center can share the Greenplum Database keytab file. You need to create a keytab file that contains the following principals:

- Service key for the `postgres` process on the Greenplum Database master host, for example `postgres/mdw.example.com@GPDB.EXAMPLE`.
- Service key created for Command Center in the previous section, for example `HTTP/mdw.example.com@GPDB.KRB`.
- A principal for every Kerberos-authenticated Greenplum Database or Command Center user.

All service keys and principals should be in the Greenplum Database realm.

To create a keytab file for Greenplum Database and Command Center, perform the following steps as root on the KDC server.

1. Create a keytab file containing the Greenplum Database service key, the command center service key, and all database and Command Center users.

```
kadmin.local -q "ktadd -k gpdb-kerberos.keytab postgres/mdw.example.com@GPDB.KRB  
B HTTP/mdw.example.com@GPDB.KRB"
```

You can enter one or more principals with each `ktadd` command. You can specify a wildcard using the `-glob` option. For example this command adds all principals in the `GPDB.KRB` realm, including service principals and admin users.

```
kadmin.local -q "ktadd -k gpdb-kerberos.keytab -glob *@GPDB.KRB"
```

2. Copy the keytab you created to the Greenplum Database master host, replacing the old keytab file. The location of the file is given by the `krb_server_keyfile` parameter in the `$MASTER_DATA_FILE/postgresql.conf` file. Set the permissions on the file so that it can be

read only by the gpadmin user.

3. Update any entries required for new Greenplum Database principals in the `pg_hba.conf` file and `pg_ident.conf` files. See [Update the Greenplum Database pg\\_hba.conf File](#) for details.

## Command Center Running on the Standby Master

If the Command Center web server is on a different host than the Greenplum Database master, you need separate keytab files for Greenplum Database and Command Center. The keytab file for Greenplum Database may not require any updates, but you will need to create a keytab file for Command Center.

- The Greenplum Database keytab file must contain the Greenplum Database service key and all principals for users with database access.
- The Command Center keytab file contains the Command Center service key and principals for users that have Command Center access. Users with Command Center access must also have Greenplum Database access, so user principals in the Command Center keytab file must also be in the Greenplum Database keytab file.

Update the Greenplum Database keytab if you created new database roles and principals for Command Center. For example, if you want to use Kerberos authentication for the gpmon user, you must create a principal and add it to both the Greenplum Database and Command Center keytab files.

To create the keytab file for Command Center, perform the following steps as root on the KDC host.

1. Create a keytab file and add the Command Center service key.

```
kadmin.local -q "ktadd -k gpcc-kerberos.keytab HTTP/smdw.example.com@GPDB.KRB"
```

2. If you want to authenticate the gpmon user with Kerberos, add the gpmon principal.

```
kadmin.local -q "ktadd -k gpcc-kerberos.keytab gpmon@GPDB.KRB"
```

3. Add principals for all Command Center users:

```
kadmin.local -q "ktadd -k gpcc-kerberos.keytab cc_user1@GPDB.KRB cc_user2@GPDB.KRB"
```

You can enter one or more principals with each `ktadd` command.

4. Enter `quit` to exit `kadmin.local`.
5. Copy the keytab you created to the the host running Command Center, for example:

```
$ scp gpcc-kerberos.keytab gpadmin@<host-name>:/home/gpadmin
```

6. Update any entries required for new principals in the `pg_hba.conf` file and `pg_ident.conf` files on the Greenplum master. See [Update the Greenplum Database pg\\_hba.conf File](#).

## Update the Greenplum Database pg\_hba.conf File

The Greenplum Database `$MASTER_DATA_DIRECTORY/pg_hba.conf` configuration file determines which

authentication methods to use to allow database access.

If you created new Command Center users, you may need to add an entry to allow access via Command Center. The entry for an individual user has this format:

```
host database <user-name> <gpcc CIDR> gss [options]
```

Authentication for the gpmon user needs to be set up in the `pg_hba.conf` file in one of the following ways.

## Basic authentication

The `/home/gpadmin/.pgpass` file contains the password for gpmon to use. See [Changing the gpmon Password](#) for details. An entry in the `pg_hba.conf` file specifies the md5 authentication method for gpmon:

```
local all gpmon md5
```

## Trust authentication

On the Greenplum Database master host only, the gpmon user can access databases without authentication:

```
local all gpmon trust
```

The `/home/gpadmin/.pgpass` file is not needed.

## Kerberos authentication

A Kerberos principal has been created for the gpmon user and added to the Greenplum Database and Command Center keytab files.

```
host all gpmon <gpcc CIDR> gss [options]
```

Remove any existing reject rules for gpmon:

```
host all gpmon <auth-method> reject
```

See [Using Kerberos Authentication](#) for more information about the `pg_hba.conf` file.

## Enable Kerberos for Command Center

Set up Command Center to use the Command Center keytab file you created.

If you are adding Kerberos authentication to an existing Command Center, use the `gpcc` command. For example:

```
$ gpcc --krbenable
```

Enter the Command Center host name and path to the keytab file at the prompts. See the [gpcc Reference](#) for more information.

## Authenticating With Kerberos on the Client Workstation

To use Kerberos Command Center authentication, the user must have authenticated with Kerberos using the `kinit` command-line tool.

The user then accesses the Command Center web server with a URL containing the host name in the format specified in the Command Center service principal and the port number, for example

`http://mdw.example.com:28080`.

The user's web browser must be configured to use the SPNEGO protocol so that it offers the user's Kerberos principal to the web browser. The method for configuring web browsers varies with different browsers and operating systems. Search online to find instructions to enable SPNEGO with your browser and OS.

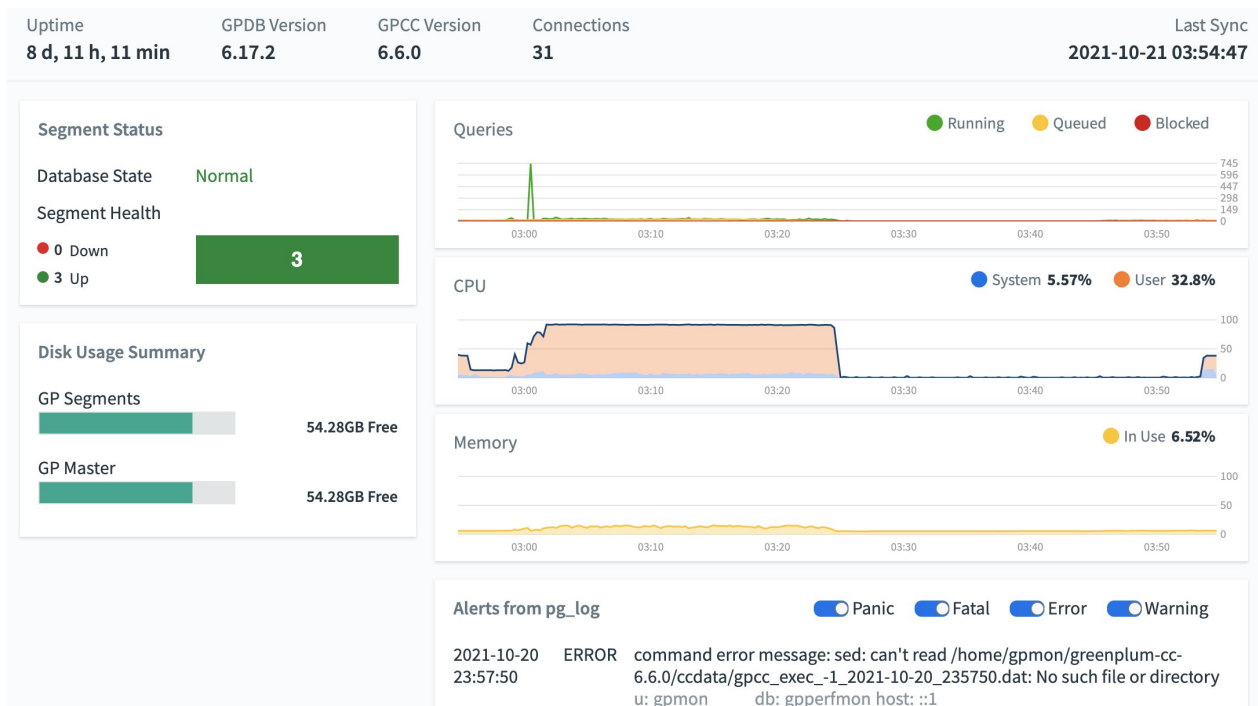
# Monitoring the Greenplum Database System

- [Dashboard](#)
- [Cluster State](#)
- [Segment Status](#)
- [Cluster Metrics](#)
- [Host Metrics](#)
- [Storage Status](#)
- [Recommendations](#)

## Dashboard

The **Dashboard** displays when you first sign in to Tanzu Greenplum Command Center. The **Dashboard** provides a quick view of the current system status, Segment Health, Queries, CPU, Memory, and Disk usage.

Clicking on a panel provides more detailed information about the metric. The Alerts panel shows the most recent messages from the Greenplum Database log file. Some information is available only to Command Center users with Admin or Operator permission level.



## System Information

The following system information is displayed at the top of the page.

### Uptime

The elapsed time since the Greenplum Database system was last started.

### GPDB Version

The version of the Greenplum Database software the monitored cluster is running.

### GPCC Version

The version of the Greenplum Command Center software.

### Connections

The number of active Greenplum Database sessions (client connections).

### Server

The display name for this Greenplum Command Center.

### Last Sync

Date and time the data was last synchronized. The Command Center user interface updates views with live data every 15 seconds.

## Segment Health

The **Segment Health** section of the Dashboard provides a quick overview of the status of the database system and segments this Command Center monitors.


## Database State

**Database State** is the current state of the Greenplum Database system. Following are some possible database states:

- **Normal:** The database is functioning with no major errors or performance issues.
- **Segment(s) Down:** The database is in change-tracking mode or resync mode. Overall performance and system reliability is greatly reduced. See the *Tanzu Greenplum Database System Administrator Guide* for information about resolving this condition.
- **Standby Down:** The master standby instance cannot be reached.
- **Standby Not Syncing:** The master standby instance is not synchronizing with the master instance.
- **Database Unreachable:** The Greenplum Performance Monitor agent cannot connect to the database. The database is likely down. See the *Tanzu Greenplum Database System Administrator Guide* for troubleshooting information.
- **Unbalanced:** Some segments are not running in their preferred roles. That is, primaries are running as mirrors and mirrors are running as primaries, resulting in unbalanced processing.
- **Resyncing:** The database is performing a recovery or rebalance operation.



**Note:** An error message or state other than the above may be an indication of a network connectivity problem, or of another undetected problem in the Greenplum



Database system. Run the `gpstate` utility on the master host to determine if there are issues to troubleshoot in the Greenplum Database system.

## Segment Status

The bar graph in the **Segment Status** section shows the up or down status of all database segments in your Tanzu Greenplum Database system. A color indicator and associated number indicate the number of database segments that are currently in that particular state. Segments can have the following states:

- **Up** (Green)
- **Down** (Red)

Clicking the **Segment Status** panel displays the [Segment Status](#) Command Center page.

## Disk Usage Summary

This chart displays total disk usage and disk available for the Greenplum master host and segment hosts at the last synchronization. Hover over the chart to see the amount of disk used, free, and total.

## Queries

This graph displays a summary view of active and queued queries for the last 60 minutes. Click on the colored dot next to the **Running** or **Queued** label to toggle the line on or off. At least one line must be visible at all times. Hover over the graph to display the number of queries for each visible line at that point in time.

## CPU

This graph displays average CPU usage across the entire cluster, for the last 60 minutes. The graph displays separate lines for system processes and user processes. The user CPU usage includes the Greenplum database master, standby, and segment processes. Click on the colored dot next to the **System** or **User** label to toggle that line on or off. At least one line must be visible at all times.

Hovering the cursor over a line in the graph displays a small window with the percentage of CPU used at that point in time for the visible lines and the total if both the system and user lines are visible.

## Memory

This graph displays the average percent of memory used across the entire cluster over the last 60 minutes. Hover over the line to display the percent of memory used at that point in time.

## Alerts

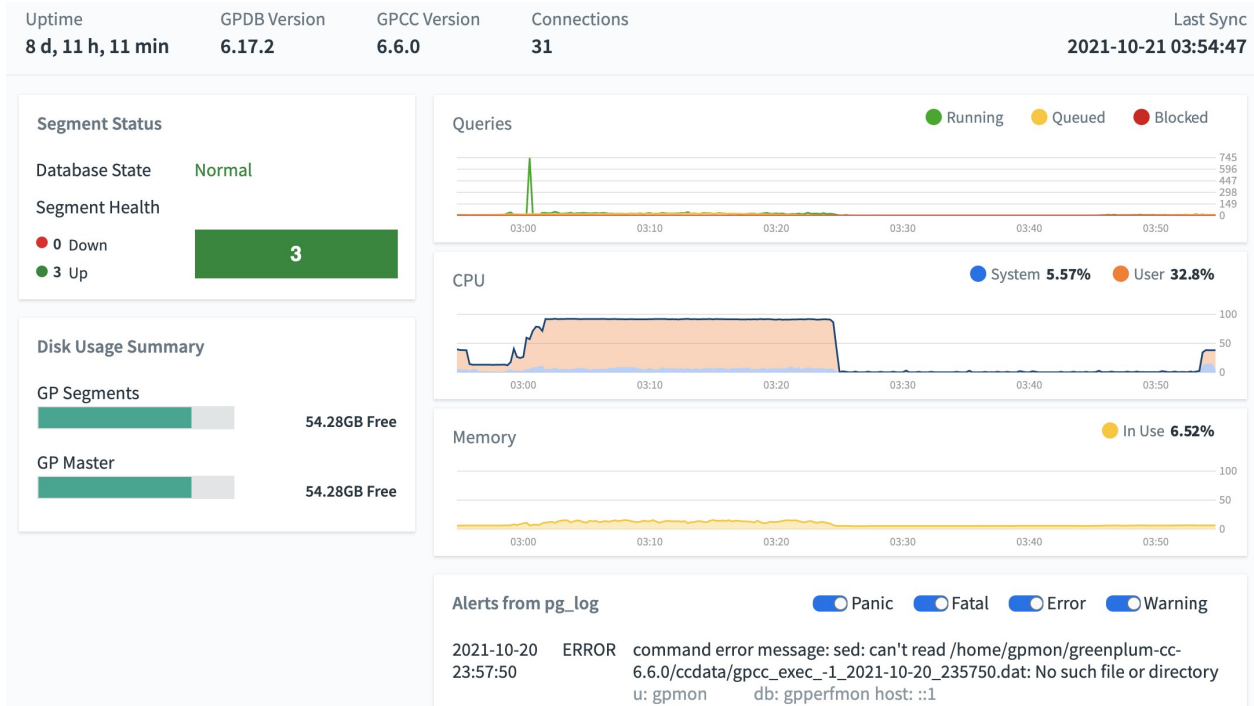
*Admin and Operator permission levels only*

The **Alerts** panel displays recent messages from the Greenplum Database `pg_log` log file. The panel is updated at each synchronization. Filter the messages by severity level using the controls at the top right of the panel.



## Greenplum Database Cluster State

The Greenplum Command Center **Dashboard** provides a quick view of the current system status, Segment Health, Queries, CPU, Memory, and Disk usage. Clicking on a panel provides more detailed information about that metric. The **Alerts** panel shows the most recent messages from the Greenplum Database log file. Some information is available only to Command Center users with Admin or Operator permission level.



## System Information

The following system information is displayed at the top of the page.

### Uptime

The elapsed time since the Greenplum Database system was last started.

### GPDB Version

The version of the Greenplum Database software the monitored cluster is running.

### GPCC Version

The version of the Greenplum Command Center software.

### Connections

The number of active Greenplum Database sessions (client connections).

### Server

The display name for this Greenplum Command Center.

### Last Sync

Date and time the data was last synchronized. The Command Center user interface updates views with live data every 15 seconds.

## Segment Health

The **Segment Health** section of the Dashboard provides a quick overview of the status of the database system and segments this Command Center monitors.

### Database State

**Database State** is the current state of the Greenplum Database system. The state can be one of the following:

- **Normal:** The database is functioning with no major errors or performance issues.
- **Segment(s) Down:** The database is in change-tracking mode or resync mode. Overall performance and system reliability is greatly reduced. See the *Tanzu Greenplum Database System Administrator Guide* for information about resolving this condition.
- **Database Unreachable:** The Greenplum Performance Monitor agent cannot connect to the database. The database is likely down. See the *Tanzu Greenplum Database System Administrator Guide* for troubleshooting information.
- **Unbalanced:** Some segments are not running in their preferred roles. That is, primaries are running as mirrors and mirrors are running as primaries, resulting in unbalanced processing.
- **Resyncing:** The database is performing a recovery or rebalance operation.

### Segment Status

The bar graph in the **Segment Status** section shows the up or down status of all database segments in your Tanzu Greenplum Database system. A color indicator and associated number indicate the number of database segments that are currently in that particular state. Segments can have the following states:

- **Up** (Green)
- **Down** (Red)

Clicking the **Segment Status** panel displays the [Segment Status](#) Command Center page.

## Disk Usage Summary

This chart displays total disk usage and disk available for the Greenplum master host and segment hosts at the last synchronization. Hover over the chart to see the amount of disk used, free, and total.

## Queries

This graph displays a summary view of active and queued queries for the last 60 minutes. Click on the colored dot next to the **Running** or **Queued** label to toggle the line on or off. At least one line must be visible at all times. Hover over the graph to display the number of queries for each visible line at that point in time.

## CPU

This graph displays average CPU usage across the entire cluster, for the last 60 minutes. The graph displays separate lines for system processes and user processes. The user CPU usage includes the Greenplum database master, standby, and segment processes. Click on the colored dot next to the **System** or **User** label to toggle that line on or off. At least one line must be visible at all times.

Hovering the cursor over a line in the graph displays a small window with the percentage of CPU used at that point in time for the visible lines and the total if both the system and user lines are visible.

## Memory

This graph displays the average percent of memory used across the entire cluster over the last 60 minutes. Hover over the line to display the percent of memory used at that point in time.

## Alerts

*Admin and Operator permission levels only*

The **Alerts** panel displays recent messages from the Greenplum Database `pg_log` log file. The panel is updated at each synchronization. Filter the messages by severity level using the controls at the top right of the panel.

## Segment Status

The **Segment Status** page provides a health overview for the Greenplum Database segments and details for each primary and mirror segment.

Segment Status <small>Role and health related information</small>									
Last Sync 2020-04-21 18:43:23									
<b>Segment Summary</b>  Database State <b>Normal</b> Mirrors Acting as Primary 0 Recommended Actions <b>None</b>					<b>Segment Health</b>  <div> <div>Status</div> <div> <div>0 Down</div> <div>8 Up</div> </div> </div> <div> <div>Replication Mode</div> <div> <div>0 Not Syncing</div> <div>0 Resyncing</div> <div>0 Change Tracking</div> <div>8 Synced</div> </div> </div> <div> <div>Preferred Role</div> <div> <div>0 Not Preferred</div> <div>8 Preferred</div> </div> </div>				
Hostname	Address	Port	DBID	Content ID	Status	Role	Preferred Role	Replication Mode	Last Event   [Total]
sdw2	sdw2	6000	4	2	Up	Primary	Primary	Synced	Apr 21st, 17:21:06 [1]
sdw1	sdw1	7000	9	2	Up	Mirror	Mirror	Synced	Apr 21st, 17:21:06 [2]
sdw1	sdw1	6000	2	0	Up	Primary	Primary	Synced	Apr 21st, 17:21:06 [1]
sdw2	sdw2	7000	7	0	Up	Mirror	Mirror	Synced	Apr 21st, 17:21:06

## Segment Summary

Greenplum Database is most efficient when all segments are operating in their preferred roles. The **Segment Summary** panel tells you the overall segment status and if any mirrors are acting as

primaries.

The **Segment Summary** panel provides the following information:

## Database State

The database state can be one of the following:

- **Normal:** The database is functioning with no major errors or performance issues.
- **Segment(s) Down:** The database is in change-tracking mode or resync mode. Overall performance and system reliability is greatly reduced. See the *Tanzu Greenplum Database System Administrator Guide* for information about resolving this condition.
- **Database Unreachable:** The Greenplum Performance Monitor agent cannot connect to the database. The database is likely down. See the *Tanzu Greenplum Database System Administrator Guide* for troubleshooting information.
- **Unbalanced:** Some segments are not running in their preferred roles. That is, primaries are running as mirrors and mirrors are running as primaries, resulting in unbalanced processing.
- **Resyncing:** The database is performing a recovery or rebalance operation.

## Mirrors Acting as Primary

The number of mirror segments acting as primary segments.

## Recommended Actions

Suggests actions to perform to restore the cluster to balance. These include:

- Recover and Rebalance
- Rebalance

These actions are executed from the command line using the `gprecoverseg` Greenplum management utility. See `gprecoverseg` in the *Tanzu Greenplum Database Utility Reference* for more information.

## Total Segments

The total number of primary and mirror segments in the Greenplum cluster.

## Segment Hosts

The total number of segment hosts in the Greenplum cluster.

## Segment Health

The **Segment Health** panel contains charts for Greenplum Database segments' status, replication mode, and preferred roles.

## Status

Numbers of segments that are down and up.

## Replication Mode

A chart that shows the number of segments in each of the possible replication modes.

- **Not Syncing:** The primary segment and mirror segment are active and all changes to the primary segment have been copied to the mirror using a file block replication process.
- **Change Tracking:** If a primary segment is unable to copy changes to its mirror segment using the file replication process, it logs the unsent changes locally so they can be replicated when the mirror again becomes available. This can happen if a mirror segment goes down or if a primary segment goes down and its mirror segment automatically assumes the primary role.
- **Resyncing:** When a down segment is brought back up, administrators initiate a recovery process to return it to operation. The recovery process synchronizes the segment with the active primary and copies the changes missed while the segment was down.
- **Synced:** Once all mirrors and their primaries are synchronized, the system state becomes synchronized.

## Preferred Roles

The red portion of the Preferred Role chart shows the numbers of segments that not operating in their preferred primary or mirror roles. If the chart is not solid green, the performance of the Greenplum cluster is not optimal.

Primary and mirror segments are distributed evenly among the segment hosts to ensure that each host performs an equivalent share of the work and primary segments and their mirror segments reside on different segment hosts. When a primary segment goes down, its mirror on another host in the cluster automatically assumes the primary role, increasing the number of primary segments running on that host. This uneven distribution of the workload will affect query performance until the down segment is restored and the segments are returned to their original, preferred, roles.

## Segment Table

The table at the bottom of the **Segment Status** page contains a detailed row for every primary and mirror segment in the Greenplum Cluster. The table has the following columns for each segment:

**Hostname**

The name of the segment host where the segment is running.

**Address**

The network interface on the segment host for the segment.

**Port**

The port number assigned to the segment.

**DBID**

The unique identifier for the segment instance.

**ContentID**

The content identifier for the segment, from 0 to the number of segments minus 1. A primary segment and its mirror have the same ContentID. The master and standby master, which have

ContentID -1, are excluded from the table.

#### Status

"UP" if the segment is running, "DOWN" if the segment has failed or is unreachable.

#### Role

The segment's current role, either "primary" or "mirror".

#### Preferred Role

The segment's intended role, either "primary" or "mirror".

#### Replication Mode

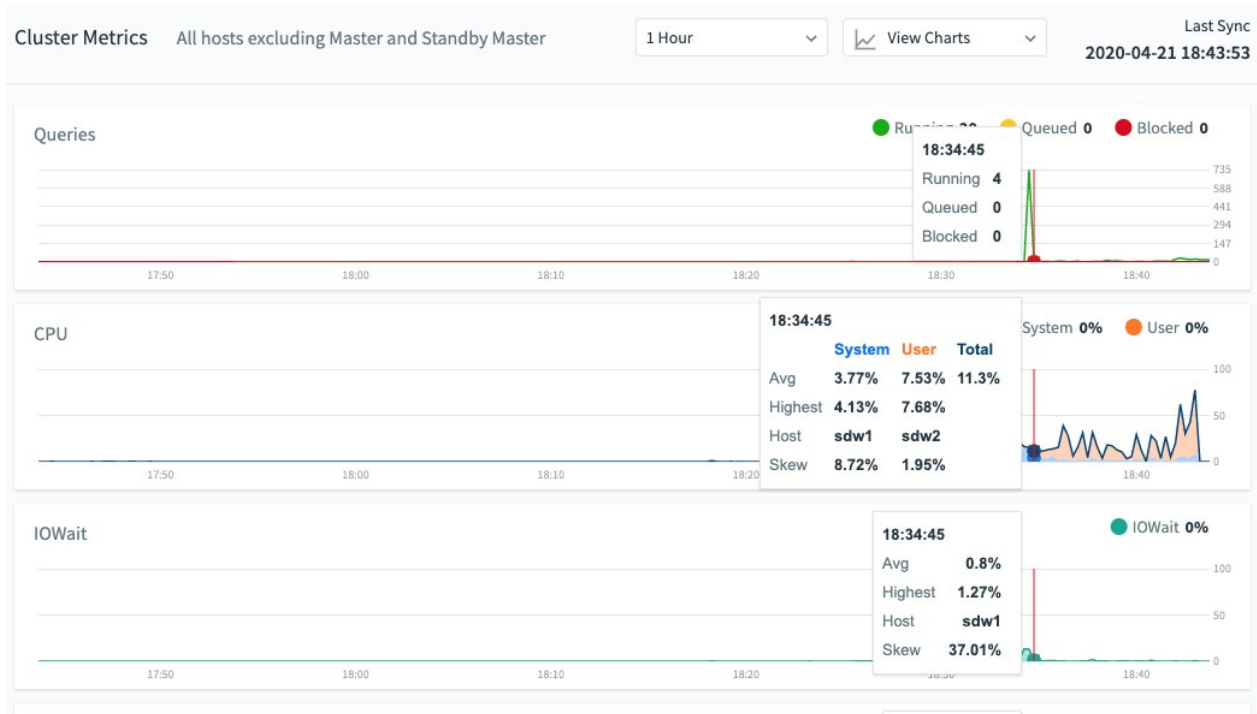
The replication status for the segment. See [Segment Health](#) for possible values.

#### Last Event[Total]

The date and time of last segment health-related activity. Click to display a list of recent events.

## Cluster Metrics

The **Cluster Metrics** page shows consolidated statistics for all segment hosts in the Greenplum cluster. Master and standby master hosts are excluded from the metrics.



Use the **Show/hide Charts** control to choose which charts to display.

Choose the time period to display using the control in the top right corner of the screen: 1 hour, 4 hours, 1 day, or 3 days. The right side of the chart corresponds to the **Last Sync** time shown in the top right of the page.

Hover over any of the charts to see values for that point in time in pop-up boxes. The charts are synchronized so that hovering over any chart shows the same point in time in all charts.

- The master and standby master hosts are excluded from the metrics.

- When the metric is an average of values collected from all segment hosts, the host with the highest value is identified in the pop-up box.
- Solid lines trace the average value for all segment hosts. The stacked graphs represent the highest values for the metric at each point in time.
- The current value of a metric is shown in the upper right corner of its chart.

The page has charts for the following metrics:

#### Queries

The number of queries running and the number of queries queued to run.

#### CPU

The percentage CPU used by system processes and the percentage CPU used by user processes. The chart represents the total of system and user CPU usage across the hosts. The system percentage includes IOWait time. Note that this is different than the CPU system percentage on the **Host Metrics** page, where IOWait time is *not* included in the system time.

#### IOWait

The percentage of CPU used waiting on IO requests.

#### Memory

Percentage of memory in use.

Memory is calculated as follows:

Total = MemTotal

Free = MemFree + Buffers + Cached

Used = MemTotal - Free

#### Disk I/O

Disk read and write rates in megabytes per second.

#### Network

Network I/O read and write rates in megabytes per second. Network metrics include traffic over all NICs (network interface cards), including internal interconnect and administrative traffic.

#### Load

System load average for 1-minute, 5-minute, and 15-minute periods.

#### Swap

Percentage of swap space used.

## Host Metrics

The **Host Metrics** page displays a table of the hosts in the cluster with statistics collected at the most recent quantum interval.

Host Metrics <span>Realtime statistics by server</span>							Last Sync <span>2020-04-21 18:45:08</span>	
Hostname ^	CPU Total/Sys/User/IOWait (%)	Memory in Use(%)	Disk R(MB/s) Skew	Disk W(MB/s) Skew	Net R(MB/s) Skew	Net W(MB/s) Skew		
sdw1	87.50	2.95	0	38.86	47.75	47.10		
sdw2	87.31	2.89	0	25.73	45.91	46.24		
mdw	9.85	2.22	0	0.02	0.11	0.43		

At the top right, **Last Sync** displays the time the statistics were last updated.

Click a column header to sort the table by that column. Click again to toggle between ascending and descending sort. Master and standby hosts are not included in the sort and are always displayed following the sorted list of segment hosts.

For each server, the following columns are displayed:

Hostname

The hostname name of the server.

CPU Total/Sys/User/IOWait (%)

The CPU column shows the total percentage of CPU used for system processes, user processes, and IOWait. Hover over the table cell to show the percentages for each of these CPU modes, as well as the idle time.

When comparing Command Center CPU values with other unix utilities like `top`, there are slight differences.

To compare Command Center CPU user time with `top`, adjust the `top` CPU user time to be: CPU user time + `nice` time. To compare CPU sys time, before Command Center 4.9/6.1, adjust the `top` utility CPU sys time to be: sys time + iowait time. After 4.10/6.2, the values are similar, for example CPU sys = sys and CPU IOWait = iowait.

Memory In Use (%)

The percentage of host memory in use is displayed next to a graph illustrating the memory in use and available. Hover over the table cell to see memory used and available in gigabytes.

Memory is calculated as follows:

Total = MemTotal

Free = MemFree + Buffers + Cached

Used = Total - Free Disk R (MB/s) | Skew Disk read rate in megabytes per second is displayed next to a graph of calculated disk read skew. Hover over the table cell to see a Low/Medium/High rating for disk skew.

Disk W (MB/s) | Skew Disk write rate in megabytes per second is displayed next to a graph of calculated disk write skew. Hover over the table cell to see a Low/Medium/High rating for disk write skew. Net R (MB/s) | Skew Network read rate in megabytes per second is displayed next to a graph of calculated network read skew. Hover over the table cell to see a Low/Medium/High rating for network read skew. Net W (MB/s) | Skew Network write rate in megabytes per second is displayed next to a graph of calculated network write skew. Hover over the table cell to see a Low/Medium/High rating for network write skew.



## About Skew Calculations

Disk and Network skew ratings are calculated as each server's standard deviation from the mean calculated from all segment hosts.

Low

Value is within 1 standard deviation from the mean. (Note: if the variance of the set is less than 3, skew is considered low regardless of deviation from mean.)

Moderate

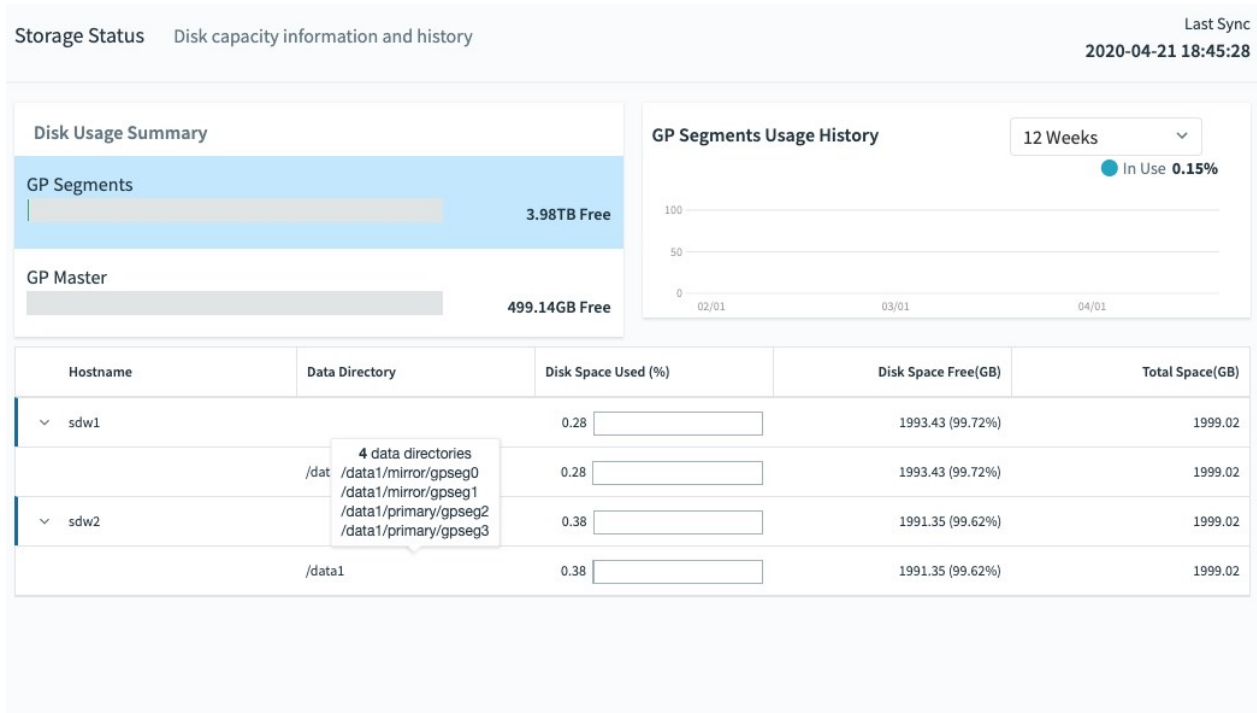
Value is between 1 and 2 standard deviations from the mean.

Very High

Value is greater than 3 standard deviations from the mean.

## Storage Status

The **Storage Status** page shows current historical disk usage for Greenplum master and segment hosts.



## Disk Usage Summary

You can see current disk space in use, space free, and total space in the **Disk Usage Summary** panel. Disk space metrics for the segment hosts (**GP Segments**) and the master (**GP Masters**) are shown in separate bar charts.

The **GP Segments** bar chart shows combined disk space for all segments.

The **GP Masters** bar chart shows combined disk space for master and standby master.

Click either of the summary charts to see details by host: space used, free, and total in gigabytes and as a percentage of the total.

Click a host name to display a list of directories on the host file system where partitions containing one or more Greenplum Database data directories are mounted. Hover over the directory name to see a list of the Greenplum data directories the partition contains.

## GP Segments Usage History

The **GP Segments Usage History** panel presents a chart of percentage of disk space in use for the time period set by the control in the panel header.

Hover over the chart to see the percentage disk in use by all Greenplum Database segments at any given point in time.

## GP Masters Usage History

The **GP Masters Usage History** panel presents a chart of percentage of disk space in use by the master and standby masters for the time period set by the control in the panel header.

Hover over the chart to see the percentage disk in use at any given point in time.

## Recommendations

Use the **Recommendations** page to find tables that require maintenance. Command Center users with Admin permission can make changes to this page.

## Scheduling Scans

Schedule a period of time during which Command Center scans databases for tables that need to be vacuumed, analyzed, or redistributed.

Command Center scans tables of type heap, append-optimized (AO), and append-optimized/column-oriented (AO/CO) that have size greater than zero. Views and external tables are not scanned.

Tables that have been updated recently or have never been scanned are scanned first.

It is best to schedule the scan during a down-time or a period of low activity to minimize the effect of the scan on database performance.

You can set the date and time the scan begins and the duration of the scan. Check **Repeat** to set the scan to repeat daily, or weekly on selected days of the week.

**Recommendations** Generate recommended database maintenance actions Current Time  
2020-04-20 23:57:17

Database Schema

All All **SELECT** ⚙ Repeat Settings

Scanned Tables **3114** Start Time **2020-04-14 22:00** Duration **00:05:18**

Command Center starts scanning automatically at the scheduled time. A message is displayed on every Command Center page while the scan is in progress. It may not be possible to scan all tables in the scheduled period, but when the next scan begins, Command Center resumes scanning where the previous scan left off, until all tables have been scanned. Then the scan starts over.

The recommendation tables always list results from the most recent scan.

Click **Repeat Settings** to view a history of scans or to update the scan schedule.

Choose a database and schema and click **SELECT** to view recommendations. Recommendations are reported in four reports:

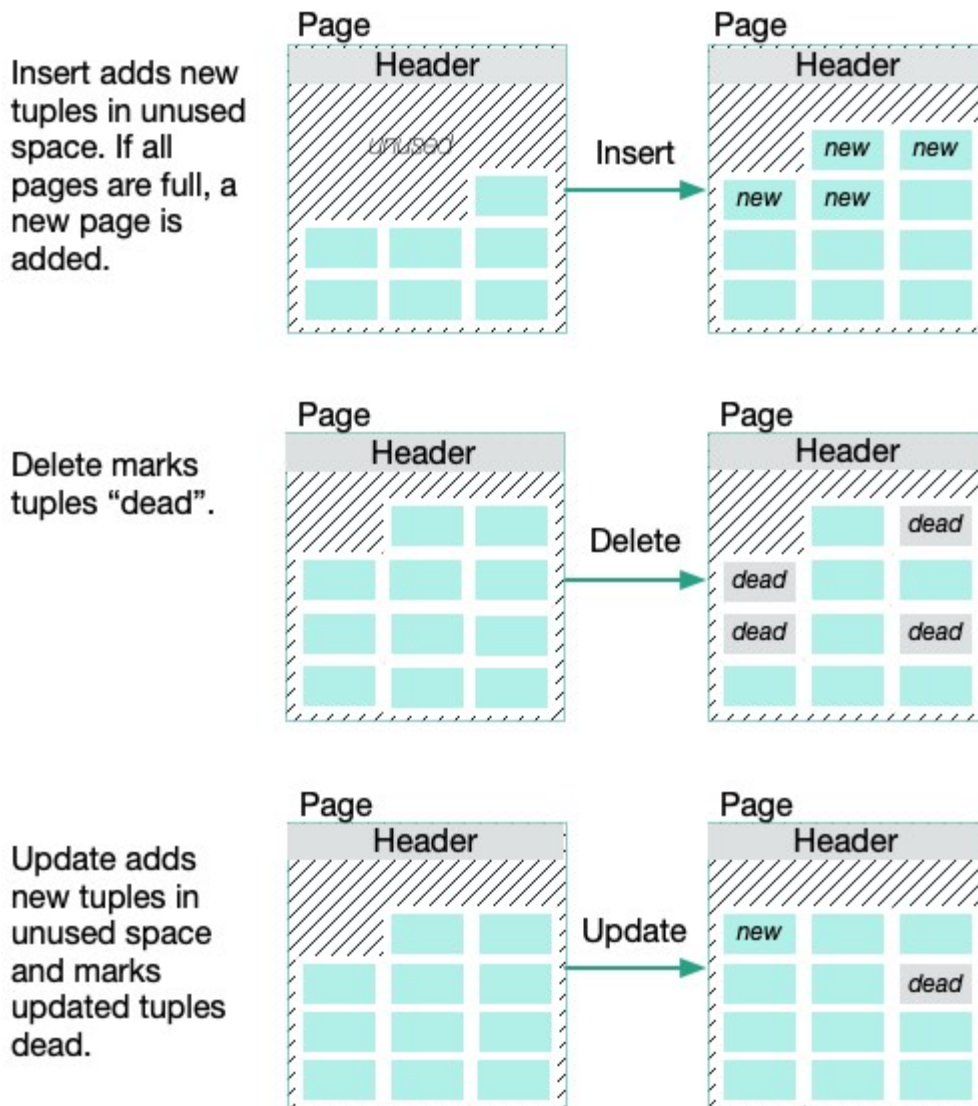
- [Bloat](#)
- [Accuracy](#)
- [Skew](#)
- [Age](#)

## Bloat

Bloat is a measure of the amount of unused space and space occupied by dead tuples in a table.

Greenplum Database uses PostgreSQL multi-version concurrency control (MVCC) to manage concurrent transactions. Table storage is organized into fixed size pages containing a header, an array of item pointers, unused space, and tuples (rows).

- Inserting a tuple into a table adds a new tuple and reduces the amount of unused space on the page. Tuples are allocated from the end of the unused space.
- Deleting tuples marks them dead, but does not remove them from the page.
- Updating a tuple inserts a new live tuple and marks the original tuple dead.
- New pages are added to the table when there is insufficient unused space to add new tuples.

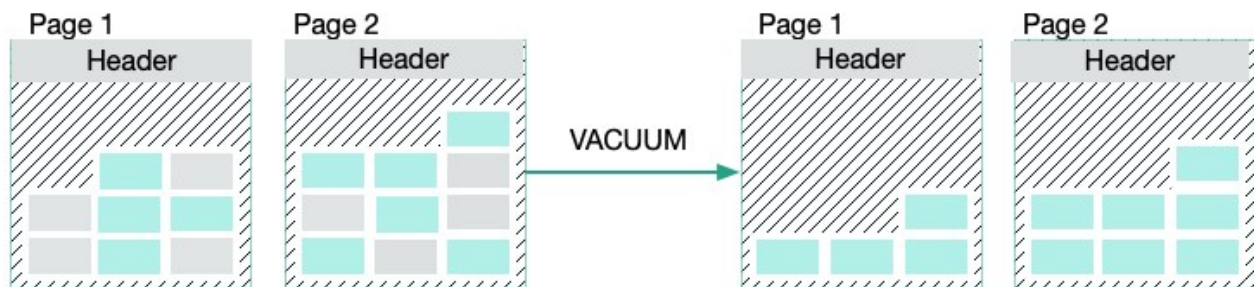


Over time, the dead tuples accumulate, consuming a greater proportion of the table. New pages are added when unused space is depleted, increasing the file size of the table. Scanning a table with a large amount of bloat requires additional I/O, and has a detrimental effect on query performance.

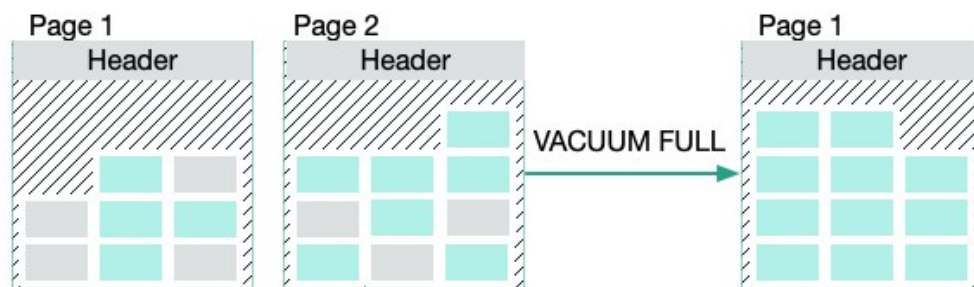
The `VACUUM` command removes the dead tuples, increasing the amount of unused space available for new live tuples. `VACUUM` only reduces the size of the table on disk if the unused space is at the end of the table. Command Center by default recommends vacuuming a table when dead tuples occupy more than 10% of the space. You can specify a different threshold by setting the `bloat_threshold` parameter in the `gpmetrics` configuration file. See the [gpmetrics Configuration File Reference](#) for details.

The `VACUUM FULL` command removes dead tuples and compacts the table to the minimum number of pages by packing live tuples on pages. By default, Command Center recommends `VACUUM FULL` for heap tables when dead tuples and unused space occupy greater than 50% of the space. You can specify a different threshold for the `VACUUM FULL` recommendation by setting the `bloat_vacuum_full_threshold` parameter in the `gpmetrics` configuration file. See the [gpmetrics Configuration File Reference](#) for details.

VACUUM removes dead tuples, increasing unused space on the page. When Command Center recommends VACUUM, **Bloat Space** is the amount of space used by dead tuples.



VACUUM FULL removes dead tuples and combines live tuples on pages to decrease file size. When Command Center recommends VACUUM FULL, **Bloat Space** is the total of unused space and space used by dead tuples.



For append-optimized (AO) and append-optimized/column-oriented (AO/CO) tables, Command Center recommends `VACUUM`. Vacuuming AO and AO/CO tables has the same effect as `VACUUM FULL` for heap tables.

`VACUUM FULL` is an expensive operation, requiring an `ACCESS EXCLUSIVE` lock and additional disk space, and it should be avoided, especially for very large tables. Running `VACUUM` frequently can prevent the need to run `VACUUM FULL`.

For more information about `VACUUM` and `VACUUM FULL`, see [VACUUM](#) in the Greenplum Database SQL Commands Reference.

The **Bloat Rank** table lists tables by the estimated amount of bloat space, from greatest to least.

Recommendations

Generate recommended database maintenance actions

Current Time  
2020-04-15 01:22:59

Database

gptest

Schema

All

SELECT

Repeat Settings

Scanned Tables 3114

Start Time 2020-04-14 22:00

Duration 00:05:18

Bloat Rank

EXPORT

Tables that have high level of suspected bloat, and are in need of vacuum.

Rank	Database	Schema	Relation Name	Type	Size	Bloat Space	Action	Last Vacuumed	Last Accessed
1	recom_test	public	bloat_heap	Heap	17.85 GB	12.67 GB	Vacuum-full	--	2020-04-10 15:30:46
2	recom_test	public	pt_heap_1_pr...	Heap	15.82 GB	10.63 GB	Vacuum-full	--	2020-04-10 15:30:46
3	recom_test	public	bloat_ao	AO	14.55 GB	10.38 GB	Vacuum	--	2020-04-10 15:30:46
4	recom_test	public	pt_ao_1_prt...	AO	13.15 GB	8.85 GB	Vacuum	--	2020-04-10 15:30:46
5	recom_test	public	pt_heap_1_pr...	Heap	11.69 GB	8.66 GB	Vacuum-full	--	2020-04-10 15:30:46
6	recom_test	public	bloat_aocs	AO/CO	10.97 GB	7.83 GB	Vacuum	--	2020-04-10 15:30:46
7	recom_test	public	pt_ao_1_prt...	AO	9.73 GB	7.23 GB	Vacuum	--	2020-04-10 15:30:46
8	recom_test	public	pt_aocs_1_pr...	AO/CO	10.15 GB	6.83 GB	Vacuum	--	2020-04-10 15:30:46

The **Bloat** table has these columns:

Rank

Tables in the list are ranked by the estimated amount of bloat

Database

Name of the database containing the table

Schema

Name of the schema to which the table belongs

Relation Name

Name of the table

Type

Storage type for the table, for example heap or AO (append-optimized)

Size

Total size of the table on all segments

Bloat Space

The amount of space that can be freed with the recommended action. If the recommended action is **VACUUM**, **Bloat Space** is the amount of space occupied by dead rows. If **VACUUM FULL** is recommended, **Bloat Space** is the total unused space and space occupied by dead rows.

Action

Running **VACUUM FULL** decreases the size of the table and is recommended only when the table size would be reduced by at least 50%. Running **VACUUM** removes only dead tuples, increasing unused space.

### Last Vacuumed

Time the table was last vacuumed. This is the end of the Command Center sampling period during which the vacuum completed.

### Last Accessed

Time the table was last updated ([DELETE](#), [INSERT](#), or [UPDATE](#) operation).

## Accuracy

The **Accuracy% Rank** table lists tables ranked by **Est.Inaccuracy**. A higher **Est.Inaccuracy** is an indication that you should run [ANALYZE](#) on the table to update the table statistics. When statistics are inaccurate, the optimizer can generate an inefficient query plan.

The **Est.Inaccuracy** metric is calculated from query history by comparing the query plan estimate of rows and the actual number of rows returned by table scans. The calculation includes scan operations (query plan scan nodes) for queries that completed in the last 30 days and that started after the scanned table was last analyzed. The formula for inaccuracy is:

```
inaccuracy = (MAX(plan_rows, actual_rows) - MIN(plan_rows, actual_rows)) / MAX(plan_rows, actual_rows)
```

Scans without conditions ([WHERE](#) clauses) are considered more predictive of row count accuracy, so if the query history has both scans without conditions and scans with conditions, then scans without conditions are weighted more heavily.

By default, the **Accuracy%** report includes tables with a calculated inaccuracy of at least 10%. You can specify a different threshold by setting the [accuracy\\_threshold](#) parameter in the [gpmetrics](#) configuration file. See the [gpmetrics Configuration File Reference](#) for details.

Accuracy% Rank							
Tables that have lowest estimate accuracy, and are in need of analyze.							
						<a href="#">EXPORT</a>	^
Rank	Database	Schema	Relation Name	Row Count	Est. Inaccuracy%	Last Analyzed	Last Modified
1	gpperfmon	gpmetrics	<a href="#">gpcc_disk_history_1_p...</a>	9095	100.00%	2019-11-23 21:14:09	--
2	gpperfmon	gpmetrics	<a href="#">gpcc_disk_history_1_p...</a>	8615	100.00%	2019-11-23 21:15:08	--
3	gpperfmon	gpmetrics	<a href="#">gpcc_disk_history_1_p...</a>	7365	100.00%	2019-11-23 21:13:34	--
4	gpperfmon	gpmetrics	<a href="#">gpcc_disk_history_1_p...</a>	6835	100.00%	2019-11-23 21:15:07	--
5	gpperfmon	gpmetrics	<a href="#">gpcc_disk_history_1_p...</a>	2660	100.00%	2019-11-23 21:14:23	--
6	gpperfmon	public	<a href="#">diskspace_history_1_p...</a>	0	100.00%	2019-11-23 21:16:09	--
7	gpperfmon	public	<a href="#">diskspace_history_1_p...</a>	0	100.00%	2019-11-23 21:09:18	--

Skew% Rank							
Tables that are most skewed and are in need of re-distribution.							
						<a href="#">EXPORT</a>	^

The **Accuracy%** table has these columns:

Rank

Tables in the list are ranked by **Est.Accuracy%**, from least to greatest.

Database

Name of the database containing the table

Schema



Name of the schema to which the table belongs

Relation Name

Name of the table

Row Count

Number of rows in the table

Est.Accuracy%

Estimate of statistics accuracy from 0.0 to 1.0

Last Analyzed

Time the table was last analyzed. This is the end of the Command Center sampling period during which the analyze operation was completed.

Last Modified

The most recent timestamp among `last_ins`, `last_del`, and `last_upd`.

## Skew

Skew is a measure of how evenly table rows are distributed to Greenplum Database segments. When some tables are distributed unevenly, queries can take longer to complete.

**Skew%** is a value between 0% and 100%, inclusive. The higher the value, the more skew is present in the distribution of the table. Given a table *T* distributed among *n* segments, *S*<sub>1</sub> is the number of tuples on segment 1 and *S*<sub>*n*</sub> is the number of tuples on segment *n*:

$$\begin{aligned} \text{Avg} &= (S_1 + S_2 + \dots S_n) / n \\ \text{Max} &= \text{MAX}(S_1, S_2, \dots S_n) \\ \text{Skew\%} &= (1 - \text{Avg} / \text{Max}) / (1 - 1/n) * 100\% \end{aligned}$$

A high **Skew%** for a table is usually caused by a poor choice of distribution key. The distribution key should be a column (or columns) with high cardinality, such as a unique ID. Columns that have a limited number of possible values make poor distribution keys and will result in skew. You can distribute the table `RANDOMLY` to ensure an even distribution if there is no suitable column.

If Command Center recommends redistributing a table, use the `ALTER TABLE` command:

```
ALTER TABLE <table-name> SET DISTRIBUTED BY (<column>, ...);
```

The table will be redistributed using the new distribution key.

Use this command to change the distribution policy for a table to `RANDOMLY`:

```
ALTER TABLE <table-name> SET DISTRIBUTED RANDOMLY;
ALTER TABLE <table-name> SET WITH (REORGANIZE=TRUE);
```

The second `ALTER TABLE` command is needed to redistribute the table. Without it, only newly added rows will be distributed randomly.

By default, the **Skew%** report includes tables with a calculated skew of at least 10%. You can specify a different skew threshold by setting the `skew_threshold` parameter in the `gpmetrics` configuration



file.

A table with a small number of rows can have a high **Skew** value because there are insufficient rows to achieve an even distribution. Command Center excludes such tables from the **Skew%** report by filtering for tables that have at least  $128 * N\text{Seg}$  rows, where **NSeg** is the number of segments in the Greenplum system. You can change this default by setting the `skew_tuple_per_segment` value in the `gpmetrics` configuration file.

See the [gpmetrics Configuration File Reference](#) for more information about setting the `skew_threshold` and `skew_tuple_per_segment` parameters.

Skew% Rank								
Tables that are most skewed and are in need of re-distribution.								
							<a href="#">EXPORT</a>	^
Rank	Database	Schema	Relation Name	Owner	Row Count	Skew%	Distributed By	Last Accessed
1	gpptestmon	gpmetrics	<a href="#">gpcc_table_info_hi...</a>	gpmon	105201	100.00%	(ctime)	--
2	postgres	public	<a href="#">heap_page</a>	gpadmin	74935	100.00%	(id)	2020-05-06 12:24:25
3	postgres	s1h	<a href="#">tb1</a>	gpmon	10000	100.00%	(id)	2020-05-06 12:24:25
4	gpptest	tpcds	<a href="#">sales</a>	gpadmin	10002239	100.00%	(id)	--
5	recom_test	public	<a href="#">skew_heap</a>	gpadmin	2001296	90.02%	(id)	2020-05-06 12:24:25
6	recom_test	public	<a href="#">skew_ao</a>	gpadmin	2000000	90.00%	(id)	2020-05-06 12:24:25
7	recom_test	public	<a href="#">skew_aocs</a>	gpadmin	2000000	90.00%	(id)	2020-05-06 12:24:25
8	recom_test	public	<a href="#">skew_aocs_z</a>	gpadmin	2000000	90.00%	(id)	2020-05-06 12:24:25
9	gpptestmon	table_size	<a href="#">sales_skew</a>	gpadmin	10000	68.45%	(id)	2020-05-06 12:24:25
10	gpptestmon	gpmetrics	<a href="#">gpcc_table_info_hi...</a>	gpmon	3343789	66.67%	(ctime)	2020-05-11 22:42:37
11	gpptestmon	gpmetrics	<a href="#">gpcc_table_info_hi...</a>	gpmon	8795082	60.71%	(ctime)	--

The **Skew%** table has these columns:

Rank

Tables in the list are ranked by Skew%, from greatest to least

Database

Name of the database containing the table.

Schema

Name of the schema to which the table belongs

Relation Name

Name of the table

Row Count

Number of rows in the table

Skew%

Amount of skew, between 0% and 100%. A lower percentage is best for Greenplum Database query performance.

Distributed By

The current distribution key for the table.

## Age

The **Age Rank** table lists tables ranked by age. A higher **Age** is an indication that you should run **VACUUM FREEZE** on the table in order to prevent database failures caused by transaction ID wraparound.

As tables age, the risk of [transaction ID \(XID\) wraparound](#) -- also known as XID exhaustion -- increases, which can result in catastrophic data loss. You can prevent this by running **VACUUM FREEZE** on older tables. **VACUUM FREEZE** marks the table's rows as frozen in order to reduce the number of past transactions, thereby diminishing the risk of issues caused by XID wraparound.

You can control how old a table must be for Command Center to issue a **VACUUM FREEZE** recommendation in the Age Rank report by configuring the `age_ratio` property in the `gpcc.conf` file. For more information, see the [gpmetrics Configuration File Reference](#).

Once the user has run **VACUUM FREEZE** on a table, that table's row in the **Age Rank** table is highlighted in grey.

To export the **Age Rank** list to a text file, click **Export**.

The **Age Rank** table lists table age, from oldest to youngest.

Rank	Database	Schema	Relation Name	Size	Age	Last Freeze	Last Accessed
1	tpcds_runner	pg_catalog	<a href="#">pg_default_acl</a>	256.00 KB	174.13%	--	--
2	gpperfmon	pg_catalog	<a href="#">pg_default_acl</a>	256.00 KB	174.13%	--	--
3	gpperfmon	pg_catalog	<a href="#">pg_type</a>	1.75 MB	174.13%	--	2022-04-09 01:30:47
4	gpperfmon	pg_catalog	<a href="#">pg_attribute</a>	7.88 MB	174.13%	--	2022-04-09 01:30:47
5	gpperfmon	pg_catalog	<a href="#">pg_proc</a>	7.00 MB	174.13%	--	2022-04-09 01:30:47

The **Age** table has these columns:

Rank

Tables in the list are ranked by Age, from oldest to youngest

Database

Name of the database containing the table

Schema

Name of the schema to which the table belongs

Relation Name

Name of the table

Owner

The table owner

Row Count

Number of rows in the table

Size

Size of the table

## Age

Age of the table, expressed as a percent of (2 billion - `xid_stop_limit` - `xid_warn_limit`). For example, if the table's age is reported as 50%, it means the table is about halfway to reaching the point where data loss could occur. When a table's age reaches (2 billion - `xid_stop_limit` - `xid_warn_limit`), it means the transaction ID has reached the value specified in `xid_warn_limit`. When this happens, users will see the warning message: "WARNING: database "<database\_name>" must be vacuumed within <number\_of\_transactions> transactions".

To read more about the role of the `xid_stop_limit` and `xid_warn_limit` GUCs in preventing transaction ID wraparound, see the [Transaction ID Wraparound](#) topic.

## Last VACUUM FREEZE

Date of the last VACUUM FREEZE performed on the table

## Last Accessed

The date the table was last accessed

# Monitoring and Managing Greenplum Database Queries

- [Query Monitor](#)
- [Query Details](#)
- [Query History](#)

## Query Monitor

The **Query Monitor** view allows you to view information for all Greenplum Database server queries, including details about queries running, queued to run, and blocked by other queries.

In addition, the **Query Monitor** displays a variety of information about sessions, such as session status, associated users and databases, idle time, and associated queries.

Users with Admin or Operator privileges can see and cancel all users' queries, or can move queries. They can also see and cancel all users' sessions, and export session details to a CSV file.

Dashboard

Query Monitor

Host Metrics

Cluster Metrics

History

Table Browser

Recommendations

Workload

System

Admin

Current queries by all users

Current Time  
2021-05-24 03:10:23

Help

Queries

Sessions

2 Running

0 Queued

1 Blocked

Advanced Search (0)

Pause Query Monitor

CANCEL QUERY

EXPORT

<input type="checkbox"/>	Query ID	Status	User	Database	Workload	Submitted	Queued Time	Run Time	Spill Files	CPU Time
<input type="checkbox"/>	1621409718-26753-3	Blocked	gpmon	gpmon	admin_group	20:28:37	3d 6h 41m 53s	--	--	--
<input type="checkbox"/>	1621409718-37684-2	Running	gpadmin	gppperfmon	default_group	08:16:50	1d 23h 59m 13s	18h 54m 27s	--	2s
<input type="checkbox"/>	1621409718-37687-2	Running	gpadmin	gppperfmon	default_group	08:16:55	1d 23h 59m 15s	18h 54m 20s	--	2s

- View and isolate queries with high CPU consumption.



#### Note

The Query Monitor does not display queries executed by the `gpmon` user in the `gpperfmon` database.

## Query Metrics

To display query metrics, click the Queries tab. The Query Monitor table displays the following columns for queries.

### Query ID

An identification string for the query. If the column is blank, no query ID has been assigned yet. In the Console, this looks like "1295397846-56415-2". Command Center generates this ID by combining the query record's `tmid`, `ssid`, and `ccnt` fields.

- `tmid` is a time identifier for the query.
- `ssid` is the session id.
- `ccnt` is the number of the command within the session.

### Status

The status of the query. This can be one of the following:

- Queued: the query has not yet started to execute
- Running: execution has started, but is not yet complete
- Blocked: the query is waiting for one or more other queries to release locks
- Done: completed successfully
- Cancelling: cancel request sent, cancel pending
- Cancelled: terminated, no longer running
- Idle Transaction: the transaction is open, but idle, for example, waiting while a user in an interactive session enters a statement

### User

The Greenplum Database role that submitted the query.

### Database

The name of the database that was queried.

### Workload

The resource group or resource queue that is managing the query. With Greenplum 6.8 or later, you can use the drop-down menu to reassign the query to a different resource group.

### Submitted

The time the query was submitted to the query planner.

### Queued Time

The amount of time the query has been (or was) in queue awaiting execution.

#### Run Time

The amount of time since query execution began.

#### Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

#### CPU Time

Shows the amount of system CPU consumed by the individual query.

## Session Information

To display session information, click the Sessions tab. The Query Monitor table displays the following columns for sessions.

#### Session ID

An identification string for the session.

#### User

The Greenplum Database role that established the session.

#### Database

The name of the database connected to in the session.

#### Status

The status of the session. This can be one of the following:

- active: the connection is working
- idle: the connection is idle
- idle in transaction: a transaction is running but not active, potentially waiting for user input
- idle in transaction (aborted): similar to idle in transaction except that one of the statements in the transaction caused an error

#### Start Time

The time the session was established.

#### Application Name

The name of the application that established the connection, for example, psql.

#### Idle Time

The amount of time since activity was last detected in the session.



#### Note

Command Center calculates the idle time from the last query's completion time. However, if this information is not available, idle time is calculated from the time that the idle status is detected. In such situations, the displayed idle time may be shorter

than the actual idle time.

**Last Query ID** The ID of the last query that was running before the session became idle. If the session is active, this displays the ID of the running query. To view the query text, hover your mouse over the ID.

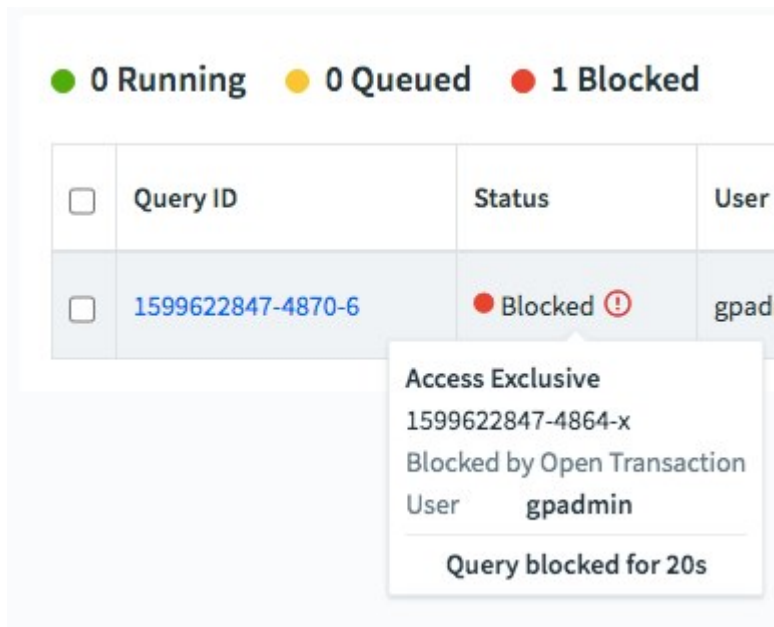
## Using the Query Monitor Controls for Queries


You can use the Query Monitor to display basic information about queries, cancel or export queries, see a query's details, and reassign the query to a different resource group.

In addition, you can pause the query monitor to see a snapshot of all running queries, filter queries by their status, and perform an advanced search of queries.

### General Tasks

- If a query shows the **Blocked** status, use the tooltip to display the transaction that is blocking the query.



<input type="checkbox"/>	Query ID	Status	User
<input type="checkbox"/>	1599622847-4870-6	Blocked 	gpadmin

**Access Exclusive**  
 1599622847-4864-x  
 Blocked by Open Transaction  
 User **gpadmin**  
 Query blocked for 20s

- Click a column heading to sort the rows on that column in ascending or descending order.
- Click the checkbox at the left of a row to choose a query to cancel or export. Click the checkbox in the heading row to choose all queries.
- Click **CANCEL QUERY** to cancel selected queries. A pop-up box prompts you to enter a reason. Enter the error message (max 128 characters) displayed to users whose sessions are cancelled.
- Click **EXPORT** to download a comma-separated values (CSV) text file containing rows for the selected queries. When no queries are selected, all rows are exported. The default file name is `spreadsheet.csv`.
- Click any query ID to see the [Query Details](#), including metrics, the text of the query, and the query plan.
- With Greenplum 6.8 or later, use the drop-down menu in the [Workload] column to reassign

a query to a different resource group.

## Pausing and Resuming the Query Monitor

You can pause the Query Monitor to view a snapshot of running queries, and then resume the Query Monitor. This allows you to see the status of a query before it disappears from the query list.

- To pause the Query Monitor, click the Pause Query Monitor button. When you pause the Query Monitor, Command Center displays a warning message that the queries you are viewing are not up to date.
- When you pause the Query Monitor, the query list stops refreshing. Pausing the Query Monitor merely suspends the UI data stream, it does not pause query execution. When you resume monitoring, you will see the updated state, which means some queries may disappear immediately if they are already done.
- You can click a Query ID to display the Query details page. The information on this page continues to be updated in real time.
- When you are viewing the details of a query that completed after pausing the Query Monitor, you will see two possible pages:
  - ◊ For queries already saved to Query History, the displayed information matches that of the Query History page.
  - ◊ If the query is not already saved to query history, you will see a temporary page containing some outdated information, as well as a warning message requesting that you reload the page.
- While the Query Monitor is paused, if you click on a Query ID to view that query's details, when you return to the Query Monitor some data will have been updated, including Status, Spill File Size, and CPU Time. If a query has completed, it will no longer appear on the list.
- To resume the Query Monitor, click the Resume Query Monitor button.

## Filtering Queries

You can filter queries to show any combination of running, queued, and blocked queries. In addition, using the Advanced Search button, you can filter queries by a variety of other attributes.

- To filter by Status, select or deselect the **Running** (green), **Queued** (yellow) or **Blocked** (red) buttons. When the colored dot is solid, the Query Monitor displays queries with that status. When empty, the Query Monitor hides queries with that status.

For example, if the **Running** and **Queued** dots are solid but the **Blocked** dot is empty, the Query Monitor displays running and queued queries but not blocked queries.

- To filter query results using Advanced Search, click the Advanced Search button and then select the criteria for the search.



**Query ID**

Returns the query or queries identified by the string or strings entered. Separate multiple query ids with commas.

**Users**

Returns queries owned by the specified users.

**Query Tags**

Returns queries whose query tags match the values entered. Enter the tags as key-value pairs. Separate multiple query tags with semicolons. Not available in Command Center 4.x.

**Database**

Returns queries that are running against the specified database.

**Departments**

Enter a department, or list of departments, from the dropdown. Filtering by department shows the queries owned by users from the specified department(s).

**Res Group**

Returns queries executed by the specified resource queue or resource group.

**Submit Time (>=)**

Returns queries that were submitted in the window of time specified.

**Queued Time (>=)**

Returns queries that have been in queue awaiting execution for at least the specified amount of time. You may specify in seconds, minutes, or hours.

**Run Time (>=)**

Returns queries that have been running for at least the specified amount of time. You may specify in seconds, minutes, or hours.

#### Spill File Size (>=)

Returns queries that generated spill files of the specified size. You may specify in MB, GB, or TB.

#### CPU Time (>=)

Returns queries that have consumed the specified amount of CPU time. You may specify in seconds, minutes, or hours.

Click **APPLY** to display results that match your criteria. The number of conditions will be retained until you click **RESET**.

## Using the Query Monitor Controls for Sessions

- Click a column heading to sort the rows on that column in ascending or descending order.
- Click the checkbox to the left of a row to choose a session to cancel or export. Click the checkbox in the heading row to choose all sessions.
- Click the checkbox just above the Session ID column labeled "Show idle in transaction Only" to display only the sessions whose status is idle in transaction.
- Click **CANCEL SESSION** to cancel selected sessions. A pop-up box prompts you to enter a reason. You can enter a message of up to 128 characters to display with the error message that is received by users whose sessions are cancelled.

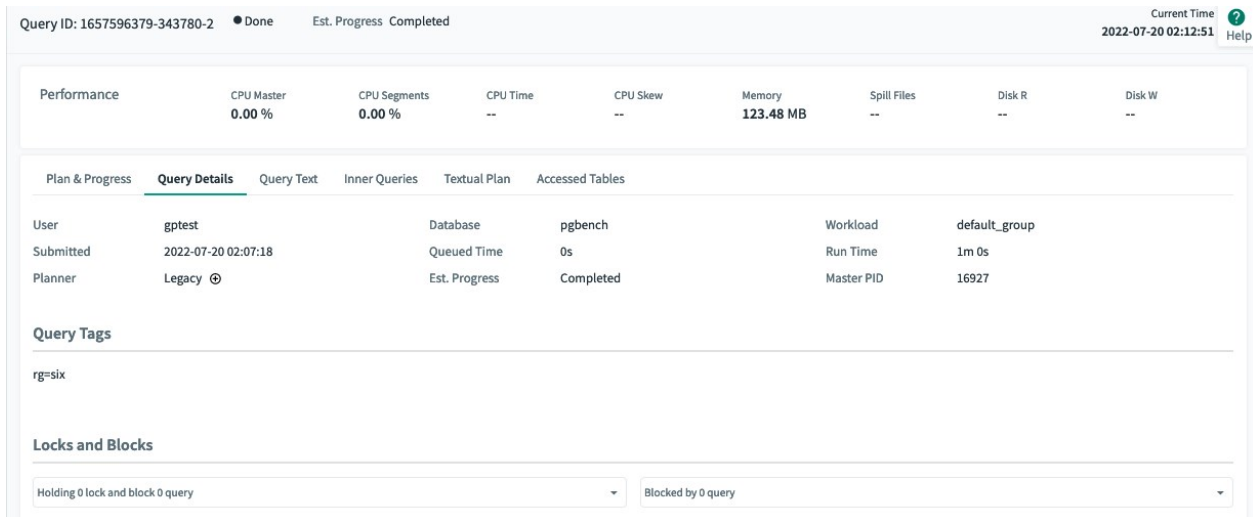


**Note:** You cannot cancel sessions whose Status is active.

- Click **EXPORT** to download a comma-separated values (CSV) text file containing rows for the selected sessions. When no sessions are selected, all rows are exported. The default file name is `spreadsheet.csv`.

## Query Details

The **Query Details** view displays query metrics, the query text, query tags, information about inner queries, the query plan and progress, and the tables the query accessed for a single query selected from the [Query Monitor](#) view.



## Query Execution Metrics

The Query ID, execution status, and run time of the query are displayed at the top.

The following metrics are displayed for the query.

User

The Greenplum Database role that submitted the query.

Database

The name of the database that was queried.

Workload

The name of the resource group or resource queue that is managing the query.

Planner

The name of the query planner used for this query, GPORCA or Legacy.

Submitted

The time the query was submitted to the query planner.

Queued Time

The amount of time the query has been (or was) in queue awaiting execution.

Master PID

The query's process ID on the master host.

Run Time

The amount of time since query execution began.

### Est. Progress

An estimate of the percentage of the query execution completed. The estimate is calculated from row count and cost estimates generated by either the GPORCA or legacy planner for the particular query and the available statistics. The estimate does not account for the many other factors that can affect query execution so it should not be seen as a reliable predictor of query completion time.

The progress for each plan node is calculated as the ratio of actual rows produced to the planner's estimate of the total number of rows the node will produce:

$$NodeProgress = \frac{ActualRows}{EstimatedRows}$$

The overall progress for the query is estimated using the calculated node progress and the planner's cost estimates:

$$OverallProgress = \frac{\sum NodeProgress(x) * Est.Cost(x)}{\sum Est.Cost(x)}$$

If the estimate is greater than 100% and the query has not yet completed, 99.9% completion is reported. 100% is reported if the formula produces an estimated percentage greater than 100%.

CPU Master Current CPU percent on the Greenplum Database master host for this query.

### CPU Segments

(Active queries only.) Current CPU percent average for all segment processes executing this query. The percentages for all processes running on each segment are averaged, and then the average of all those values is calculated to render this metric. Current CPU percent average is always zero in historical and tail data. The master and standby master are excluded from the calculation.

### CPU Time

Total CPU time consumed by all processes on all segments executing this query.

### CPU Skew

The amount of CPU skew. CPU skew occurs when query executor processes for one segment use a disproportionate amount of CPU compared to processes for other segments executing the query. This value is calculated as

$$1 - (\text{average\_segment\_CPU} / \text{maximum\_segment\_CPU})$$

### Memory

Memory consumed by all segment processes executing the query.

### Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

### Disk R

The current average disk read rate for all segment hosts.

### Disk W

The current average disk write rate for all segment hosts.

### Locks and Blocks

Contains two lists of locks currently blocking transactions. Click a list to expand and view the contents.

- A list of locks held by this query, including the type of each lock and the queries blocked by that lock.

#### Locks and Blocks

Holding <b>1</b> lock and block <b>3</b> query	
Access Exclusive	<a href="#">1531266727-15345-2</a>
--	<a href="#">1531266727-15341-2</a>
	<a href="#">1531266727-15342-2</a>
Blocked by <b>12</b> query	

- A list of queries that hold locks that block this query and the lock type.

#### Locks and Blocks

Holding <b>1</b> lock and block <b>3</b> query	
Blocked by <b>12</b> query	
<a href="#">1531266727-15295-2</a>	Access Share
<a href="#">1531266727-15157-2</a>	Access Share
<a href="#">1531266727-15160-2</a>	Access Share

## Query Text, Query Tags, Inner Queries, and Execution Plan

The query text, the query's tags, statements within the function(s) the query calls -- "inner queries", and the query's plan and execution progress are shown in the lower panels of the Query Details view.

### Query Text

The **Query Text** panel displays the text of the query as it was submitted to Greenplum Database.

Command Center can display up to 100K characters. If you click **COPY**, up to 100K characters of the query text are copied to the clipboard.

If the query text is longer than 100K characters, a message is displayed with a link you can use to download the full text of the query. The name of the text file is the ID of the query with a `.txt` extension. The file is available to download for 24 hours after the query completes, or until the query has been saved to history, once history collection is enabled.

### Query Tags

The **Query Tags** panel displays the tags associated with the query.

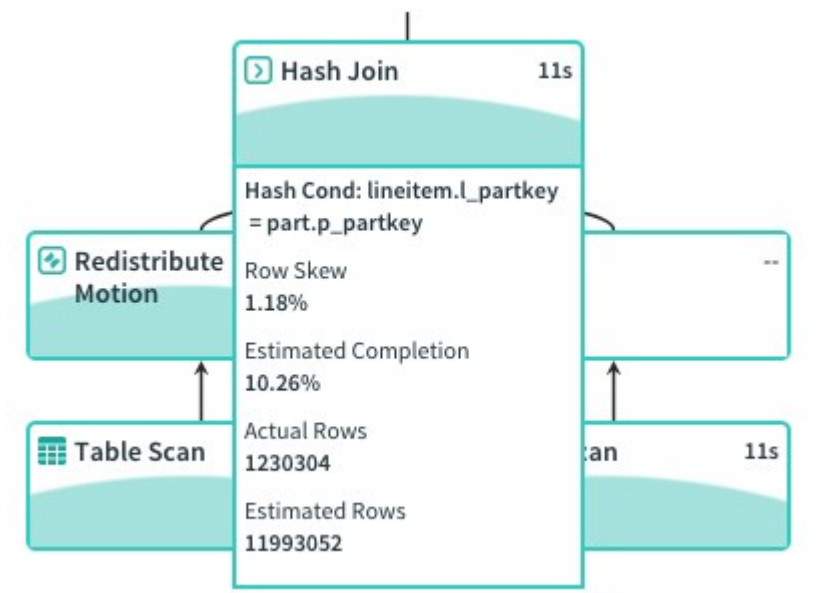
## Query Plan and Progress

The **Plan & Progress** tab in the lower right panel is a graphical representation of the query plan with animation and real-time metrics to show execution progress. Each box in the tree represents a step in the query execution plan. The boxes are labeled with the operation they represent and have a CPU usage metric. Query execution begins at the bottom of the tree and ends at the top.

Before a step begins to execute, the box has a solid white fill. When the step becomes active, the box is animated with a green and white fill to indicate that the operator is working. When the step has completed, the box has a solid green fill.

Query execution plans are executed in "slices," portions of the query plan that segments can work on independently in parallel. The plan is sliced wherever a data motion operator occurs. The time displayed in the upper right corner of each box is the amount of CPU time used for all steps in the slice to which the step belongs. Each slice is displayed in its own color, making it easy to distinguish the slices from each other. The visual query plan does not illustrate slices, but you can find information about slices in the textual plan.

If you click a step, the box expands to show additional details.



The expanded plan box contains the following metrics.

- The type of operator. When the operator is a table scan, the name of the scanned table is included. See [Query Plan Execution](#) for descriptions of the operators.
- Information related to the current operation, such as the hash key, merge key, join condition, or filter condition.
- Row Skew - the amount of row skew for the current operator, a value from 0% to 100%. Row skew occurs when some segments process more rows than other segments. The percentage is calculated as  $(1 - (\text{average\_segment\_rows} / \text{maximum\_segment\_rows})) * 100$ . Nodes with a row skew of greater than 30% are marked with an exclamation point.
- Estimated Completion - the current percentage of actual rows to estimated rows for this plan step. The percentage can exceed 100% if the operation produces more rows than the optimizer's estimate. The percentage changes to "Completed" when the operation

completes.

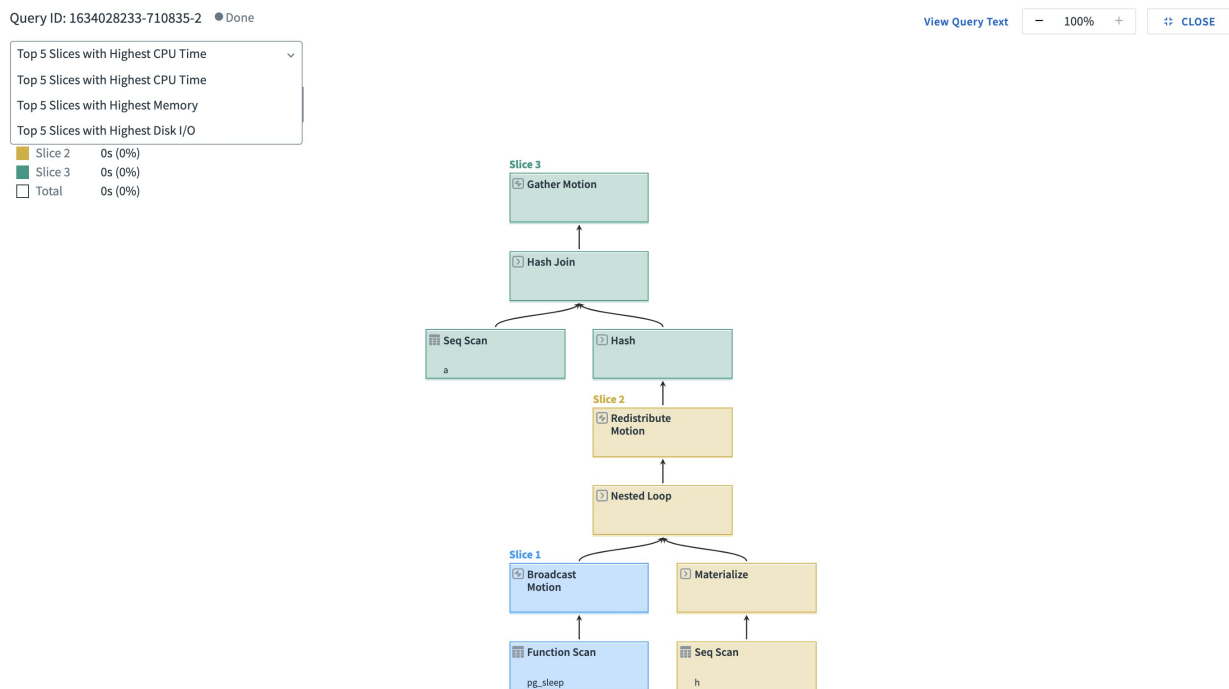
- **Actual Rows** - The current number of rows produced by this step. Note that for nested join operators, the Actual Rows is estimated since the actual row counts are not available while the join is executing.
- **Estimated Rows** - The estimated number of rows the operator will produce.

## Display Slice Metrics

From within the **Plan and Progress** tab you may display key metrics about slices, including:

- the top 5 slices that consume the most CPU time
- the top 5 slices that consume the most memory
- the top 5 slices that use the most disk I/O

To display this information, click the **Expand** button on the far right. This displays a screen with a dropdown menu from which you can choose which slice metrics you want to display:



## Inner Queries

The **Inner Queries** panel is disabled by default, which means the panel is blank. When enabled, it shows statements being called by the function(s) the query calls. Command Center displays a series of non-continuous snapshots of statements being executed, rather than a continuous execution flow. To enable this panel, set the `gpcc.enable_send_inner_query` server configuration parameter to `true`.

## Textual Plan

Select the **Textual Plan** tab and click **RUN EXPLAIN** to generate the text representation of the explain plan.



**Note:** The **RUN EXPLAIN** button is dimmed if Command Center is unable to

generate the explain plan. Command Center is unable to generate the explain plan if the size of the query text is greater than 100K characters or if the query text contains multiple statements.

Query ID: 1527709348-39-3 ●Running Run Time 4m 9s CANCEL QUERY Current Time 2018-05-30 14:47:53 help

Details	Current Performance		Blocking (0)	Blocked by (0)
User: <b>tpch_4</b>	CPU: <b>11.08 %</b>	CPU Skew: <b>41.67 %</b>		
Database: <b>gpadmin</b>	Memory: <b>511.64 MB</b>	Spill Files: <b>608.69 MB</b>		
Res Queue: <b>pg_default</b>	Disk R: <b>0.00 MB/s</b>	Disk W: <b>2.06 MB/s</b>		
Submitted: <b>14:43:51</b>				
Queued Time: <b>0s</b>				
Run Time: <b>4m 9s</b>				

### Query Text

```

select
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice,
  sum(l_quantity)
from
  customer,
  orders,
  lineitem
where
  o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 5
  )
  and c_custkey = o_custkey
  and o_orderkey = l_orderkey
group by
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice
order by
  o_totalprice desc,
  o_orderdate
;

```

### Plan & Progress BETA Textual Plan

```

Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..6554.33 rows=2402130 width=49)
Merge Key: orders.o_totalprice, orders.o_orderdate
-> GroupAggregate (cost=0.00..6160.61 rows=600533 width=49)
  Group By: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
  -> Sort (cost=0.00..6115.27 rows=600533 width=48)
    Sort Key: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
    -> Hash Join (cost=0.00..2977.88 rows=600533 width=48)
      Hash Cond: public.lineitem.l_orderkey = orders.o_orderkey
      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
      -> Hash (cost=1888.16..1888.16 rows=124319 width=41)
        -> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..1888.16 rows=124319 width=41)
          Hash Key: orders.o_orderkey
          -> Hash Join (cost=0.00..1872.21 rows=124319 width=41)
            Hash Cond: orders.o_custkey = customer.c_custkey
            -> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..1371.15 rows=124319 width=22)
              Hash Key: orders.o_custkey
              -> Hash EXISTS Join (cost=0.00..1362.59 rows=124319 width=22)
                Hash Cond: orders.o_orderkey = public.lineitem.l_orderkey
                -> Table Scan on orders (cost=0.00..459.64 rows=374619 width=22)
                -> Hash (cost=784.20..784.20 rows=124319 width=4)
                  -> Result (cost=0.00..784.20 rows=124319 width=4)
                    Filter: (sum(public.lineitem.l_quantity)) > 5::numeric
                    -> HashAggregate (cost=0.00..773.98 rows=310798 width=12)
                      Group By: public.lineitem.l_orderkey
                      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
                  -> Hash (cost=434.95..434.95 rows=37596 width=23)
                    -> Table Scan on customer (cost=0.00..434.95 rows=37596 width=23)

```

Settings: optimizer=on  
Optimizer status: PQO version 2.58.0

The textual plan is the output of the Greenplum Database **EXPLAIN** command for the query. The query plan steps are labeled with arrows (->) and the structure of the query plan tree is indicated with indentation.

The **Optimizer status:** line at the bottom of the textual plan reports whether the explain plan was generated using the GPORCA optimizer (PQO) or the legacy query optimizer.

For help reading the textual explain plan see the **EXPLAIN** command in the *Greenplum Database Reference Guide* and **Query Profiling** in the *Greenplum Database Administrator Guide*. See **Query Execution** for descriptions of the query operators.

## Accessed Tables

The **Accessed Tables** panel displays the tables this query accessed.

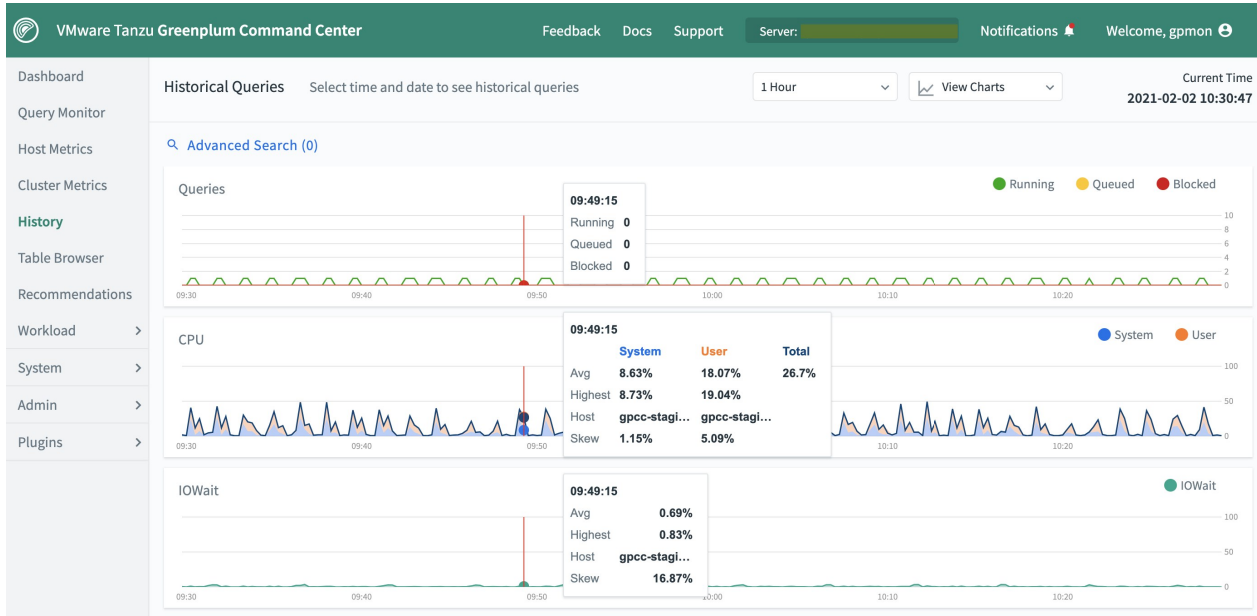
## History



The **History** page allows you to display system metrics charts and details about queries executed during a specified time period.

Use the [Advanced Search](#) tool to refine the query search results. The [Queries Table](#) shows the results and details of the search.

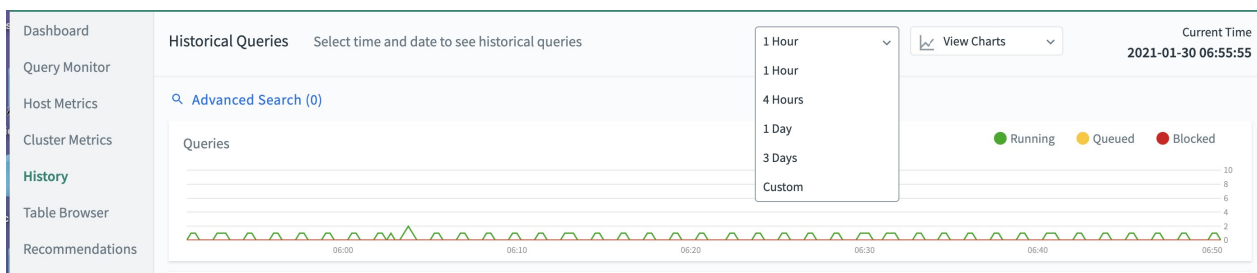
By default, the dashboard shows the last 1 hour of query history.



**Note:** On the [Admin> Settings](#) page, you can enable saving the real-time metrics collected by the Greenplum Database metrics collector extension to history in the gpmetrics schema of the gpperfmon database. When you enable collecting this history, the Command Center History, Query Monitor, and Query Detail views all use data derived from the same data collection method.

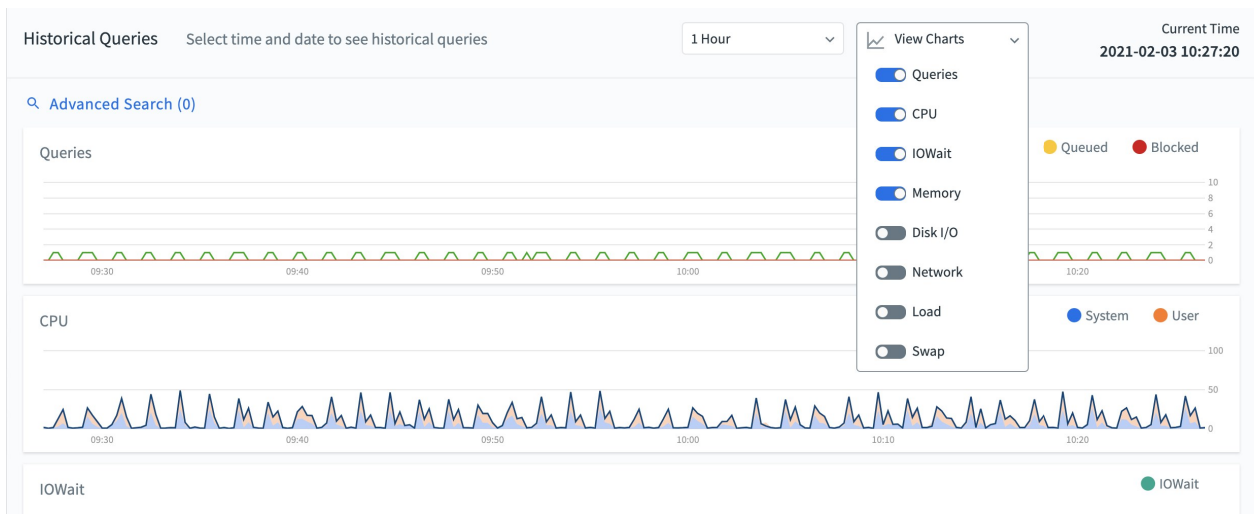
If you disable GPCC query history collection on the **Admin> Settings** page, the history view displays data collected and saved in the gpperfmon database by the `gpmmmon` and `gpsmon` agents. That data is not derived from the real-time metrics displayed in the Query Monitor and Query Detail view. Variations should be expected due to the different data collection methods.

To select a different time period (1 hour, 4 hours, 1 day, 3 days, and custom), use the time range dropdown:



Selecting "custom" from the time period dropdown leads you to the **Advanced Search** popup window. See [Advanced Search](#) for more details.

To select which charts to view for the latest queries, use the **View Charts** dropdown. By default, the following charts are selected:

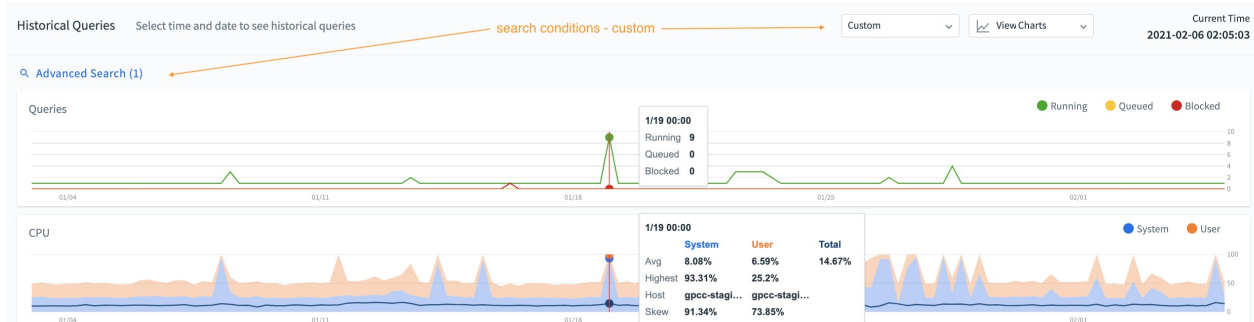


The following metrics charts are available:

**Queries** The number of queries running and the number of queries queued to run. **CPU** The percentage of CPU used by system processes and user processes. **IOWait** The time waiting for disk I/O. **Memory** The percentage of memory in use. **Disk I/O** Disk read and write rates in megabytes per second. **Network** Network I/O read and write rates in megabytes per second. Network metrics include traffic over all NICs (network interface cards), including internal interconnect and administrative traffic. **Load** System load average for 1-minute, 5-minute, and 15-minute periods. **Swap** The percentage of swap space used.

## Advanced Search

Use the **Advanced Search** tool to restrict the query results displayed at the bottom of the history dashboard.



The number of search conditions is displayed beside the **Advanced Search** link. If the search conditions are greater than (0), the Time Range dropdown defaults to **Custom**. The search conditions number is retained between different Command Center dashboards. To reset it, select an option from the Time Range dropdown, or click the **RESET** button in the **Advanced Search** popup window. The search conditions go back to (0) if the user logs out.

**Advanced Search**

Time Between\*   To

Query ID

Query Text

Query Tags

Database

Users

Departments

Resource Groups

CPU Time(≥)   Disk I/O(≥)

Peak Memory(≥)   Run Time(≥)

Spill File Size(≥)   Planner Cost(≥)

Status  Has CPU Skew ☐ Yes

RESET **APPLY**

**Note:** In the **Advanced Search** popup window, you must set the **Time Between** start field in order for the **APPLY** button to be highlighted.

### Time Between

(Required) Select a valid time range in the HH:MM, HHMM, H:MM, HMM format. If you enter start time but no end time, the results will be all the queries that are running at that specific time. If you select a date but do not specify a time, the default is 00:00.

**Query ID** Enter the query IDs for your query search, separated by commas.

### Query Text

Enter the keywords for your query search, separated by spaces, and within a limit of 256 characters. You may enter a % wildcard to match any characters, for example `SELECT%ABC%`.

### Query Tags

Enter the query tags for your query search, separated by semicolons.

### Database

Select a database from the dropdown. Filtering by database affects only the queries displayed in query results. The metrics displayed in charts include all activity during the selected time period.

### Users

Enter a user, or a list of users, from the dropdown. Filtering by users shows the queries owned by those users. The metrics displayed in charts include all activity during the selected time period.

### Departments

Enter a department, or list of departments, from the dropdown. Filtering by department shows the

queries owned by users from the specified department(s).

### Res Group

Display queries executed by a specified resource queue or resource group.

### CPU Time (>=)

Select an integer for the seconds, minutes, or hours of CPU time the queries consume.

### Disk I/O (>=)

Select queries with disk I/O above the specified Megabytes (MB), Gigabytes (GB), or Terabytes (TB).

### Peak Memory (>=)

Select queries that consume the specified maximum memory usage number, in MB, GB, or TB.

### Run Time (>=)

Select queries that consumed a specific CPU time in seconds, minutes, or hours.

### Spill File Size (>=)

Display queries that generated spill files of a certain size.

### Planner Cost (>=)

Select a planner, and a cost.

### Status

Display queries that completed with the specified status: **Done**, **Cancelled**, or **Error**.

### Has CPU Skew

Display queries that had CPU skew greater than zero.

Click **APPLY** to display results that match your criteria. The number of conditions will be retained until your click **RESET** or select a time period in the History dashboard.

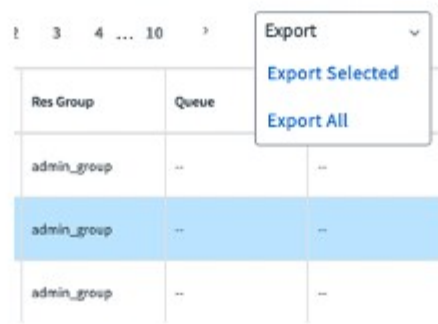
## Queries Table

The Queries results table displays queries that were active during the specified Time Range or the **Advanced Search** conditions. The results can include queries that started before or finished after the specified time. However, queries that are still active are not included in the table; these queries can be viewed on the [Query Monitor](#) page.

24 Queries found < 1 > Export

<input type="checkbox"/>	Query ID	Status	User	Database	Submitted	Queued Time	Run Time	Ended ~	CPU Skew	Spill Files	Res Group	Queue	Priority
<input type="checkbox"/>	<a href="#">1604026601-2386291-13</a>	Done	gpmon	euc_cn_test	2021-02-02 15:50:01	0s	35s	2021-02-02 15:50:36	6.43	0.00 MB	admin_group	--	--
<input type="checkbox"/>	<a href="#">1604026601-2364362-15</a>	Done	gpadmin	gppperfmon	2021-02-01 15:27:59	0s	1m 47s	2021-02-01 15:29:46	3.50	0.00 MB	admin_group	--	--
<input type="checkbox"/>	<a href="#">1604026601-2362660-13</a>	Done	gpmon	euc_cn_test	2021-02-01 13:51:32	0s	36s	2021-02-01 13:52:08	6.78	0.00 MB	admin_group	--	--
<input type="checkbox"/>	<a href="#">1604026601-2279093-13</a>	Done	gpmon	euc_cn_test	2021-01-29 13:08:55	0s	37s	2021-01-29 13:09:32	6.69	0.00 MB	admin_group	--	--
<input type="checkbox"/>	<a href="#">1604026601-2276606-13</a>	Done	gpmon	euc_cn_test	2021-01-29 11:08:16	0s	46s	2021-01-29 11:09:02	9.45	0.00 MB	admin_group	--	--

To export any selected query information, or the whole table, use the Export dropdown at the top right of the table:



The Queries results table has the following columns:

#### Query ID

An identification string for the query. In the Console, this looks like "1295397846-56415-2".

#### Status

The final status of the query: Done, Cancelled, or Error.

#### User

The Greenplum Database user who submitted the query.

#### Database

The name of the database that was queried.

#### Submitted

The time the query was submitted to the query planner.

#### Queued Time

The time the query waited before it was executed. In addition to time in the queue, this includes other time such as time in the optimizer.

#### Run Time

The amount of time the query required to produce a result.

#### Ended

The time the query completed or was cancelled.

#### CPU Skew

The amount of CPU skew. CPU skew occurs when query executor processes for one segment use a disproportionate amount of CPU compared to processes for other segments executing the query. This value is the coefficient of variation for the CPU used by processes running this query on each segment, multiplied by 100. For example, a value of .95 is shown as 95.

#### Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

#### Res Group or Res Queue

The name of the resource group or resource queue for the query. For more information about Resource Queues, Resource Groups, and Query Plans, refer to the [Greenplum Database Administrator Guide](#).

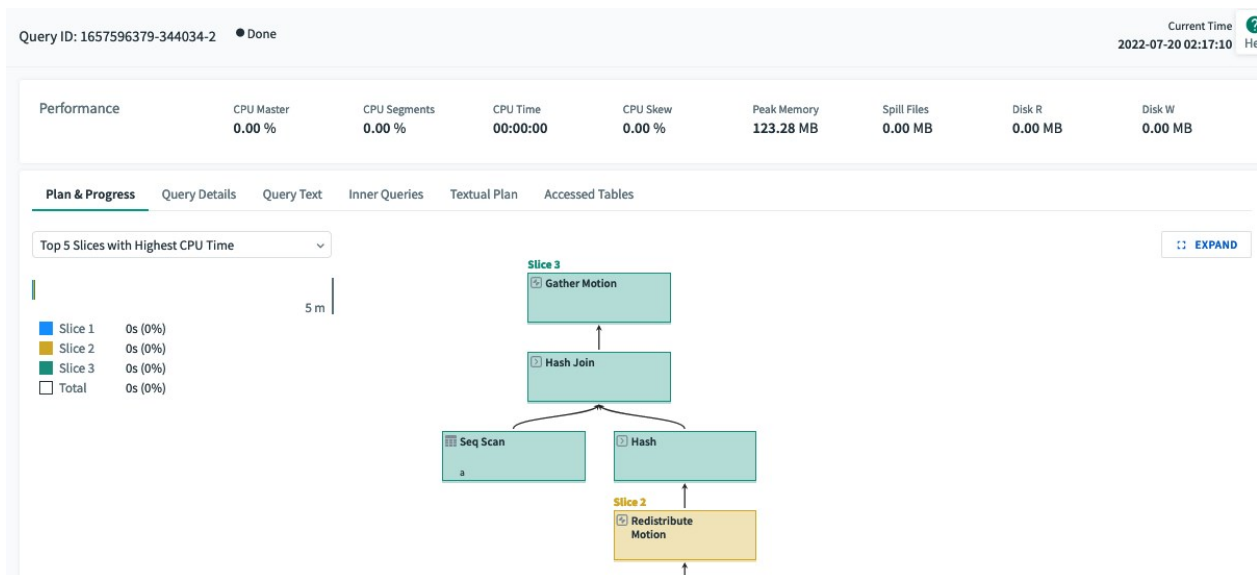
### Priority

(Res Queue only) A query executes with the CPU priority assigned to its resource queue.

Select one of the query links to move to the [Query History Details](#) dashboard, which includes details about the query metrics, the query text, and the query plan and execution statistics for that query.

## Query History Details

The **Query History Details** view displays query metrics, the query text, query tags, information about inner queries, the query plan and progress, and the tables the query accessed for a single query selected from the [Query History](#) view.



## Query History Metrics

The Query ID, execution status, and run time of the query are displayed at the top.

The following metrics are displayed for the query. Under the **Details** heading:

### User

The Greenplum Database role that submitted the query.

### Database

The name of the database that was queried.

### Res Group / Res Queue

The name of the resource group or resource queue that is managing the query.

### Planner

The name of the query planner used for this query, GPORCA or Legacy. Press the "+" to see the cost of the query, and copy using the mouse.

Res Group	admin_group	Run Time	0s
Planner	Legacy 		
	1.2		

**Submitted**

The time the query was submitted to the query planner. Hover over the submitted time to view the submitted date.

**Details**

User	gpmon	Submitted	22:54:12
Database	euc_cn_test	Queued T	2021-02-05 22:54:12
Res Group	admin_group	Run Time	10s
Planner	Legacy 		

**Queued Time**

The time the query waited before it was executed. In addition to time in the queue, this includes other time such as time in the optimizer.

**Run Time**

The amount of time the query executed.

Under the **Performance** heading, the metrics indicate the final state of the query before it ends:

**CPU Master**

The CPU percent on the Greenplum Database master host for this query.

**CPU Segments**

CPU percent average for all segment processes executing this query. The percentages for all processes running on each segment are averaged, and then the average of all those values is calculated to render this metric. The master and standby master are excluded from the calculation.

**CPU Time**

Total CPU time consumed by all processes on all segments executing this query.

**CPU Skew**

The amount of CPU skew. CPU skew occurs when query executor processes for one segment use a disproportionate amount of CPU compared to processes for other segments executing the query.

This value is calculated as  $1 - (\text{average\_segment\_CPU} / \text{maximum\_segment\_CPU})$

**Peak Memory**

The maximum amount of memory consumed by all segment processes executing the query. For queries that ran before version 6.7/4.15 of Command Center, this metric will display "--", as peak memory was not collected at that time.

### Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

### Disk R

The total disk read rate for all segment hosts.

### Disk W

The total disk write rate for all segment hosts.

### Locks and Blocks

Contains lists of queries blocked by locks this query holds, and queries held by other transactions that block this transaction. Click a list to expand and view the contents.

## Query Text and Execution Plan

The query text and the query's plan and execution progress are shown in the lower panels of the Query Details view. The text of the query is displayed in the left panel, and the plan and progress is displayed in the right panel. The plan is available only for queries that ran for at least ten seconds.

### Query Text

The **Query Text** panel displays the text of the query as it was submitted to Greenplum Database.

Command Center can display up to 100K characters. If you click **COPY**, up to 100K characters of the query text are copied to the clipboard.

If the query text is longer than 100K characters, a message is displayed with a link you can use to download the full text of the query. The name of the text file is the ID of the query with a `.txt` extension. The file is available to download for 24 hours.

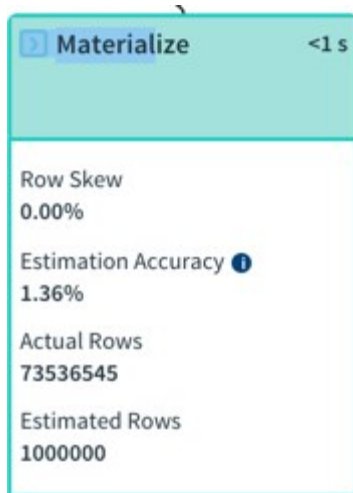
### Query Plan and Progress

The **Plan & Progress** tab in the lower right panel is a graphical representation of the query plan with metrics to show the planned and actual query execution. Each box in the tree represents a step in the query execution plan. The boxes are labeled with the operation they represent. Query execution begins at the bottom of the tree and ends at the top.

Query execution plans are executed in "slices," portions of the query plan that segments can work on independently in parallel. The plan is sliced wherever a data motion operator occurs. The time displayed in the upper right corner of each box is the amount of CPU time used for all steps in the slice to which the step belongs. Each slice is displayed in its own color, making it easy to distinguish the slices from each other. The visual query plan does not illustrate slices, but you can find information about slices in the textual plan.

If you click a step, the box expands to show additional details.





The expanded plan box contains the following metrics.

- The type of operator. When the operator is a table scan, the name of the scanned table is included. See [Query Plan Execution](#) for descriptions of the operators.
- Information related to the current operation, such as the hash key, merge key, join condition, or filter condition.
- Row Skew - the amount of row skew for the current operator, a value from 0% to 100%. Row skew occurs when some segments process more rows than other segments. The percentage is calculated as  $(1 - (\text{average\_segment\_rows} / \text{maximum\_segment\_rows})) * 100$ . Nodes with a row skew of greater than 30% are marked with an exclamation point.
- Estimation Accuracy - a percentage calculated from the estimated rows the node would produce and the actual rows the node produced when the query executed. The percentage is calculated as  $\min(\text{estimated\_rows}, \text{actual\_rows}) / \max(\text{estimated\_rows}, \text{actual\_rows}) * 100$
- Actual Rows - The number of rows produced by this step.
- Estimated Rows - The estimated number of rows the operator will produce.

## Display Slice Metrics

From within the **Plan and Progress** tab you may display key metrics about slices, including:

- the top 5 slices that consume the most CPU time
- the top 5 slices that consume the most memory
- the top 5 slices that use the most disk I/O

To display this information, click the **Expand** button on the far right. This displays a screen with a dropdown menu from which you can choose which slice metrics you want to display:

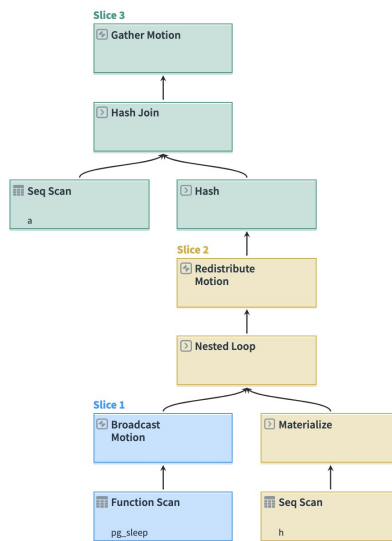
Query ID: 1634028233-710835-2 ● Done

[View Query Text](#)

- 100% +

[CLOSE](#)

Top 5 Slices with Highest CPU Time	▼
Top 5 Slices with Highest CPU Time	
Top 5 Slices with Highest Memory	
Top 5 Slices with Highest Disk I/O	
<div> <div></div> <div>Slice 2</div> <div>0s (0%)</div> </div> <div> <div></div> <div>Slice 3</div> <div>0s (0%)</div> </div> <div> <div></div> <div>Total</div> <div>0s (0%)</div> </div>	



## Textual Plan

Select the **Textual Plan** tab and click **RUN EXPLAIN** to generate the text representation of the explain plan. The **EXPLAIN** command is run against the current state of the database, so the plan generated may differ from the the plan used when the query executed.



**Note:** The **RUN EXPLAIN** button is dimmed if Command Center is unable to generate the explain plan. Command Center is unable to generate the explain plan if the size of the query text is greater than 100K characters or if the query text contains multiple statements.

Query ID: 1527709348-39-3
Running
Run Time 4m 9s
CANCEL QUERY
Current Time 2018-05-30 14:47:53
help

Details	Current Performance		Blocking (0)	Blocked by (0)
User	tpch_4	CPU	CPU Skew	
Database	gpadmin	11.08 %	41.67 %	
Res Queue	pg_default	Memory	Spill Files	
Submitted	14:43:51	511.64 MB	608.69 MB	
Queued Time	0s	Disk R	Disk W	
Run Time	4m 9s	0.00 MB/s	2.06 MB/s	

### Query Text

```

select
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice,
  sum(l_quantity)
from
  customer,
  orders,
  lineitem
where
  o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 5
  )
  and c_custkey = o_custkey
  and o_orderkey = l_orderkey
group by
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice
order by
  o_totalprice desc,
  o_orderdate
;

```

### Plan & Progress (BETA)

#### Textual Plan

```

Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..6554.33 rows=2402130 width=49)
Merge Key: orders.o_totalprice, orders.o_orderdate
-> GroupAggregate (cost=0.00..6160.61 rows=600533 width=49)
  Group By: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
  -> Sort (cost=0.00..6115.27 rows=600533 width=48)
    Sort Key: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
    -> Hash Join (cost=0.00..2977.88 rows=600533 width=48)
      Hash Cond: public.lineitem.l_orderkey = orders.o_orderkey
      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
      -> Hash (cost=1888.16..1888.16 rows=124319 width=41)
        -> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..1888.16 rows=124319 width=41)
          Hash Key: orders.o_orderkey
          -> Hash Join (cost=0.00..1872.21 rows=124319 width=41)
            Hash Cond: orders.o_custkey = customer.c_custkey
            -> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..1371.15 rows=124319 width=22)
              Hash Key: orders.o_custkey
              -> Hash EXISTS Join (cost=0.00..1362.59 rows=124319 width=22)
                Hash Cond: orders.o_orderkey = public.lineitem.l_orderkey
                -> Table Scan on orders (cost=0.00..459.64 rows=374619 width=22)
                -> Hash (cost=784.20..784.20 rows=124319 width=4)
                  -> Result (cost=0.00..784.20 rows=124319 width=4)
                    Filter: (sum(public.lineitem.l_quantity)) > 5::numeric
                    -> HashAggregate (cost=0.00..773.98 rows=310798 width=12)
                      Group By: public.lineitem.l_orderkey
                      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
                  -> Hash (cost=434.95..434.95 rows=37596 width=23)
                    -> Table Scan on customer (cost=0.00..434.95 rows=37596 width=23)

```

Settings: optimizer=on  
Optimizer status: PQO version 2.58.0

The textual plan is the output of the Greenplum Database **EXPLAIN** command for the query. The query plan steps are labeled with arrows (->) and the structure of the query plan tree is indicated with indentation.

The **Optimizer status:** line at the bottom of the textual plan reports whether the explain plan was generated using the GPORCA optimizer (PQO) or the legacy query optimizer.

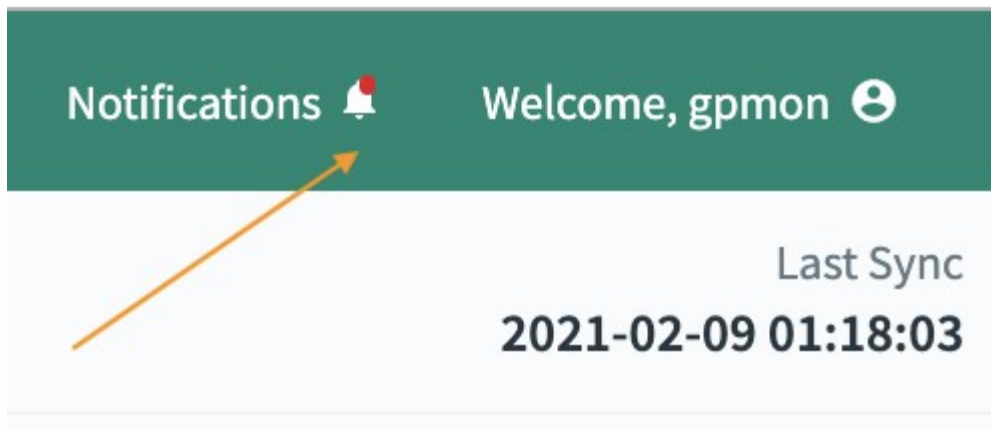
For help reading the textual explain plan see the **EXPLAIN** command in the *Greenplum Database Reference Guide* and **Query Profiling** in the *Greenplum Database Administrator Guide*. See **Query Execution** for descriptions of the query operators.

## Accessed Tables

The **Accessed Tables** panel displays the tables this query accessed.

## Alerts

To quickly view any notification alerts, use the **Notifications** label on the header bar of the Command Center dashboard.



When you open the **Notifications** window, unread notifications are highlighted in light blue. The alerts are ordered by time and are kept for 7 days. When you close the **Notifications** window, all alerts are considered read.

---

## × Notifications

New (84)

To better manage the type and frequency of the alerts, you can go to Admin > Alerts.

• 2021-02-08 12:37:08	[Urgent Alert] Master Fatal 2021-02-08 12:36:23: terminating connection due to administrator command
• 2021-02-05 22:12:00	[Urgent Alert] Master Fatal 2021-02-05 22:11:38: terminating connection due to administrator command
• 2021-02-05 18:08:24	[Urgent Alert] Master Fatal 2021-02-05 18:07:38: terminating connection due to administrator command
• 2021-02-05 12:33:22	[Urgent Alert] Master Fatal 2021-02-05 12:33:21: terminating connection due to administrator command
• 2021-02-04 19:55:37	[Urgent Alert] Master Fatal 2021-02-04 19:55:31: terminating connection due to administrator command
• 2021-02-04 18:36:15	[Urgent Alert] Master Fatal 2021-02-04 18:35:59: terminating connection due to administrator command
• 2021-02-08 12:36:21	[Alert] Segment Hosts Avg Memory exceeds 0% for 1 min
• 2021-02-08 12:36:16	[Alert] Number of connections exceeds 1
• 2021-02-08 12:36:12	[Alert] Number of connections exceeds 1
• 2021-02-08 12:35:58	[Alert] Number of connections exceeds 1

To customize the alert notifications, go to the **Admin > Alerts** page, where an administrator can set up **alert rules**. The rules help detect and respond to events that occur in the Greenplum Database system and in currently executing database queries. When a rule is matched, Command Center logs a record.

You can set up email alerts by configuring an SMTP server in Greenplum Database or in Command Center. Additionally, you can create a `send_alert.sh` shell script to forward alerts to other destinations, such as an SMS gateway or a Slack channel. If the script is present, Command Center runs it whenever an alert is raised.

Command Center creates the `gpmetrics` schema in the `gpperfmon` database to store both rules and log records. See [gpmetrics Schema Reference](#) for information about the `gpcc_alert_rule` and `gpcc_alert_log` tables in the `gpmetrics` schema.

This topic contains the following subtopics:

- [Configuring Alert Rules](#)

- [Configuring Alert Email](#)
- [Creating a Send Alert Script](#)

## Configuring Alert Rules

Click **EDIT** to manage alert event rules. To enable an alert rule, enter any data required in the fields and check the box. Uncheck the box to disable the rule. Click **SAVE** when you have finished making changes to the alert configuration.

Alerting
Set up automatic alerts when events are triggered

Current Time  
2020-05-28 05:00:54
?
Help

Receive email alerts for selected events:

CANCEL
APPLY

Cluster Level	<input checked="" type="checkbox"/> Database connectivity failure	<input checked="" type="checkbox"/> Number of connections exceeds <input type="text" value="50"/>
Master Level	<input checked="" type="checkbox"/> Master Panic <input checked="" type="checkbox"/> Master Fatal	<input type="checkbox"/> Master CPU exceeds <input type="text"/> % for <input type="text"/> min <input type="checkbox"/> Master Memory exceeds <input type="text"/> % for <input type="text"/> min
Segment Level	<input checked="" type="checkbox"/> Segment failure <input checked="" type="checkbox"/> Disk space exceeds <input type="text" value="85"/> % on a segment host	<input type="checkbox"/> Segment Hosts Avg CPU exceeds <input type="text"/> % for <input type="text"/> min <input checked="" type="checkbox"/> Segment Hosts Avg Memory exceeds <input type="text" value="65"/> % for <input type="text" value="10"/> min
Query Level	<input checked="" type="checkbox"/> Out of memory errors <input checked="" type="checkbox"/> Spill files for a query exceeds <input type="text" value="250"/> GB	<input type="checkbox"/> Query runtime exceeds <input type="text"/> min <input checked="" type="checkbox"/> Query is blocked for <input type="text" value="15"/> min

## Cluster Level

### Database connectivity failure

An alert is raised when either of these conditions is detected:

- Command Center is running on the master host, the master host is up, but the database is down or cannot be reached.
- Command Center is running on the standby master host, the standby master host is up, but the master host is down or cannot be reached, or the database is down or cannot be reached.
- Command Center checks three times before raising the alert. If the host where Command Center should be running is down, no alert is raised.

### Number of connections exceeds [N]

An alert is raised when the total number of database connections exceeds the number specified. The number of connections is checked every 30 seconds. After an alert is raised, the metrics collector checks the number of connections every 30 minutes until the number of connections drops below the threshold, and then it resumes checking every 30 seconds.

## Master Level

### Master Panic

An alert is raised when messages with PANIC severity are detected at the master host. An PANIC error causes all database sessions to abort.

#### Master Fatal

An alert is raised when messages with FATAL severity are detected at the master host. A FATAL error causes the current session to abort. To reduce noise, fatal messages logged due to password authentication errors do not raise alerts.

#### Master CPU exceeds [%] for [N] min

An alert is raised when the CPU usage on the master host exceeds the specified percentage for the specified number of minutes. Command Center samples CPU usage on the master host every 15 seconds and calculates the mean of the samples.

#### Master Memory exceeds [%] for [N] min

An alert is raised when the percent of memory used on the master host exceeds the specified percentage for the specified number of minutes. Command Center samples memory usage on the master host every 15 seconds and calculates the mean of the samples. Only memory in use is considered; memory for buffers and cache is not included.

## Segment Level

#### Segment failure

An alert is raised when one or more failed segments are detected. After the alert email is raised, Command Center will raise the alert every 30 minutes until the segments are recovered.

#### Total disk space exceeds [%] on a segment host

An alert is raised when the total of disk space in use for all segment hosts exceeds the specified percentage. Command Center gathers the *available disk space* and *total disk space* from each segment host in the Greenplum Database cluster. The *percent of total disk space in use* is calculated by the following formula:

$$100 - \text{sum}(\text{<available disk space>}) / \text{sum}(\text{<total disk space>}) * 100$$

A disk space alert is raised no more than once every 24 hours.

#### Segment Hosts Avg CPU exceeds [%] for [N] min

An alert is raised when the average percent of CPU used for all segment hosts exceeds the specified percentage for the specified number of minutes. Command Center samples all segment hosts every 15 seconds and calculates the mean of the samples.

#### Segment Hosts Avg Memory exceeds [%] for [N] min

An alert is raised when the average memory for all segment hosts exceeds the specified percentage for the specified number of minutes. Command Center samples all segment hosts every 15 seconds and calculates the mean of the samples. Only memory in use is considered; memory for buffers and cache is not included.

## Query Level

#### Out of memory errors

An alert is raised when an executing query fails with an out of memory (OOM) error. Note that no alert is raised if there is insufficient memory to start the query.

Spill files for a query exceeds [N] GB

An alert is raised when the total disk space consumed by a running query's spill files exceeds the specified number of gigabytes. An alert is raised only once per query.

Query runtime exceeds [N] min

An alert is raised when a query runtime exceeds the number of minutes specified. This alert is raised just once for a query.

Query is blocked for [N] min

An alert is raised if a query remains in a blocked state for longer than the specified number of minutes. If an alert is raised, and then the query unblocks, runs, and blocks again for the specified time, an additional alert is raised. Blocked time excludes the time a query is queued before it runs. It is possible for a "Query runtime exceeds [N] min" rule to also trigger while a query is blocked.

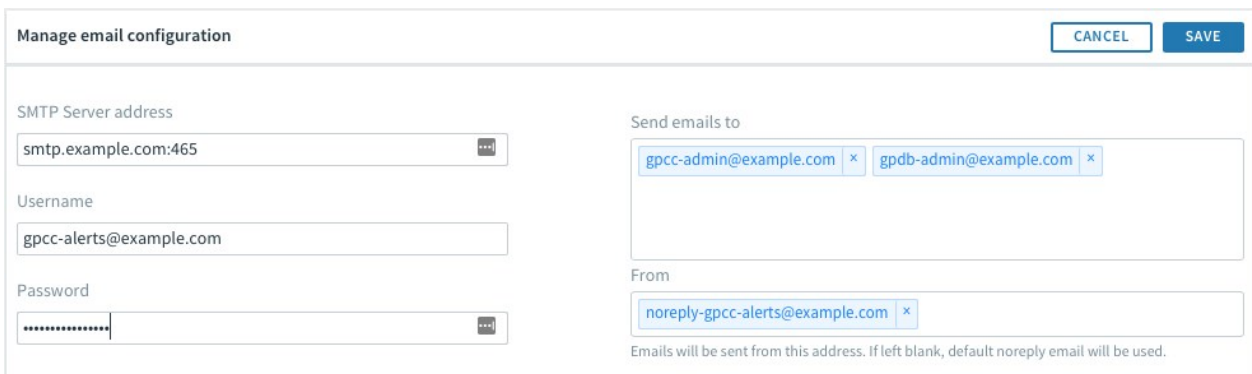
## Configuring Alert Email

Command Center requires an SMTP server to send alert emails.

Command Center first attempts an encrypted TLS connection and then falls back to an unencrypted connection if TLS is not supported. The SMTP server must support one of the following authentication methods: NONE, PLAIN, LOGIN, or CRAM-MD5. Command Center will use the most secure of these methods the SMTP server allows.

## Configuring email With Command Center

Click **EDIT** in the **Manage email configuration** panel.



The alert email configuration is set with the following Greenplum Database server configuration parameters:

SMTP Server address

The name or IP address of the SMTP server and the SMTP port number. The port number is typically 587 for connections with TLS encryption or 465 without encryption. Example: `smtp.example.com:465`

Username

The username of the account to authenticate with the SMTP server. This is an optional field, only required if the SMTP server requires a username for authentication. Example: `gpcc-alerts@example.com`



## Password

The password for the SMTP username. For security, the password is masked. This field is optional, only needed if the SMTP server requires a username and password for authentication.

## Send emails to

To add an address to the list, enter the address and press Enter. To remove an email address, click the **x** on the address.

## From

The email address to use for the **From:** address in the alert email. Example: `do-not-reply@example.com`.

If you leave this field blank, Command Center uses the default value, `noreply-gpcc-alerts@vmware.com`.

When you click **SAVE**, Command Center sends a test email to the addresses in the **Send emails to** field. The email contains a list of the currently configured alert rules. If there is an error in the SMTP server or username/password configuration and the email cannot be sent, Command Center displays an error message.

## Creating a Send Alert Script

The send alert script is a shell script that you can use to send Command Center alerts to destinations such as SMS gateways, pagers, team collaboration tools like Slack, chat servers, archive files, alternative email servers, and so on. You can use the send alert script in addition to sending email from Command Center, or as an alternative to sending alert emails from Command Center.

Command Center looks for the script `$HOME/gpmetrics/send_alert.sh` on the host where Command Center is running—either the master host or standby host. If the file exists and is executable by the `gadmin` user, Command Center executes the script. The following variables are set on the command line when the script runs.

Variable	Description
LINK	URL of the Greenplum Command Center web server.
QUERYID	ID of the query, if the alert was triggered by a query.
SERVERNAME	Name of the Greenplum Command Center server.
QUERYTEXT	The text of the query, if the alert was triggered by a query.
ACTIVERULENAME	Current text of the rule, with user-specified values included.
LOGID	Value of this alert's <code>id</code> column in the <code>gpmetrics.gpcc_alert_log</code> table.
RULEDESCRIPTION	Text of the rule, including user-specified values, at the time the alert was raised.
ALERTDATE	Date the alert was raised.
ALERTTIME	Time the alert was raised.
SUBJECT	Subject line for email.

An example script that you can customize is provided at `$GPCC_HOME/alert-`

`email/send_alert.sh.sample`. The example formats the alert as HTML email text and pipes it through the Linux `mail` command.

To set up a send alert script:

1. Copy the `$GPCC_HOME/alert-email/send_alert.sh.sample` file to `$HOME/gpmetrics/send_alert.sh`.
2. Customize the script with code to format and deliver the alert to your desired destination.
3. Run `gpcc start` to restart Command Center and enable the script.

## SNMP Alert Script Example

SNMP support is discontinued in Greenplum Database 6, but you can use a Command Center alert script to send alerts to an SNMP manager.

This example alert script sends Command Center alerts as SNMP notifications in a format similar to Greenplum Database 5 SNMP notifications, using the SNMP OIDs for Greenplum Database as arguments.

This is the OID hierarchy structure for Greenplum Database:

```
iso(1)
  identified-organization(3)
    dod(6)
      internet(1)
        private(4)
          enterprises(1)
            gpdbMIB(31327)
              gpdbObjects(1)
                gpdbAlertMsg(1)
```

The following table lists the Greenplum Database OIDs:

Name	Address	Type: Description
gpdbAlertMsg	1.3.6.1.4.1.31327.1.1	string: alert message text
gpdbAlertSeverity	1.3.6.1.4.1.31327.1.2	integer: severity level
gpdbAlertSqlState	1.3.6.1.4.1.31327.1.3	string: SQL standard error codes
gpdbAlertDetail	1.3.6.1.4.1.31327.1.4	string: detailed alert message text
gpdbAlertSqlStmt	1.3.6.1.4.1.31327.1.5	string: SQL statement generating this alert, if applicable
gpdbAlertSystemName	1.3.6.1.4.1.31327.1.6	string: hostname

Command Center sets environment variables that can be mapped to the `gpdbAlertMsg`, `gpdbAlertSqlStmt`, and `gpdbAlertSystemName` OIDs. For the other OIDs, the script sends an empty string or 0, depending on data type.

## Prerequisites

The SNMP service must be configured and enabled on the host where Command Center is running. If the `snmpd` file is not present in the `/usr/sbin` directory, then SNMP is not installed on the system.

To install SNMP on Red Hat and CentOS, use `yum` to install the `net-snmp`, `net-snmp-libs`, and `net-snmp-utils` packages. Configure SNMP following the operating system documentation. This example script specifies the SNMP community `public`, but you can set a different community string with the `rocommunity` parameter in the `/etc/snmp/snmpd.conf` file.

Start the `snmpd` daemon with the `systemctl` command:

```
$ sudo systemctl enable snmpd.service
```

Verify the system `snmpd` daemon is running with this command:

```
$ sudo snmpwalk -v 1 -c public localhost .1.3.6.1.2.1.1.1.0
```

If the daemon is running, the `snmpwalk` command output is similar to the following:

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux gp-mdw 3.10.0-957.12.2.el7.x86_64 #1 SMP Tue Ma
y 14 21:24:32 UTC 2019 x86_64
```

## Setting up SNMP Notifications

1. As the `gpadmin` user, edit the `$HOME/gpmetrics/send_alert.sh` file.
2. Locate the comment that reads:  
"Following is an example for sending email with above input arguments Can be replaced with any operation, such as SMS, slack, wechat, etc..."
3. Replace the content **between that comment and the last ";" character** with the following code snippet:

```
RUNNINT_TIME=`psql -c 'select extract(epoch from (current_timestamp - pg_postma
ster_start_time())) * 100;' | sed -n '3p' | awk '{print $1}'`
HOST=`hostname`
snmptrap -v 2c -c public \
    {##trapserver##} \
    "${RUNNINT_TIME}" \
    "enterprises.31327.5.0.1" \
    SNMPv2-SMI::enterprises.31327.1.1 s "${RULEDESCRIPTION}" \
    SNMPv2-SMI::enterprises.31327.1.2 i 0 \
    SNMPv2-SMI::enterprises.31327.1.3 s "" \
    SNMPv2-SMI::enterprises.31327.1.4 s "" \
    SNMPv2-SMI::enterprises.31327.1.5 s "${QUERYTEXT}" \
    SNMPv2-SMI::enterprises.31327.1.6 s "${HOST}"
```

Also, replace `{##trapserver##}` with the name of the host running `snmpd`.

4. Save and exit.
5. Make the script executable.

```
$ chmod +x $HOME/gpmetrics/send_alert.sh
```

6. Restart Command Center to enable the script.

```
$ gpcc start
```

## Testing the SNMP Alert Script

To test the script, you can use the `snmptrapd` trap receiver.

1. As root, edit the `/etc/snmp/snmptrap.conf` file and add the following line to allow the daemon to receive unauthenticated notifications:

```
disableAuthorization yes
```

2. Start the `snmptrapd` daemon with this command:

```
# /usr/sbin/snmptrapd -m ALL -Lf ~/snmpptest.log
```

Check the `/home/root/snmpptest.log` file for notifications.

# Managing Users

User Management

View users and their resource consumptions

Last Sync

2021-11-12 02:34:18

Help

Time

Oct 2021

Role Name

Filter by

Department

Departments (6/7)

Permission

All

SEARCH

EXPORT ALL

Role Name	Department	Permissions	CPU Time	Memory	Total Queries	Spill Files	Disk Read	Disk Write
gptest	IT	Operator Basic	15h 56m 48s	660.07 TB	57,408	0	17.67 GB	17.67 GB
gptest1	Finance	Basic	23h 17m 40s	964.21 TB	83,860	307.58 GB	0	0
gptest10	Sales	Operator	0s	0	0	0	0	0
gptest100	Sales	Basic	7d 0h 43m 24s	70.64 GB	6	117.19 KB	0.75 KB	5.86 MB
gptest1000	Support	Admin (superuser)	7d 0h 43m 24s	70.64 GB	6	58.59 KB	7.50 KB	600.00 KB
gptest101	Support	Admin (superuser)	34d 22h 20m 0s	75.00 MB	6	117.19 KB	75.00 KB	60.00 KB
gptest105	IT	Basic	28d 4h 40m 0s	37.50 MB	6	292.97 KB	52.08 KB	27.48 KB

The Command Center's **User Management** view allows administrators to perform a variety of user management tasks, including:

- Displaying resource usage for each role for a specific month
- Displaying resource usage for specific roles
- Displaying resource usage for one or more specific departments
- Displaying resource usage for roles with no department
- Displaying roles holding a specific permission
- Displaying resource usage for all permission levels
- Creating a department name
- Assigning a department name to a specific role
- Changing a role's permissions
- Exporting resource usage of specific roles
- Renaming a department using a user-defined function
- Deleting an unwanted department name from the `gpperfmon` database using a user-defined function

The resources displayed include:

- CPU Time
- Memory
- Total Queries
- Spill Files
- Disk Read

- Disk Write

## Display Resource Usage by Role Per Month

This task requires Operator Basic or higher permission.

To display resource usage by role per month:

1. Click the Time dropdown menu
2. Select a month
3. Click the SEARCH button

Command Center displays resource usage for all roles for that month.

## Display Resource Usage for Specific Roles

This task requires Operator Basic or higher permission.



**Note:** The results returned may not cover up to the preceding 6 hours, due to delays.

To display resource usage for specific roles:

1. Enter the role name in the Role Name box
2. Click the SEARCH button

Command Center displays resource usage for that role.



**Note:** The resource usage of the gpmon and gpadmin roles is not included.

## Display Resource Usage for Departments

These tasks require Operator Basic or higher permission.

To display resource usage for a specific department:

1. Select a department from the Department dropdown menu
2. Click the SEARCH button

Command Center displays resource usage for that department.

To display resource usage for all departments:

1. Select all departments from the Department dropdown menu
2. Click the SEARCH button

Command Center displays resource usage for all departments.

To display resource usage for a selection of departments:

1. Select multiple departments from the Department dropdown menu
2. Click the SEARCH button

Command Center displays resource usage for this selection of departments.

## Display Resource Usage for Roles with No Department

This task requires Operator Basic or higher permission.

To display resource usage for roles with no department:

1. Select "--" from the Department dropdown
2. Click the SEARCH button

Command Center displays resource usage for roles who have no department.

## Display Resource Usage for Roles Holding a Specific Permission

This task requires Operator Basic or higher permission.

To display resource usage for roles holding a specific permission:

1. Select a permission from the Permission dropdown menu
2. Click the SEARCH button

Command Center displays resource usage for roles with that permission.

## Display Resource Usage for All Permission Levels

This task requires Operator Basic or higher permission.

To display resource usage for all permission levels:

1. Select "All" from the "Permission" dropdown menu
2. Click the SEARCH button

Command Center displays resource usage for all permission levels.

## Create a Department Name

This task requires Admin permission.

To create a department name and apply it to a specific role:

1. Click the dropdown menu in the Department column
2. Enter a new department name
3. Hit Enter
4. Click the Department dropdown again; the new department name is now visible

## Assign a Department Name to a Specific Role

This task requires Admin permission.

To assign a department name to a specific role:

1. Click the dropdown menu in the Department column
2. Select a name from the list
3. That department name is now associated with the role

## Change a Role's Permissions

This task requires Admin permission.

To set a role's permission level:

1. Click the dropdown menu in the Permissions column
2. Select the desired permission
3. That permission is now associated with the role

## Export Resource Usage of Specific Roles

This task requires Operator Basic permission.

To export resource usage of specific roles:

1. Specify search conditions in the **User Management** interface
2. Click the SEARCH button
3. View the resource usage report
4. Click the EXPORT ALL button

A CSV file that begins with the string `gpcc_user_usage_report` is downloaded to your machine's default download location.

## Rename a Department with the `gpcc_update_department` User-Defined Function

You may use the `gpcc_update_department` user-defined function to rename a department name, passing it the old name (`from_name`) and the new name (`to_name`). Once it has run, all the roles associated with the old name will now be associated with the new name.

For more information on the `gpcc_rename_dept_name` user-defined function, see the [gpmetrics Schema Reference](#) documentation page.

## Delete a Department Name from the gpperfmon Database with the `gpcc_delete_department` User-Defined Function

You may use the `gpcc_delete_department` user-defined function to delete a department name, passing it the name of the department you want to delete. You may only delete department names that have no roles associated with them.



For more information on the `gpcc_delete_department` user-defined function, see the [gpmetrics Schema Reference](#) documentation page.

# Data Loading

[About Data Loading](#)

[Configuring GPSS Server Access](#)

[Creating and Managing Data Loading Jobs](#)

[Viewing Data Loading Logs](#)

## About Data Loading with Command Center

Command Center allows you create and manage [Greenplum Streaming Server \(GPSS\)](#) client jobs for loading data from external sources into Greenplum Database. The Command Center data loading feature can be thought of as a wrapper around the [GPSS client utility `gpsscli`](#), which is used for creation and management of GPSS client data loading jobs.

**NOTE:** Currently, Amazon S3 is the only external data source type supported and CSV is the only data file format supported.

Via the **Data Loading** view, you perform three categories of tasks:

- Job creation and management, in the **Data Loading> Jobs** view. When you create a job, you may do so either by entering data in the UI via a 5-step process -- if your source data has fewer than 30 columns of data -- or by uploading a YAML file. For a full explanation of job creation and management, see [Creating and Managing Data Loading Jobs](#).
- Reviewing of log data, in the **Data Loading> Action Logs** view. The log portion of the UI reports a variety of general information about data loading jobs and, for each job, lets you drill down for more details. For more information, see [Viewing Data Loading Logs](#).
- Configuration and viewing of GPSS server access information, in the **Data Loading> Settings** view. For more information, see [Configuring GPSS Server Access](#).

**NOTE:** You must configure GPSS server access before creating jobs.

## Configuring GPSS Server Access

In order to create and manage data loading jobs from Command Center, you must configure access to a GPSS Server, via the **Data Loading> Settings** view.

Data Loading	Current Time 2022-04-09 03:45:33
--------------	-------------------------------------

Jobs	Action Logs	<u>Settings</u>
------	-------------	-----------------

**GPSS Configuration**[EDIT](#)

GPSS Address	Port
10.117.190.96	5050
Username	
test_user	
Password	
*****	

**Authentication**

Authentication Method

Without TLS

1. To configure client access to a GPSS Server for the first time or edit an existing configuration, click **EDIT** and fill in the following fields:

- **GPSS Address**

The hostname or IP address of the GPSS server.

- **Port**

The port the GPSS server is listening on for client connections.

- **Username**

The client username, configured on the GPSS server side. For more information, see [Configuring the Streaming Server for Client-to-Server Authentication](#).

- **Password**

The client password, configured on the GPSS server side. For more information, see [Configuring the Streaming Server for Client-to-Server Authentication](#).

- **Authentication Method**

There are three possible authentication methods:

- ✦ Without TLS. This means the client connection to the GPSS server will be an unencrypted gRPC connection. This is the default.
- ✦ TLS. The client will use Transport Layer Security for authentication and Command Center will verify that the GPSS server's certificate is signed by a Certificate Authority (CA). Command Center displays a second input box for a "CA Certificate File. You may leave it blank if your certificate is issued by a well-known CA. Otherwise, you must provide a CA certificate file located on the Command Center server host and readable by Command Center.
- ✦ TLS (CA Skip Checks). This configuration is for clients with self-signed certificates. The client will use Transport Layer Security for authentication, but Command Center will not verify that the GPSS server's certificate has been signed by a CA.

2. Click **APPLY** to apply changes. Click **Cancel** to leave without saving changes.

**NOTE:** Before a GPSS server connection has been configured for the first time, all fields will be empty except for **Port**, which will be set to the default: 5000.

For more information on setting up encrypt connections, see the section "Configuring gpss and gpsscli for Encrypted gRPC Communications" in [Configuring the Streaming Server for Encryption and Authentication](#).

## Creating and Managing Data Loading Jobs

Command Center allows you create and manage [Greenplum Streaming Server \(GPSS\)](#) client jobs for loading data from external sources into Greenplum Database. Currently, the only external data source type supported is Amazon S3.

You access Command Center's data loading feature via the **Data Loading > Jobs** view.

Data Loading

Current Time  
2022-04-14 03:02:56

Jobs

Action Logs

Settings

+ Create

▶ Start

□ Stop

✕ Remove

<input type="checkbox"/>	Job ID	Job Name	Status	Type	Creator	Start Time	Stop Time
<input type="checkbox"/>	73b017b61c7a95e0c432071635186934	gpcc_msv...	Submitted	S3	gpmon	--	--
<input type="checkbox"/>	8aab84812c3c091ef5b2ba8e5bdecc7c	gpcc_msv...	Error	S3	gpmon	2022-04-13 02:33	2022-04-13 02:33
<input type="checkbox"/>	37048fc3f705eb312911569762ca12d6	gpcc_msv...	Success	S3	gpmon	2022-04-13 02:24	2022-04-13 02:24
<input type="checkbox"/>	4ebd5688d04c5fd572453af37e5c99f6	gpcc_msv...	Error	S3	gpmon	2022-04-13 02:33	2022-04-13 02:33
<input type="checkbox"/>	b471b8685de0b24899fa0e545ffaf9a4	gpcc_msv...	Success	S3	gpmon	2022-04-13 02:33	2022-04-13 02:33
<input type="checkbox"/>	1bbf29ab55b794ffa8cd5edbd201580f	gpcc_msv...	Success	S3	gpmon	2022-04-13 02:41	2022-04-13 02:41

## Prerequisites for Data Loading Feature

The following are the prerequisites for using the Command Center data loading feature:

- You must configure GPSS server access before creating jobs. See [Configuring GPSS Server Access](#) for details.
- You must install GPSS server version 1.7.0 or higher and configure it correctly.
- The GPSS extension must be registered on the target database.

## Creating Jobs

There are two different workflows for creating your job:

- The "Default Creation Workflow", in which you enter data in the UI via a 5-step process. This approach requires that your source data have fewer than 30 columns of data.
- The YAML file workflow, in which you upload a YAML file specifying the properties of your desired job configuration.

## Creating a Job with the Default Creation Workflow

Follow these steps to create a data loading job by entering data in the UI:

1. Click **Create**.
2. The "Default Creation Workflow" radio button is already selected; simply click **Next**.
3. Fill in the fields as follows:
  - ✦ Enter the Job Name
  - ✦ The Source Type defaults to S3, the only supported source type at this time
  - ✦ Enter the S3 File Location. The S3 File Location should be in this format: `s3://s3."Region".amazonaws.com/"bucket-name"/"key-name"`, for example `s3://s3.us-west-2.amazonaws.com/mybucket/mydata.csv`. For more information, see [this section](#) in the Greenplum Database Administrator Guide.
  - ✦ Enter the S3 Access Key
  - ✦ Enter the S3 Secret Key
  - ✦ The File Type defaults to CSV, the only supported file type at this time
  - ✦ Under "Advanced File Options", choose CSV options
4. Click **Next**.
5. Define the input columns for your source data; this includes the column name and datatype for each column.
 

**NOTE:** You must add every column from the data source file; for example, if there are 30 columns in the data source file there must be 30 rows added to this page. Otherwise, data loading will fail.
6. Click **Next**.
7. Enter information about the Greenplum Database table you want to load data into. This information includes:
  - ✦ the mode: the choices are INSERT, UPDATE, or MERGE
  - ✦ the database name
  - ✦ the schema name
  - ✦ the table name
  - ✦ the error limit, which determines the number of errors that can occur before the data loading job must be stopped. This can be a number or a percentage.
  - ✦ the hostname or IP address of the master host
  - ✦ the username of the GPDB user loading the data into the database
  - ✦ the password of the GPDB user loading the data into the database
8. Click **Next**.
9. Command Center populates the **Column Name** and **Type** columns with data from the table you provided in Step 7. Use the **Expression** column to enter expressions for transforming the data extracted from the S3 data source into column data for the Greenplum Database table

you're loading data into. For details on how to set up mappings of source columns to output columns, see the [filesource-v3.yaml](#) topic.

**NOTE:** Command Center will discard any row data for output rows whose **Expression** column you leave blank.

10. Click **Next** to bring up the "Review and Start" screen.
11. At this point you have one of three choices:
  - ❖ Click **Submit** to submit the load to the GPSS server instance; GPSS then registers the job in its job list. The job now has a status of "Submitted."
  - ❖ Click **Dry Run** to perform a trial load without writing to Greenplum Database.
  - ❖ Click **Start Now** to start up the job. The job now has a status of "Started."

## Creating a Job by Uploading a YAML File

Follow these steps to create a data loading job using a YAML file:

1. Click **Create**.
2. Click the radio button in the "Upload YAML File" box.
3. Click **Next** to bring up the "Choose Input Source" screen.
4. Enter a job name. The job name is composed of:
  - A prefix, which is made up of "gpcc\_" followed by another underscore.
  - The remainder of the job name. The default for this portion is the current GPCC login user name followed by an underscore, followed by a timestamp.

Here is a sample job name: `gpcc\_msvm\_gpmon\_202204070412`

5. Choose a YAML file to upload, by clicking **CHOOSE FILE** and entering a path to the YAML file you want to upload. To create a YAML file, you can refer to [this sample YAML](#), which is a sample GPSS load configuration file for loading data from an S3 data source.
6. Click **Next** to bring up the "Review and Start" screen.
7. At this point you have one of three choices:
  - ❖ Click **Submit** to submit the load to the GPSS server instance; GPSS then registers the job in its job list. The job now has a status of "Submitted."
  - ❖ Click **Dry Run** to perform a trial load without writing to Greenplum Database.
  - ❖ Click **Start Now** to start up the job. The job now has a status of "Started."

## Managing Jobs

In addition to creating new jobs, you may start, stop, or remove existing jobs:

- To start a job, select the job and click **Start**.
- To stop a job, select the job you want to stop and click **Stop**.
- To remove a job, select the job you want to remove and click **Remove**.

To perform an operation on multiple jobs, select the desired jobs and then click **Start**, **Stop**, or **Remove**.

## Permissions Required for Job Management

All users can view, start, stop, remove, and view logs for jobs they created. However, the superuser -- known as "Admin" -- role is required to perform these actions on jobs created by other users.

## About Job IDs

Once a job is created, it is assigned a job ID. A job's ID is determined by its data source and the target (database name plus schema name plus table name).

**IMPORTANT:** The same combination of a data source and a target always produces the same job ID. This means that if a user creates a job with the same data source and target with an existing job, then the job is not created as a new job. Instead, the existing job is updated because the ID is identical.

## Viewing Data Loading Logs

Command Center logs a variety of information related to data loading jobs. Using the **Data Loading> Action Logs** view, you can track the status of jobs by job id and job name and view any error messages associated with failed jobs.

**Note:** Unless you hold the Admin ("superuser") role, you may view logs only for jobs that you created.

Data Loading					Current Time 2022-04-09 02:54:05
Jobs	Action Logs	Settings			
Timestamp	Job ID	Job Name	Detail	Source	Error Message
2022-04-01 10:54:12	37048fc3f705eb312...	<a href="#">gpcc_msvm_abc</a>	The job status has changed to Success	GPSS	
2022-04-01 10:54:12	73b017b61c7a95e0c...	<a href="#">gpcc_msvm_fa</a>	The job status has changed to Success	GPSS	
2022-04-01 11:51:31	5e8d5641ff4b5b1a8...	<a href="#">gpcc_msvm_ibm3</a>	The job status has changed to Submitted	GPSS	
2022-04-01 10:54:12	8aab84812c3c091ef...	<a href="#">gpcc_msvm_vvv</a>	The job status has changed to Success	GPSS	
2022-04-07 06:47:48	--	--	[Failed] Submit the job	GPCC	Jobname[gpcc_msvm_gpmon_2...
2022-04-01 11:51:31	5e8d5641ff4b5b1a8...	<a href="#">gpcc_msvm_ibm3</a>	The job status has changed to Submitted	GPSS	
2022-04-01 11:51:31	5e8d5641ff4b5b1a8...	<a href="#">gpcc_msvm_ibm3</a>	[Succeeded] Submit the job	GPCC	
2022-04-01 11:50:33	5e8d5641ff4b5b1a8...	<a href="#">gpcc_msvm_ibm3</a>	The job status has changed to Submitted	GPSS	
2022-04-01 11:45:25	5e8d5641ff4b5b1a8...	<a href="#">gpcc_msvm_ibm3</a>	The job status has changed to Submitted	GPSS	

The **Data Loading> Action Logs** view displays the following information:

### Timestamp

The time Command Center detected the job status.

### Job ID

The unique identifier of the job.

## Job Name

The name of the job. Click the job name to display:

- Basic job information such as ID, start time, stop time, the name of the Command Center user that created it, and the name of the Greenplum Database user loading data into Greenplum Database.
- Its latest output, including an error log if pertinent
- Its configuration settings
- The YAML file associated with the job

## Detail

What the job's status has most recently changed to. This includes whether the job has been started, stopped, submitted, or removed and whether starting, stopping, submission, and removal succeeded or failed.

## Source

The source of the information about the job's status: either GPCC or GPSS.

## Error Message

An error message related to the job status, if applicable.



# Managing Greenplum Database Workloads

About Workloads

Managing Greenplum Database Resource Groups

Importing Resource Queues to Resource Groups

Troubleshooting Enabling Resource Groups

## About Greenplum Command Center Workload Management

Greenplum Database segment hosts have a set amount of memory, CPU, I/O, and network resources. If these resources start to run short due to heavy database load, queries can fail or the entire database system can become unacceptably slow. For this reason, it is important to manage database requests to prevent resource depletion.



**Note:** Greenplum Database has two resource management systems: *resource queues* and *resource groups*. Command Center workload management is based on resource groups, the resource management system introduced in Greenplum Database version 5. Resource groups require enabling Linux control groups (cgroups), so Greenplum Database initially is set to perform resource management using resource queues.

When Command Center starts, it checks the resource management system enabled in your Greenplum Database system. If you have not yet enabled resource groups in your Greenplum Database system, the Command Center workload management view displays a message encouraging you to enable resource groups, including a link to documentation with the procedure for completing the task. When you start Command Center after enabling resource groups, click the **ENABLE WORKLOADS** button. Command Center presents a view to help you set the initial resource group configuration by importing your existing resource queues to resource groups.

See [Using Resource Groups](#) in the *Greenplum Database Administrator Guide* for a full description of resource management features available with resource groups and instructions to enable resource groups in Greenplum Database.

On the Command Center **Workload> Resource Groups** view, you can perform the following tasks:

- Create new resource groups in Greenplum Database
- Delete existing resource groups
- Change the number of concurrent transactions each resource group allows

- Change the percentages of available system CPU and memory each resource group manages
- Change the default resource group assignment for Greenplum Database roles

See [Resource Groups](#) for details about creating resource groups and query timeout rules.

For more information about Linux cgroups and Greenplum Database resource groups see [Using Resource Groups](#) in the *Greenplum Database Administrator Guide*.

On the Command Center **Workload> Workload Mgmt** view, you can perform the following tasks:

- Create workload management rules to cancel a query or move it to a different resource group, based on conditions such as the amount of CPU time used, planner cost, query running time, and disk I/O utilization.
- Create workload management rules to terminate an idle session based on which resource group it belongs to and how long it has been idle.
- Monitor workload management rules when they perform actions against matching queries or idle sessions.

Workload management rules allow you to flexibly assign transactions to resource groups or cancel queries or idle sessions. The following sections provide more information about these Command Center features.

## About Query Assignment Rules and Workload Rules

Greenplum Database defers to the workload management database extension to assign transactions to resource groups. Command Center users with Admin permission level can create assignment rules in Command Center to assign transactions to resource groups before they run.

When a transaction begins, Greenplum Database calls the metrics collector extension to determine the resource group. The extension evaluates the assignment rules and, if a matching rule is found, returns that rule's resource group. If no assignment rule matches, Greenplum Database falls back to the default behavior, assigning the transaction to the resource group specified for the current user in the `pg_authid` system table.

Assignment rules can redirect a transaction to a resource group based on any combination of query tag and/or the Greenplum Database role that executes the transaction.

A query tag is a user-defined `name=value` pair that you can set in a Greenplum Database session when resource group-based workload management is enabled. Query tags are defined by setting the `gpcc.query_tags` parameter on the connect string when requesting a database connection or in the session with `SET gpcc.query_tags TO '<query-tags>'`. Multiple query tags can be set by separating them with a semicolon. Query tags are set before a transaction begins and cannot be changed inside of a transaction.

If multiple assignment rules would match the same query, Command Center matches only the first assignment rule (the rule with the lowest ID number) to the query.

See [Workload Management](#) for details about creating assignment rules, and for examples that use query tags.

## Workload Rules

A workload rule defines:

- For workload rules for queries, additional conditions beyond query assignment rule conditions under which a running query may be canceled or (for Greenplum versions 6.8 or later) moved into a different resource group

Workload rules for queries are first matched to a transaction based on any combination of the database role and query tags. After the transaction execution begins, additional conditions are evaluated to determine if configured CPU time, I/O, slice, and/or planner cost limits are exceeded.

- For workload rules for idle sessions, the conditions under which an idle session may be terminated

If all configured conditions for a workload rule are exceeded, the rule action is performed against the transaction. Available actions are either cancelling the transaction or, for Greenplum versions 6.8 or later, moving the transaction to a different resource group.

Command Center always ensures that a given workload rule is applied only once to a matched transaction. For example, two workload rules could be created with one rule that moves queries from resource group A to group B based on planner cost, and another rule that moves queries from any resource group to group A based on the query running time. The first of these rules would only be applied to a given query once, even if the second rule again placed the query back into resource group A.

See [Workload Management](#) for details about creating workload rules.

## Resource Groups

### Defining Resource Groups and Resource Attributes

Command Center allows you to view resource groups that have been created in Greenplum Database, to add or delete resource groups, and to edit the resource group attributes **Concurrency**, **CPU %**, and **Memory %**.



To change values of the `MEMORY_AUDITOR`, `CPuset`, `MEMORY_SHARED_QUOTA`, or `MEMORY_SPILL_RATIO` resource group attributes, use the `ALTER RESOURCE GROUP` SQL command.

Resource Groups						<a href="#">CANCEL</a>	<a href="#">APPLY</a>
Name	Concurrency	CPU %	Memory %	Memory Spill Ratio %	Statement Memory		
ad_hoc	<input type="text" value="3"/>	<input type="text" value="10"/>	<input type="text" value="12"/>	<input type="text" value="75"/>	4164.59MB		
admin_group	<input type="text" value="5"/>	<input type="text" value="20"/>	<input type="text" value="25"/>	<input type="text" value="30"/>	2082.30MB		
default_group	<input type="text" value="10"/>	<input type="text" value="40"/>	<input type="text" value="30"/>	<input type="text" value="75"/>	3123.45MB		
priority_low	<input type="text" value="3"/>	<input type="text" value="10"/>	<input type="text" value="10"/>	<input type="text" value="75"/>	--		
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	--		
<a href="#">+ ADD RESOURCE GROUP</a>						80	77

- Click **EDIT** to open the Resource Group editor.
- To delete a resource group, select the resource group, and click the minus sign that appears

at the right.

You cannot delete the `default_group` or `admin_group` resource groups. You cannot delete a resource group that is assigned to any Greenplum Database role.

3. To add a resource group, click **ADD RESOURCE GROUP** and enter a name for the resource group in the **Name** column. Resource group names must be unique and are case-sensitive.
4. Adjust the values of the **Concurrency**, **CPU %**, and **Memory %** resource group attributes.

### Concurrency

The maximum number of concurrent transactions, including active and idle transactions, that are permitted in the resource group. **Concurrency** sets the `CONCURRENCY` attribute of the resource group. The total of the **Concurrency** columns cannot exceed the value of the Greenplum Database `max_connections` master server configuration parameter. **CPU %** The percentage of CPU resources available to this resource group. The percentage is the portion of the total CPU percentage allocated for all resource groups (reserved CPUs excluded), which is set with the `gp_resource_group_cpu_limit` server configuration parameter. **CPU %** sets the `CPU_RATE_LIMIT` attribute of the resource group. **Memory %** The percentage of memory resources available to this resource group. The percentage is the portion of the total memory allocated for all resource groups, which is set with the `gp_resource_group_memory_limit` Greenplum Database configuration parameter. Changing the **Memory %** value sets the `MEMORY_LIMIT` attribute of the resource group. **Memory Spill Ratio %** The memory usage threshold for memory-intensive transactions. When a transaction reaches this threshold, it spills to disk. **Statement Memory** The amount of memory allocated to a query. This column is recalculated as you adjust other resource group settings. If a query needs more memory, it is allocated from the resource group shared memory pool and the global shared memory pool, if available.

The totals of the **CPU %** and **Memory %** columns must not exceed 100%. You should not allow the total of the **CPU %** column to exceed 90%, because this could cause resource-intensive queries to consume nearly all CPU, starving other Greenplum Database processes. If the total of the **Memory %** column is less than 100%, the unreserved memory is part of the resource group shared global memory pool. See "Global Shared Memory" in [Using Resource Groups](#) in the *Greenplum Database Administrator Guide* for information about the global resource group shared memory pool.

5. Click **Apply** to save your changes or click **Cancel** to abandon your changes.

## Assigning Roles to Resource Groups

Every Greenplum Database role is assigned to a single resource group in the `pg_roles` system table. Transactions executed by a role are managed by its assigned resource group, unless you create an assignment rule to override the default.

You can view the current resource group assignments for all roles and change a role's resource group by adding it to a different resource group.

**Assignment by Role**  
Queries are routed to Resource Group based on gpdb role, unless diverted by Query Tag filter(below)

default_group(3)	admin_group(2)	etl(1)	priority_low(2)	priority_high(1)
ralph sallyr cashk	gpadmin gpmon	nickd	tpch_4 tpch_1	kristiem
<input type="text" value="add role"/>	<input type="text" value="add role"/>	<input type="text" value="add role"/>	<input type="text" value="add role"/>	<input type="text" value="add role"/>

To move a role to a different resource group:

1. Enter all or part of the role name in the **add role** field beneath the new resource group.
2. Choose the role from the list that is displayed and press Enter.

The change is immediately applied to the Greenplum Database `pg_roles` system table.

## Monitoring Resource Groups

You can determine whether your resource group settings are optimal by viewing resource utilization of resource groups, using the Command Center **Resource Groups> Monitor** view.



You may view the following information:

- CPU utilization of a single resource group
- CPU utilization of a selection of resource groups
- Memory utilization of a specific resource group
- Memory utilization of a selection of resource groups
- Concurrency status of selected resource groups

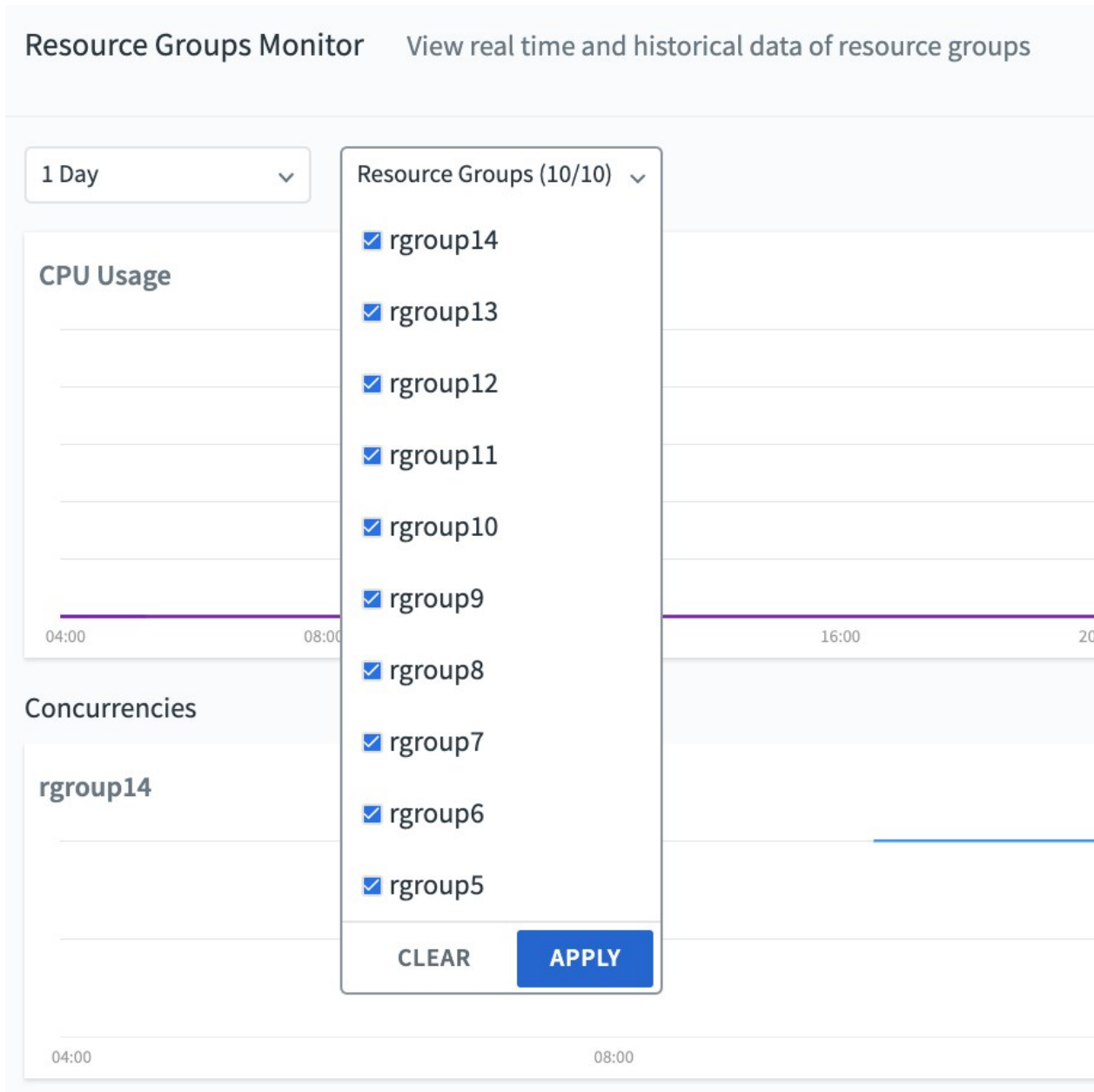
**Note:** To perform these tasks, you must have a permission level of Operator Basic or above.

## About Resource Group Selection

Using the dropdown menus on the Resource Groups Monitor page, you may select:

- the time interval for the data you are interested in viewing

- the resource groups whose data to display; you may select up to 10 resource groups at a time. Once you have selected resource groups whose information you want to view, you may click **Apply** to save the selection or **Clear** to clear it.



## Workload Management

### Defining Workload Management Rules

You can define workload management rules for both queries and for idle sessions.

#### Defining Workload Management Rules for Queries

Query assignment rules enable you to configure which resource group a transaction uses before the transaction begins. Workload rules monitor additional conditions for queries, such as the amount of CPU time or disk I/O consumed, and enable you to cancel a running query or (for Greenplum version 6.8 or later) move a query to a different resource group after it has started. You configure

both types of rule on the **Workload> Workload Mgmt** page using the same rule interface.

Transactions are first matched to a configured rule using any combination of user-defined query tags, the current role in the database session, or the resource group originally used for the transaction. When no rule matches, the transaction remains assigned to the role's default resource group. See [About Assignment Rules](#) for more information about assignment rules.

A rule that provides additional conditions acts as a query assignment rule, and determines the resource group where the statement should run.

A rule that provides additional conditions such as a maximum CPU time, Disk I/O, planner cost, or slices used, is a workload rule. With workload rules, Command Center monitors the query to evaluate those conditions. If all conditions are met, the workload rule action is performed.

A additional configuration parameter, `wlm_short_query_threshold`, is provided to ensure that only queries that run for the configured number of seconds are canceled or moved according to workload rules. This can help in preventing Command Center from applying workload rules to short-running queries. See [Greenplum Command Center Parameters](#) for information on changing these parameters.

If Command Center cannot successfully apply any rule's action (for example, if an attempt to move a query to another resource group fails due to resource availability), then the action is retried 2 times, after waiting a minimum of 15 seconds between attempts. You can configure the wait interval by using the `wlm_query_cooldown_time` configuration parameter.

**Workload Management** Manage your workload through Resource Groups

**Rules**  
Create rules to better manage your incoming and ongoing queries. You can set up the conditions to trigger a rule and the actions the rule will take.

[+ CREATE A NEW RULE](#)

ID	Category	Conditions
1	Idle Sessions	If resource group = admin_group AND idle time is greater than 1m
2	Queries	If slice number is greater than 9
3	Queries	If spill file size is greater than 1GB

**Logs**  
All actions performed based on rules will be shown here.

Category	Actions
Idle Sessions	Terminate session 1621219453-3113
Idle Sessions	Terminate session 1621219453-2742
Idle Sessions	Terminate session 1621219453-3099
Idle Sessions	Terminate session 1621219453-2849
Idle Sessions	Terminate session 1621219453-3057

**Create Rule**

**Queries** **Idle Sessions**

Create rules to better manage your incoming and ongoing queries. You can set up the conditions to trigger a rule and the actions the rule will take.

**Step 1 Choose Identifiers**

Resource Groups

All resource groups

[+ Add a new option](#)

**Step 2 Choose Conditions**

Condition Selection ...

[+ Add a new option](#)

**Step 3 Choose Action**

Action Selection ...

[DISCARD](#) [CREATE](#)

1. To edit an existing query rule, click **EDIT** next to the rule definition to open the Workload Management Rules editor.
2. To delete a query rule, click **EDIT** next to an existing rule and then click **DELETE** in the



Workload Management Rules editor.

3. To create a new query rule, click **CREATE A NEW RULE** and fill in the fields in each section.

1. Choose one or more identifiers to match queries to the rule. By default, rules are applied to all resource groups. Click **Add a new option** to include additional identifiers to filter based on query tags and/or database role. Note that queries are matched to a rule only if all of the configured identifiers match.

**Resource Groups** Choose a resource group name from the list, or accept the default **All resource groups** to match the rule with any available resource group. Note that if you *remove* the **Resource Groups** identifier from your rule, Command Center still uses **All resource groups** as the default identifier for matching a query's resource groups. **Query Tags** Enter one or more query tags to match against the `gpcc.query_tags` parameter in the Greenplum Database session. A query tag is a user-defined `<name>=<value>` pair. Separate multiple query tags with semicolons. See [Defining and Setting Query Tags](#) for more information about query tags. **Role** Enter a role name in the field to match the rule with a role of the same name in the database session.

2. To create an query assignment rule, define no additional conditions for the rule (skip to the next step). To create a workload rule, choose one or more conditions that a matched, running query must meet before the selected action is performed. Click **Add a new option** to specify multiple conditions, all of which must apply before any action is taken.

**CPU Skew** - The threshold value for CPU skew and the maximum amount of time -- in seconds, minutes, or hours -- the query can run at that skew threshold value. **CPU Time** The maximum amount of CPU time that the query consumes, specified in seconds, minutes, or hours. **Planner Cost** The query planning cost assigned to the query. Specify two separate values in the fields. The first field, **ORCA cost**, indicates the maximum Tanzu Query Optimizer (GPORCA) planning cost for the query. The second field, **Planner cost**, indicates the maximum Postgres Planner cost for the query, used only if the query falls back to using the Postgres Planner instead of GPORCA. Note that the two optimizers use different cost models, as well as different algorithms, to determine the cost of a query execution plan. See [Query Profiling](#) in the Greenplum Documentation for more information. Both planner cost values are required, because you cannot predict which planner will be chosen for a particular query. **Query Running Time** The maximum total running time for the query, specified in seconds, minutes, or hours. **Slice Number** The maximum number of slices allocated by the query planner to execute the query in parallel on different segments. **Spill File Size** The maximum value of a query's spill file size, specified in MB, GB, or TB. **Total Disk I/O** The maximum total disk I/O that the query performs, specified in MB, GB, or TB. Note that Command Center can only make an approximate estimate of the actual I/O used by a query, so actual usage may occasionally exceed the configured value.

3. Choose an action to perform when a query is matched to the rule and all conditions apply. For query assignment rules (rules that have no additional conditions), choose



### Move to another resource group.

**Cancel query** Cancel the query that matches all of the configured conditions. You must specify at least one condition (creating a workload rule) to trigger this action.

**Move to another resource group** For a query assignment rule, this action defines the resource group where the query executes. If Command Center matches the rule based on the query tags and/or role identifiers for the rule, the configured source group is used to execute the query. For a workload rule that configures one or more conditions, this action moves the running query to the specified resource group if all of the configured conditions are met. Note that moving an active query as part of a workload rule is supported only when using Greenplum version 6.8 or later. You can create a workload rule with this action in earlier versions of Greenplum, but the rule will be created in the **Inactive** state. **Create a log entry** This action causes Command Center to log the query in the Workload Management logs.

4. Click **CREATE** to create the new rule, or **DISCARD** to dismiss the window without creating the rule.
4. Use the **Active/Inactive** toggle to make a rule active or inactive.

## Defining Workload Management Rules for Idle Sessions

You can create a workload rule that terminates an idle session associated with a particular resource group once the session has been idle for a specified number of seconds, minutes, or hours. If all conditions are met, the workload rule action is performed.

Workload Management Manage your workload through Resource Groups

### Create rules to automatically free up workload in the database

**Cancel Queries**  
You can create rules to abort long-running or resource-intensive queries to release more system capacity.

**Move Queries to Other Resource Groups**  
You can create rules based on resource consumptions, like CPU time, disk I/O, and others to move queries to other resource groups.

[+ CREATE A NEW RULE](#)

×

Create Rule

Queries

Idle Sessions

Create rules based on different conditions to manage idle sessions.

Step 1 Choose Identifiers

Resource Groups

Choose

Step 2 Choose Conditions

Idle Time

>  seconds

Step 3 Choose Action

Terminate Session

The default message is "WLM Rule Engine canceled the session because it has been idle for too long."

☐ Idle in Transaction sessions are excluded.

DISCARD CREATE

1. To edit an existing idle sessions rule, click **EDIT** next to the rule definition to open the Workload Management Rules editor.
2. To delete an idle sessions rule, click **EDIT** next to an existing rule and then click **DELETE** in the Workload Management Rules editor.

3. To create a new idle sessions rule, click **CREATE A NEW RULE** and fill in the fields in each section.
  1. Choose a resource group name from the list.
  2. Choose an idle time for the session -- in seconds, minutes, or hours.
  3. In the box under Terminate Session, enter a message to print to the log file and to the console where the session is running. If you do not enter anything, the default message "WLM Rule Engine canceled the session because it has been idle for too long" will be printed.
  4. To exclude Idle in Transaction sessions from the new rule, click the checkbox labeled "Idle in Transaction sessions are excluded."
  5. Click **CREATE** to create the new rule, or **DISCARD** to dismiss the window without creating the rule.

## Defining and Setting Query Tags

A query tag is a user-defined `<name>=<value>` pair, set in the Greenplum Database `gpcc.query_tags` parameter in the Greenplum Database session. The `gpcc.query_tags` parameter is defined when the `gp_wlm` database extension is enabled in the postgres database. If you try to set query tags when the `gp_wlm` extension is not enabled, you get an unrecognized configuration parameter error. To see if the extension is enabled, run the following command.

```
$ psql postgres -c "\dx"
                                List of installed extensions
  Name  | Version | Schema | Description
-----+-----+-----+-----
 gp_wlm | 0.1     | gpcc   | Greenplum Workload Manager Extension
(1 row)
```

When you submit a transaction and the `gp_wlm` extension is enabled, Greenplum Database calls the `gp_wlm` extension to determine the resource group for the transaction. The extension evaluates the current role and query tags set in the session against the rules you have defined in Command Center. If there is a match, the extension returns the rule's resource group. If there is no match, Greenplum Database assigns the transaction to the role's default resource group.

The following command, executed in the Greenplum Database session, sets the `appName` and `appUser` query tags to "tableau" and "bi\_sales", respectively.

```
=# SET gpcc.query_tags TO 'appName=tableau;appUser=bi_sales';
```

To match a rule, all tags in the rule's query tag field must be present in the `gpcc.query_tags` parameter in the database session. The order of the tags is not significant, and the `gpcc.query_tags` parameter can have a superset of the tags defined in the `queryTags` value.

When query tags are surrounded by single quotes and those tags have been set by the `PGOPTIONS` parameter, the single quotes become part of the query tags. In this case, Command Center ignores the single quotes, rather than processing them as part of the query tag name -- which would cause resource group assignment rules to fail.

If you set the `gpcc.query_tags` parameter inside of a transaction, you must commit the transaction before the new query tags are used to evaluate assignment rules.

You can set the value of the `gpcc.query_tags` parameter using the `SET` command, as in the example above, or as a connection parameter with database clients that support it, such as `psql`. Following are two examples that show how to specify query tags on the `psql` command line.

```
$ PGOPTIONS="-c gpcc.query_tags=appName=tableau;appUser=bi_sales" psql

$ psql postgresql://mdw:5432/postgres?options="-c gpcc.query_tags%3DappName%3Dtableau;appUser%3Dbi_sales"
```

In the second example, it is necessary to code the equals signs as `%3D` to prevent `psql` from interpreting the query tags as command-line arguments.



**Note:** Setting query tags in a `psql -c` command does not work because resource group assignment occurs before the command specified with the `-c` option is executed. For example, this command will not have the desired effect.

```
psql -c "SET gpcc.query_tags TO 'appName=tableau;appUser=bi_sales'; SELE
CT * FROM sales_data;"
```

## Monitoring Workload Rules

The **Logs** section displays a row of information for each instance where a query or idle session both matched a workload rule and triggers the workload rule conditions to perform an action. Note that query assignment rules are not currently logged in this section.

Columns of the log entry provide details about how the rule was applied:

**Actions** Summarizes the rule actions that were taken for the specified query ID. **Resource Group** The resource group that matched the query or idle session rule. **Role** The database role of the query session. **Rule** The ID number of the rule. This ID matches an ID from the list of **Rules** on this page. Place your cursor over the ID to display a summary of the rule conditions that were in effect when the action was triggered. **Status** Indicates whether the rule's action succeeded or failed. **Execution Time** The time when the rule action was performed.

## Importing Resource Queues to Resource Groups

Greenplum Command Center workload management works with resource groups, the new Greenplum Database resource management system. The default resource management system for Greenplum Database is resource queues. To use the Command Center workload management features, you must first enable resource groups in Greenplum Database.

Command Center can assist you in enabling resource groups and in importing existing resource queues to resource groups.

## Step One: Enable Resource Groups in Greenplum Database



## Workload Management Manage your workload through Resource Groups

Here is a preview of your resource groups converted from your resource queues. Please input the resource allocations. Your roles will be matched with the assigned resource groups.

Resource Group	Concurrency	CPU %	Memory %	Min mem per query
default_group	<input type="text" value="20"/>	<input type="text" value="30"/>	<input type="text" value="30"/>	86.16MB
admin_group	<input type="text" value="20"/>	<input type="text" value="30"/>	<input type="text" value="30"/>	86.16MB
vip	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
etl	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
adhoc	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
bi_analytics	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
		60	60	Total CPU and Memory must be less than or equal to 100%

[IMPORT RESOURCE GROUPS](#)
[SKIP IMPORT](#)

Your roles will be imported to the matching resource groups. [Close preview.](#)

default_group (0)	admin_group (2)	vip (3)	etl (2)	adhoc (0)	bi_analytics (4)
	gpmon gpadmin	nickd ralphs katrinab	kristiem richd		jillianr brentd sallyg anny

The resource group list includes the required `admin_group` and `default_group` resource groups, and a row for each of your existing resource queues.

Roles are assigned to the resource group matching the resource queue to which they are assigned. Click the **Preview roles** link to see the role assignments.

Your roles will be imported to the matching resource groups. [Close preview.](#)

default_group (0)	admin_group (2)	vip (3)	etl (2)	adhoc (0)	bi_analytics (4)
	gpmon gpadmin	nickd ralphs katrinab	kristiem richd		jillianr brentd sallyg anny

If you want to set up resource groups later, you can click **SKIP IMPORT**. Only the `default_group` and `admin_group` resource groups are created. Roles with the superuser attribute are assigned to the `admin_group` resource group; roles without superuser privilege are assigned to the `default_group` resource group.

If you want Command Center to import resource queues to resource groups, you must complete the resource allocation fields for all resource groups.

Set the **Concurrency**, **CPU %**, and **Memory %** resource group attributes to allocate Greenplum Database resources to the resource queues. The **Concurrency** fields must each contain a positive integer. The **CPU %** and **Memory %** fields must each contain positive integers between 1 and 99 and the totals for the **CPU %** and **Memory %** columns must not exceed 100%. See [Defining Resource Groups and Resource Attributes](#) for help determining the values to enter.

The **IMPORT RESOURCE GROUPS** button is disabled until you have entered valid values in the allocation fields for every resource group.

When you are ready to import the resource groups, click **IMPORT RESOURCE GROUPS** to create the resource groups.

## Step Three: Enable Command Center Workload Management

After you import (or skip importing) resource queues to resource groups, you can enable Command Center workload management.

Resource Groups

EDIT

Name	Concurrency	CPU %	Memory %	Min mem per query
default_group	5	2	5	86.16MB
admin_group	5	2	5	57.44MB
vip	10	10	15	137.86MB
etl	5	5	5	91.90MB
adhoc	5	2	3	55.14MB
bi_analytics	50	30	30	55.14MB

Assignment by Role

Queries are routed to Resource Group based on gpdb role, unless diverted by Query Tag filter (below)

default\_group (0)

admin\_group (2)

vip (3)

etl (2)

adhoc (0)

bi\_analytics (4)

add role

add role

add role

add role

add role

add role

Workload Management Rules

EDIT

Queries matching a filter's query tag connection attributes will be diverted, overriding its role's assigned Resource Group. Drag to change filter order.

If matching query tags

Assign to Resource Group

no assignment rules

ENABLE WORKLOAD MANAGEMENT

Automatically terminate connections when idle for prescribed period of time

Enable Workload Management will allow you set query tags and idle session rules.

Resource Group

Time before idle connections killed

Exempted roles

Message

no timeout rules

Click **ENABLE WORKLOAD MANAGEMENT** to enable workload management in Greenplum Command Center. Greenplum Command Center creates the `gp_wlm` extension, the `gpcc.workload_config` table, and the associated user-defined functions in Greenplum Database.

You are now able to use the Command Center Workload Management interface to add, remove, and configure resource groups; change role assignments; and define workload management rules. See [Workload Management](#) for help using the Command Center Workload Management view.

## Troubleshooting Enabling Resource Groups

If you experience problems enabling resource groups in Greenplum Command Center, review the following list to ensure prerequisites are met and all of the dependencies are properly configured.

- Red Hat 6.x and 7.x and CentOS 6.x and 7.x are currently supported.
- You must be running Greenplum Database version 5.7.0 or later.
- Configure the Linux cgroups kernel feature on your hosts by following the instructions at "Prerequisite" in [Using Resource Groups](#).

- Make sure the `/etc/cgconfig.d/gpdb.conf` file contains the objects `perm`, `cpu`, and `cpuacct`. If the document is incorrect and the `gp_resource_manager` configuration parameter is set to `"group"`, Greenplum Database can hang at startup.

```
group gpdb {
  perm {
    task {
      uid = gpadmin;
      gid = gpadmin;
    }
    admin {
      uid = gpadmin;
      gid = gpadmin;
    }
  }
  cpu {
  }
  cpuacct {
  }
}
```

- On Red Hat 7, make sure you run `cgconfigparser -L /etc/cgconfig.d` to parse changes to the `/etc/cgconfig.d/gpdb.conf` file. This command must also be set up to run at boot time.
- Set the Greenplum Database `gp_resource_manager` server configuration parameter to `"group"` and restart Greenplum Database.

```
$ gpconfig -c gp_resource_manager -v "group"
$ gpstop -ar
```

Verify by showing the value of the parameter:

```
$ gpconfig -s gp_resource_manager
Values on all segments are consistent
GUC          : gp_resource_manager
Master value: group
Segment value: group
```

- After installing a Tanzu Greenplum Database distribution, the `shared_preload_libraries` configuration parameter contains the metrics collector shared library. Make sure this library is still present:

```
$ gpconfig -s shared_preload_libraries
Values on all segments are consistent
GUC          : shared_preload_libraries
Master value: metrics_collector
Segment value: metrics_collector
```

Check that the shared library exists at `$GPHOME/lib/postgresql/metrics_collector.so`. If the library does not exist, make sure you have installed the Tanzu Greenplum Database distribution. This extension is not available in the Greenplum Database Open Source version.

If the shared library file exists in the `$GPHOME/lib/postgresql` directory, but not in the `shared_preload_libraries` parameter, add it with the `gpconfig` command:

```
$ gpconfig -c shared_preload_libraries -v 'metrics_collector'
```

Note that adding `metrics_collector` to the `shared_preload_libraries` parameter does not enable the `metrics_collector` or `gp_wlm` extensions, but is a prerequisite for enabling them.

- The `gpmon` user must be able to connect to databases from the Command Center host. Make sure to add a `host` entry like the following in the `$MASTER_DATA_DIRECTORY/pg_hba.conf` file.

```
host    all    gpmon    <IP_of_host>/32    md5
```

- Check whether the `gp_wlm` extension is installed. The extension is added when you click **Enable Workloads** in the Greenplum Command Center **Workload Mgmt** view.

```
$ psql gpperfmon
\dx
gpperfmon=# \dx
```

List of installed extensions			
Name	Version	Schema	Description
gp_wlm	0.1	gpmetrics	Greenplum Workload Manager Extension
metrics_collector	1.0	gpmetrics	Greenplum Metrics Collector Extension
plpgsql	1.0	pg_catalog	PL/pgSQL procedural language

(3 rows)

- Make sure the `gpmetrics.workload_config` table and functions are present in the `gpperfmon` database:

```
gpperfmon=# \d gpmetrics.workload_config
Table "gpmetrics.workload_config"
  Column | Type | Modifiers
-----+-----+-----
 dist_col | integer |
 config | json |
Distributed by: (dist_col)

gpperfmon=# \df gpmetrics.*
```

List of functions			
Schema	Name	Result data type	Argument data types
gpmetrics	get_workload_config	json	
gpmetrics	metrics_collector_start_worker	void	
gpmetrics	set_workload_config	boolean	wlm_json_config

(3 rows)

If the `gpmetrics.workload_config` table or the functions are not present, dropping and recreating the `gp_wlm` extension will create them. Note that any assignment rules saved in



the `gpmetrics.workload_config` table will have to be recreated in Command Center.

```
$ psql gpperfmon
gpperfmon=# DROP EXTENSION gp_wlm;
DROP EXTENSION
gpperfmon=# CREATE EXTENSION gp_wlm;
CREATE EXTENSION
```

# Greenplum Database Table Browser

Command Center users with Basic permission can view details about the tables in a Greenplum database.

1. Select **Table Browser** to view the Command Center Table Browser.
2. On the **Table Browser** view, choose a database and then use the dropdown to select a schema. All filter criteria except **Database** are optional.

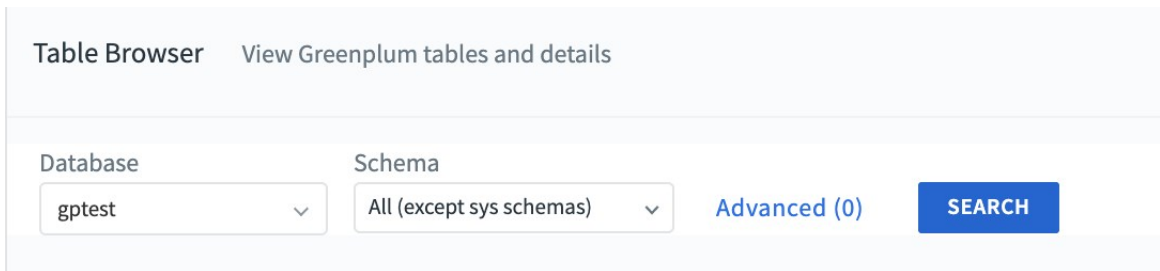


Table Browser View Greenplum tables and details

Database Schema

gptest All (except sys schemas) Advanced (0) SEARCH

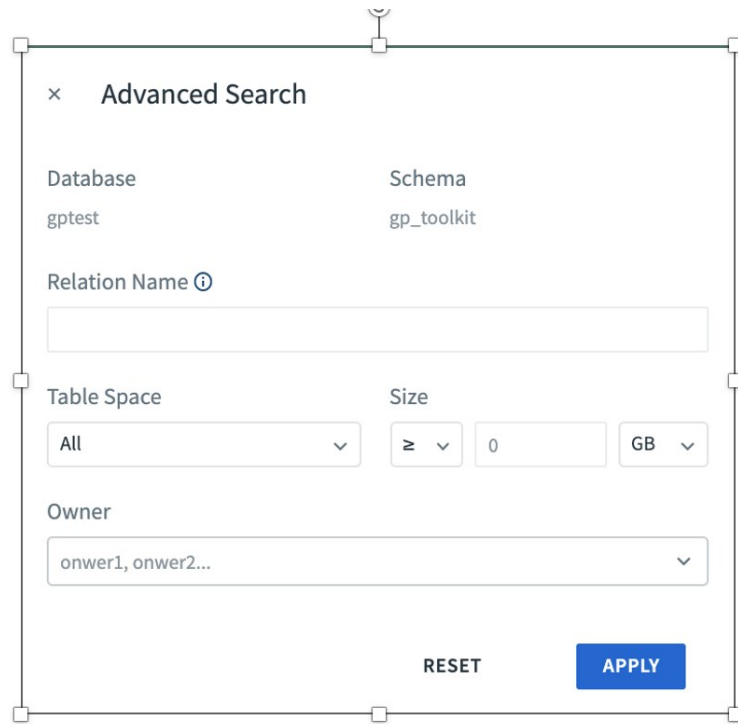
## Database

Select the name of the database to browse.

## Schema

Enter the name of a schema to select only the tables that belong to that schema. The default is "All (except sys schemas)" where system schemas include: `pg_catalog`, `pg_toast`, `pg_bitmapindex`, `pg_aseg`, `information_schema`, or `gp_toolkit`.

3. Click **Advanced** to display more search criteria for the database and the schema you have selected.



**Advanced Search**

Database: gptest      Schema: gp\_toolkit

Relation Name ⓘ

Table Space: All      Size: ≥ 0 GB

Owner: onwer1, onwer2...

RESET      **APPLY**

### Relation Name

Enter the table name for your search. You may enter a % wildcard to match any characters.

### Table Space

Select the table space from the dropdown options.

### Table Size

To select tables by size, select a ≤ (less than or equal) or ≥ (greater than or equal) operator, and enter a size number with a size unit (KB, MB, GB, TB).

### Owner

Enter a role name to select only tables owned by the role.

- Click **SEARCH** to display tables matching the criteria you have entered.

Table Browser    View Greenplum tables and details    Current Time  
2021-02-09 12:22:32

Database: gptest    Schema: All (except sys schemas)    Advanced (1)    **SEARCH**

23 Tables found in gptest    1    **EXPORT ALL**

Schema	Relation Name	Partitions	Size	Owner	Est. Rows	Last Analyzed	Last Vacuumed	Last Accessed	Storage
public	a	--	3.94 MB	gpadmin	111111	2019-01-16 17:39:49	2020-03-21 19:28:15	--	Heap
tpcds	catalog_page	--	2.11 MB	gpadmin	11718	--	2020-03-21 19:28:18	--	AO/CO
tpcds	catalog_returns	275	91.13 MB	gpadmin	0	--	2020-03-21 19:28:35	--	AO/CO
tpcds	catalog_sales	80	25.31 MB	gpadmin	0	--	2020-03-21 19:28:39	--	AO/CO
public	columns1600	--	625.41 MB	gpadmin	100000	2019-12-18 17:06:02	2020-03-21 19:28:59	--	Heap
bm	het_bm	6	8.91 MB	gpadmin	20	--	2020-03-21 19:28:17	--	Poly
tpcds	inventory	23	7.50 MB	gpadmin	0	--	2020-03-21 19:28:40	--	AO/CO

Table Browser statistics are updated every 300 seconds by default. Activity that occurs between updates is not displayed in the Table Browser view until Command Center updates the table statistics at the end of the sampling interval. This means that, for example, it can take up to five minutes for a new table to appear in the Table Browser. The times displayed for **Last Analyzed**, **Last Vacuumed**, and **Last Accessed** are the time that Command Center

updates the table statistics after the event has been detected, not the actual time of the event. The actual event could have occurred at any point in the previous five minutes (or in the time period specified by `stats_check_interval`, if you have modified the parameter).

You can change the sampling interval by setting the `stats_check_interval` parameter in the `$GPCC_HOME/conf/app.conf` configuration file. Restart Command Center after updating the configuration file. See [Command Center Console Parameters](#) for more information about the `app.conf` file.

**Note:** using the `gppersistentrebuild` utility might refresh the `Last Accessed` table data even if the table has not been accessed by a query.

- Click the **EXPORT ALL** button, at the top right of the table results, to save the search results in a local file. The file is created in the background and when the export is ready, a notification is entered at the Notifications center, with a link to a CSV file, similar to:

• 2021-02-09 14:28:57

**[Notice]** Your table export has been generated. You can download from here.  
[tables\\_20210209142856\\_RAKCVIAH.csv](#)

If the export operation is greater than 1 million rows, an alert is raised.

You can change the sampling interval by setting the `stats_check_interval` property in the `$HOME/gpmetrics/gpcc.conf` configuration file. Restart Command Center after updating the configuration file. See the [gpmetrics Configuration File Reference](#) for more information about the `gpcc.conf` file.

## Table List

The **Table Browser** table list displays these columns:

### Schema

Name of the schema the table belongs to.

### Relation Name

Name of the table.

### Partitions

Total number of leaf partitions. If the table has no partitions, this column contains 0.

### Size

Combined size of the table's files on all segments.

### Owner

Database role name of the table owner.

### Est. Rows

Estimated number of rows in the table. For external tables this column contains 0.

### Last Analyzed

Time the table was last analyzed. The actual event occurred at some point during the preceding

sampling interval.

### Last Vacuumed

Time the table was last vacuumed. (Does not include `AUTOVACUUM`.) The actual event occurred at some point during the preceding sampling interval.

### Last Accessed

Time the table was last accessed. (`SELECT`, `INSERT`, `UPDATE`, `DELETE`, or `TRUNCATE` operations.) The actual event occurred at some point during the preceding sampling interval.

### Storage

Storage type for the table: Heap, AO, AO/CO, External, or Poly (Polymorphic - used for tables that combine different storage types in a single table).

## Table Details

Click the name of a table in the **Relation Name** column to view details for that table.

Table Browser

>

customer\_demographics

Current Time

2022-07-14 15:17:13

Details

Database

gadmin

Owner

gadmin

Storage

External

Schema

ext\_tpcds

Distributed By

--

Compression Type

--

Table Space

--

Columns

9

Compression Level

--

Size

0

Partitions

--

Block Size

0

Usage

Last Vacuumed

--

Last Analyzed

--

Last Accessed

--

Est. Rows

1,000,000

Recent Queries

No matching results found

Table Content

DDL

Column	Type	Modifiers	Storage	Comp Type	Comp Level	Block Size	Description
cd_demo_sk	integer	--	plain	--	--	--	--
cd_gender	character(1)	--	extended	--	--	--	--
cd_marital_status	character(1)	--	extended	--	--	--	--
cd_education_status	character varying(20)	--	extended	--	--	--	--
cd_purchase_estimate	integer	--	plain	--	--	--	--

The **Table Detail** view displays this information:

### Name

Name of the table.

### Database

Name of the database.

### Schema

Name of the schema the table belongs to.

### Table Space

Name of the tablespace where the table's files reside.

**Size**

The size of the table.

**Owner**

Database role name of the table owner.

**Distributed By**

Distribution policy for the table. This can be a list of the distribution key columns (`key1, key2, key3, ...`), `Randomly`, or `Replicated`.

**Columns**

Number of user columns in the table. System columns are not included in the count.

**Partitions**

Number of partitions the table has, including middle-level and leaf partitions. Click `Show` to list up to 100 partitions. The list includes the partition's name, table size, partition criteria, storage type, and compression parameters. Middle-level partitions are displayed with a different background color.

**Storage**

Storage type for the table: Heap, AO, AO/CO, External, or Poly (Polymorphic - used for tables that combine different storage types in a single table).

**Compression Type**

For append-optimized tables, the type of compression used for the table: `ZLIB`, `ZSTD`, `QUICKLZ`, `RLE_TYPE`.

**Compression Level**

For append-optimized tables, the level of compression. For `ZLIB`, the level is 1 (fastest compression) to 9 (highest compression ratio). For `ZSTD`, the level is 1 (fastest compression) to 19 (highest compression ratio). For `QUICKLZ`, the level is 1. For `RLE_TYPE` the compression level is 1 (fastest compression) to 4 (highest compression ratio).

**Blocksize**

Size, in bytes, for each block in the table: 8k, 16K, 1M, or 2M.

**Usage Section****Last Analyzed**

Time the table was last analyzed.

**Last Vacuumed**

Time the table was last vacuumed. (Does not include `AUTOVACUUM`.)

**Last Accessed**

Time the table was last accessed. (`SELECT`, `INSERT`, `UPDATE`, `DELETE`, or `TRUNCATE` operations.)

**Est. Rows**

Estimated number of rows in the table.

## Recent Queries Section

This section lists the five most recent queries in the query history to access the table. The list is limited to queries that executed in the last 30 days and that ran for ten seconds or more.

Click a Query ID to view the [Query History Details](#) for the query.

## Table Contents Tab

A list of the table's columns, types, constraints, and compression types.

### Column

Name of the column.

### Type

Data type of the column.

### Modifiers

Constraints defined for the column.

### Storage

Storage type for the column. `Plain` is for fixed-length values such as `integer` and is inline, uncompressed. `Main` is for inline, compressible data. `Toast` is for data that exceeds the size of a data block and is stored in chunks in a separate table associated with the user table. `Extended` is for external, compressed data. `Extended` is the default for most data types that support non-plain storage.

### Compression Type

The type of compression used for the column: `none`, `ZLIB`, `ZSTD`, `QUICKLZ`, `RLE_TYPE`.

### Compression Level

The level of compression.

For `ZLIB`, the level is 1 (fastest compression) to 9 (highest compression ratio).

For `QUICKLZ`, the level is 1.

For `ZSTD`, the level is 1 (fastest compression) to 19 (highest compression ratio).

For `RLE_TYPE` the compression level is 1 (fastest compression) to 4 (highest compression ratio).

### Blocksize

Size, in bytes, for each storage block.

## DDL Tab

Click the DDL tab to display the `CREATE TABLE` statement for the table.

```

CREATE TABLE tpcds.item (
  i_item_sk integer NOT NULL ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_item_id character varying(16) NOT NULL ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_rec_start_date date ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_rec_end_date date ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_item_desc character varying(200) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_current_price numeric(7,2) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_wholesale_cost numeric(7,2) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_brand_id integer ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_brand character varying(50) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_class_id integer ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_class character varying(50) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_category_id integer ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_category character varying(50) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_manufact_id integer ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_manufact character varying(50) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_size character varying(20) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),
  i_formulation character varying(20) ENCODING (compressstype=none,blocksize=32768,compresslevel=0),

```



# Greenplum Command Center Plugins

Tanzu Greenplum Command Center supports plugins for monitoring other Greenplum services. As of this release, the only plugin supported is one that monitors Greenplum Text clusters.

To include the **Plugins** view in Greenplum Command Center, you must [install the plugin extension in Greenplum Database](#).

To access the **Plugins** view in Greenplum Command Center, click **Plugins** at the bottom of the left navigation bar.

GPText

Current Time

2022-08-09 23:15:42

Help

Overview

Cluster Name

GPDB GPText

Status

yellow

Solr Version

7.4.0

Uptime

up 13 hours

Indices

Num Indices

2

Num Shards

4

Num Docs

3

Indices Size

0.01 MB

Nodes

Num Nodes

3

Heap Used

1.23 GB

Heap Max

5.75 GB

Non-heap

241.49 MB

Indices

Nodes

Index Name ^	Status	Replication Factor	Index Size	Documents Count	Search Rate In 1 Min	Index Rate In 1 Min
demo.public.articles2	green	2	0.01 MB	3	0.00	0.00
demo.wikipedia.articles	yellow	2	N/A	N/A	0.00	N/A

For more information, see [Monitoring Greenplum Text](#).

# Query Monitor Help Topics

- [CPU](#)
- [CPU Skew](#)
- [Locks](#)
- [Query Optimization](#)
- [Memory](#)
- [Spill Files](#)

## CPU

The **CPU percent** metric is the average current CPU percentage for all backend processes executing this query. The percentages for all processes running a query on each segment are averaged, and then the average of all those values is calculated to render this metric.

You can manage the percentage of CPU that queries can consume by creating workloads and specifying the maximum percent of CPU each workload can consume. That percentage is further divided among the segments running on each host and then among the concurrent queries the workload can execute.

CPU allocated to idle workloads is reallocated to active queries and reclaimed when the idle workload becomes active again. This means that the **CPU percent** value for a query can exceed limits defined for workloads and can increase and decrease as other queries start or finish.

Memory and disk I/O resources are more likely causes for degraded query performance than lack of CPU cycles. The ways to reduce CPU contention mirror the solutions for insufficient memory:

- Reduce concurrency of workloads to make more CPU available to each query.
- Reduce the number of workloads and reallocate CPU to the remaining workloads.

If CPU is not constrained and the size of spill files for some queries is very large, make sure that the `gp_workfile_compression` server configuration parameter is set to `on`. Compressing spill files reduces disk I/O, but uses CPU cycles to compress and decompress the data.

See [Using Resource Groups](#) for more about managing performance with resource groups.

If your Greenplum Database system is configured to manage resources with resource queues, see [Using Resource Queues](#).

## CPU Skew

### What is CPU Skew?

CPU skew occurs when the work to execute a query is not distributed evenly among the segments.

The **CPU** metric is the average of the CPU percentages used by each process executing the query. The **CPU skew** metric is a variance statistic based on the difference between the average and each segment's current **CPU** metric. The smaller the **CPU skew**, the more equally the work is distributed. The **CPU skew** metric varies between 0.0 (no skew) and 1.0. The lower the skew metric the more fully the database cluster's resources are utilized.

CPU skew is usually related to the volume of data processed by the segments while executing the query execution plan. There are two types of skew you should investigate: data skew and computational skew.

## Data Skew

A high CPU skew may be an indication of data skew, where tables used by the query are distributed unevenly, so that some segments have more data to process than their peers. You can check for data skew in a table by running a query like this one:

```
=# SELECT gp_segment_id, COUNT(*) FROM <table-name> GROUP BY gp_segment_id;
```

The row count should be approximately equal for each segment. If the rows are distributed unevenly, check the distribution key for the table. A good distribution key is a column or list of columns with unique or nearly unique values, such as the table's primary key. Setting the distribution policy to **DISTRIBUTED RANDOMLY** also ensures a well-distributed table, but precludes taking advantage of performance-enhancing strategies such as co-location for tables with equivalent primary keys.

## Computational Skew

High CPU skew can be the result of computational skew, which occurs during query execution. Some of the operations in the query plan can cause some segments to do more work than others. For example, joins, sorts, or aggregations on columns with low cardinality or unevenly distributed values can contribute to CPU skew by causing some segments to process many more tuples than others.

See [Distribution and Skew](#) in the *Greenplum Database Administrator Guide* and [Tuning SQL Queries](#) in the *Greenplum Database Best Practices* guide for more help finding the causes of skew.

## Locks

Greenplum Command Center displays the locks currently held by queries and queries blocked by locks.

A block occurs when one query needs to acquire a lock that conflicts with a lock held by another query. If a query is blocked for a long period of time, you can investigate the blocking query and, if necessary, cancel one of the queries.

Locks can be acquired using the **LOCK TABLE** SQL statement. Some SQL commands acquire locks automatically. Following are descriptions of the lock modes, the Greenplum Database commands that acquire them, and which lock modes conflict with them.

ACCESS SHARE

Acquired by `SELECT` and `ANALYZE` commands.

Conflicts with ACCESS EXCLUSIVE locks.

In general, any query that only reads a table and does not modify it acquires this lock mode.

## ROW SHARE

Acquired by `SELECT FOR SHARE` command.

Conflicts with EXCLUSIVE and ACCESS EXCLUSIVE locks.

A ROW SHARE lock is placed on the specified table and an ACCESS SHARE lock on any other tables referenced in the query.

## ROW EXCLUSIVE

Acquired by `INSERT` and `COPY` commands.

Conflicts with SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE locks.

A ROW EXCLUSIVE lock is placed on the specified table and ACCESS SHARE locks are placed on any other referenced tables.

## SHARE UPDATE EXCLUSIVE

Acquired by `VACUUM` and `VACUUM FULL`.

Conflicts with the SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE locks.

SHARE UPDATE EXCLUSIVE protects a table against concurrent schema changes and `VACUUM` runs.

## SHARE

Acquired by `CREATE INDEX`.

Conflicts with ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE locks.

Protects a table against concurrent data changes.

## SHARE ROW EXCLUSIVE

This lock mode is not automatically acquired by any Greenplum Database command.

Conflicts with ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE locks.

## EXCLUSIVE

Acquired by `UPDATE`, `SELECT FOR UPDATE`, and `DELETE` commands in Greenplum Database.

Conflicts with ROW SHARE, ROW EXCLUSIVE, SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE locks. This lock mode allows only concurrent ACCESS SHARE locks - a table can be read by another transaction while this lock is held. This is more restrictive locking than in regular PostgreSQL.

## ACCESS EXCLUSIVE

Acquired by the `ALTER TABLE`, `DROP TABLE`, `TRUNCATE`, `REINDEX`, `CLUSTER`, and `VACUUM FULL` commands. Default lock mode for `LOCK TABLE` statements that do not specify a lock mode. Also briefly acquired by `VACUUM` (without `FULL`) on append-optimized tables during processing.

Conflicts with all locks.

This lock mode guarantees that the holder is the only transaction accessing the table in any way.

For more on locks in Greenplum Database queries, see the [LOCK](#) command Reference. See also [Tuning SQL Queries](#).

## Memory

The Greenplum Command Center Query Monitor reports the current total memory consumed by all processes executing a query. When there is insufficient memory available for a query to complete, the query has an error status in the query monitor and an out of memory error is logged.

If you have enabled resource groups in Greenplum Database, you can manage the amount of memory available to queries by tuning resource group parameters, and by setting Greenplum Database configuration parameters that affect resource group memory.

- For a detailed description of resource group memory management, see [Using Resource Groups](#) in the *Greenplum Database Administrator Guide*.
- If you are using resource queues, see [Memory and Resource Management with Resource Queues](#) and [Using Resource Queues](#) for ways to troubleshoot memory problems with resource queues.
- See [Tuning SQL Queries](#) for help with query optimization.

The following summary describes the resource group parameters and related Greenplum Database server configuration parameters that determine the amount of memory available to database queries and how configuration choices affect concurrency, spill file usage, and query performance.

## Resource Group Memory Configuration Parameters

A resource group has parameters `CONCURRENCY`, `MEMORY_LIMIT`, `MEMORY_SHARED_QUOTA`, and `MEMORY_SPILL_RATIO`, which determine how much memory is allocated to execute a query. The `CPU_LIMIT` parameter has no effect on memory allocation. See the `CREATE RESOURCE GROUP` SQL reference for command syntax and information about these parameters.

### `MEMORY_LIMIT`

This parameter sets the amount of memory the resource group manages as a percentage of the memory available to resource groups. The sum of all resource groups' `MEMORY_LIMITs` must not exceed 100. If the sum of all resource groups' `MEMORY_LIMITs` is less than 100, the remaining, unallocated memory is *global resource group shared memory*, available to queries from all resource groups on a first-come, first-served basis.

### `MEMORY_SHARED_QUOTA`

A resource group divides the memory it manages into a fixed portion and a shared portion, called *resource group shared memory*. This parameter specifies the percentage of a resource group's memory that is shared. The default is 20 and the value can range from 0 to 100.

### `CONCURRENCY`

This parameter limits the number of concurrent transactions a resource group allows. The fixed portion of the memory the resource group manages is divided equally among `CONCURRENCY`

transaction slots. Every transaction starts with this fixed portion of memory and, if needed, Greenplum Database uses additional memory from the resource group shared memory and global resource group shared memory.

#### `MEMORY_SPILL_RATIO`

This parameter sets a limit for the amount of memory a query can use before it spills to disk. The parameter value is expressed as a percentage of the fixed memory allocation. The default is 20 and the value can range from 0 to 100. A higher value uses more memory, but can improve query performance. A transaction can override this value by setting the `memory_spill_ratio` configuration parameter in the session.

When a query executes, Greenplum Database allocates memory to it from the fixed portion of the resource group's memory. If the query needs more memory and the resource group has available shared memory, Greenplum Database allocates additional memory. If insufficient shared memory is available, Greenplum Database allocates additional memory from global shared memory, if available. If the required memory is not available the transaction fails.

## Greenplum Database Memory Configuration Parameters

The following Greenplum Database configuration parameters affect resource group memory allocation and concurrency.

#### `gp_resource_group_memory_limit`

This Greenplum Database server configuration parameter sets the percentage of each host's system memory to be managed by resource groups. The default is `0.7` (70%). This memory is divided equally among the primary segments on each host, and further divided among resource groups with the `MEMORY_LIMIT` resource group parameter. Any memory not allocated to resource groups becomes global shared memory available to queries from all resource groups. See `gp_resource_group_memory_limit` for a complete reference for this parameter.

#### `gp_resgroup_memory_policy`

This parameter determines how Greenplum Database allocates memory to query operators. The default value, `eager_free`, re-allocates memory from completed operators to operators later in the query plan. The alternative value for this parameter, `auto`, allocates a fixed amount of memory to operators that are not memory-intensive and the rest to the memory-intensive operators. The default value is usually the optimal setting. See `gp_resgroup_memory_policy` for a complete reference for this parameter.

#### `memory_spill_ratio`

A transaction can override the resource group's `MEMORY_SPILL_RATIO` value by setting the `memory_spill_ratio` configuration parameter in the session. The value is a percentage of the fixed memory allocation for transactions in the resource group, expressed as an integer from 0 to 100. The performance of queries with low memory requirements can be improved by setting the `memory_spill_ratio` parameter in the session to a low setting, for example 0 to 2. See `memory_spill_ratio` for more information about this parameter.

## Query Plan Execution

The Greenplum Database legacy and GPORCA query optimizers generate execution plans that produce the results requested by the query. A plan is a sequence of operators, such as table scans, joins, sorts, aggregates, and data motions.

When you select a query on the Command Center **Query Monitor** view, a **Query Details** view presents a graphical representation of the execution plan.

You can switch between the graphical and textual representations of the query execution plan by selecting the **Plan & Progress** tab or the **Textual Plan** tab. In the textual format, each plan node is flagged with an arrow (->). In the graphical view, the nodes are represented by boxes that fill as the plan executes.

A query execution plan executes from the bottom up. Each node in the plan performs an operation and passes results up to the next node in the plan.

The **Optimizer status:** line on the **Textual Plan** tab reports whether the explain plan was generated using the GPORCA optimizer or the legacy query optimizer.

## Slices and Gangs

Segments can work on portions of a query in parallel, each segment executing operators independently on their local data. When the plan requires exchanging data between segments, a data motion operator coordinates the data transfer between segments. The plan is divided into "slices" where these data motions occur.

A data motion node in a textual query plan identifies the slice and the number of segments participating in the motion.

Example:

```
-> Broadcast Motion 4:4 (slice2; segments: 4) (cost=0.00..867.15 rows=10000 width=30)
```

In a broadcast motion, each segment broadcasts all of its rows for a table over the network so that every segment has a complete copy of the table. In this example, the broadcast motion marks the completion of **slice2** with four segments sending and four segments receiving.

Each segment has one or more backend processes working on a slice. Backend processes working on the same slice are called a "gang".

## Operators

Operators are processes that take as input database tables or the output from other operators, and perform some action to produce a transformed output.

### Scan Operators

Init plan

A query that runs before the main query is optimized to find the partitions to scan.

Sequential scan

The optimizer may choose a sequential table scan if there is no index on the condition column or if most rows are expected to satisfy the condition. Because each segment scans an equal portion of

the data in parallel with other segments, a table scan in Greenplum Database is very efficient. A query on a partitioned table may be able to eliminate partitions to make the scan even faster.

#### Append-only scan

Scans rows in a row-oriented, append-optimized table.

#### Append-only columnar scan

Scans rows in a column-oriented, append-optimized table.

#### Dynamic table scan

Scans selected partitions in a partitioned table.

#### Function scan

A Function Scan node selects the partitions to scan. The function can be one of the following:

- `gp_partition_expansion` - chooses all nodes
- `gp_partition_selection` - chooses a partition with an equality expression
- `gp_partition_inversion` - chooses partitions with a range expression

#### Index scan

Scans a B-tree index on a table to find rows. The rows are then retrieved from disk.

#### Bitmap index scan

A Bitmap Index Scan is an index scan optimized by storing rows in a bitmap instead of retrieving them from the table immediately. When the scan is complete, rows in the bitmap are retrieved with a Bitmap Heap Scan operation.

#### BitmapAnd and BitmapOr

Generates a new bitmap by running logical AND or OR on multiple bitmaps.

#### Bitmap heap scan

Retrieves rows from heap storage using a bitmap generated by a Bitmap index scan or BitmapAnd or BitmapOr operation.

#### Nested loop with inner sequential scan join

For each row in the first table, the operator tests every row in the second table with a sequential scan.

One table must be broadcast so that every segment can compare all rows of one table with the rows it has from the other table. This is expensive and is best used only for small tables.

#### Nested loop with inner index scan

For each row in the first table, the operator searches an index on the second table.

One table must be broadcast so that every segment can compare all rows of one table with the rows it has from the other table.

#### Append

Concatenates data sets. For example, combines rows scanned from multiple partitions.

#### Filter



Selects rows using a **WHERE** clause.

Limit

Limits the number of rows returned.

Materialize

Saves results from a subselect so that it is not necessary to process the inner table for every row in the outer table.

## Join Operators

Hash join

Creates a hash table on the join key of the smaller table. Scans the larger table and looks up matching rows in the hash table. Hash join is very fast. The hash table is held in memory, so a hash join can use a lot of memory, depending on the size of the smaller table.

Sort merge join

The tables to be joined are sorted on the join attribute and then scanned in parallel to find the matching values. This is a good join method for tables that are too large to use a hash join.

Product join

Joins every qualifying row in the first table with every qualifying row in the second table. This type of join can be very expensive if spill files must be used.

## Sort and Aggregate Operators

Sort

Sorts rows to prepare for operations such as an aggregation or merge join.

Group by

Groups rows by one or more columns.

Group / hash aggregate

Aggregates rows using a hash.

## Motion Operators

Broadcast motion

Every segment sends its own local data to all other segment instances so that every segment instance has a complete local copy of the table.

Redistribution motion

Sends data from one table to another segment so that matching rows are located together, enabling a local join.

Gather motion

All segments send rows to the master where they are gathered into a single result set.

## DML Operators

Assert

Performs constraints checking.

Split

Used for update operations.

## Spill Files

Greenplum Command Center reports the total size for all spill files created for a query.

Greenplum Database creates spill files, also called workfiles, to save intermediate results when there is insufficient memory to execute a query in memory. Disk I/O is much slower than memory access, so a query that creates spill files will take longer to complete.

## Investigating Spill File Usage

The `gp_toolkit` schema contains views you can use to see details about spill file usage for current queries. You can see the number and sizes of spill files created for each operator in a query execution plan, and totals by query and segment. This is useful information to detect data skew and to help tune queries.

See the [gp\\_toolkit Administrative Schema](#) reference for descriptions of these views.

## Eliminating or Reducing Spill Files

You can work to eliminate spill files by increasing the amount of memory available to the query or by optimizing the query to use the memory available more efficiently.

You may be able to revise the query to prevent spilling by eliminating or postponing memory-intensive operators.

Following are some ways to increase memory available to queries when resource group resource management is enabled in Greenplum Database.

- Decrease the resource group's concurrency so that each query's share of memory increases.
- Increase the resource group's `MEMORY_SHARED_QUOTA` parameter to increase the amount of resource group shared memory.
- Decrease the percentage of memory allocated to all resource groups to increase the amount of global shared memory.

When resource queue resource management is active, Greenplum Database can detect and terminate "runaway" queries that consume a high percentage of available memory. You can prevent runaway queries by limiting the number of spill files created or the total size of spill files created. See the `gp_workfile_limit*` configuration parameters below for more information.

If you cannot prevent queries from spilling, it is important to ensure that the number of spill files created is minimized and that problems such as CPU or data skew are found and corrected. Skew can create excessive numbers of spill files on one or more segments.

To minimize disk usage and I/O when spill files are created, make sure the `gp_workfile_compression` configuration parameter is set to `on`. When enabled, Greenplum Database uses Zstandard compression for spill files.

## Limiting Spill Files with Server Configuration Parameters

Greenplum Database by default limits the number of spill files allowed per query for each segment to 100,000. You can raise or lower this limit, and you can also limit the number of spill files for all queries on a segment, and limit the disk space consumed by spill files per query and per segment. Use the following Greenplum Database server configuration parameters to manage spill files.

`gp_workfile_limit_files_per_query`

Sets the maximum number of spill files allowed per query per segment. Default is 100,000.

`gp_workfile_limit_per_query`

Sets the maximum disk size an individual query is allowed to use for spill files at each segment. The default value is 0, which means no limit is enforced.

`gp_workfile_limit_per_segment`

Sets the maximum total disk size that all running queries are allowed to use for creating spill files at each segment. The default value is 0, which means a limit is not enforced.

`gp_workfile_compress_algorithm`

Specifies the compression algorithm to use for spill files when a hash aggregation or hash join operation spills to disk during query processing. The default is `'none'`. Set to `'zlib'` to enable compression. Using compression reduces the number of I/O operations at the expense of increased CPU.

See also [Managing Spill Files Generated by Queries](#).

# Alert Help Topics

- Database connectivity failure
- Segment failure
- Average memory (segment hosts) exceeds [%] for [min]
- Memory (master) exceeds [%] for [N] minutes
- Total disk space exceeds [%] full
- Query is blocked for [N] minutes
- Number of connections exceeds [N]
- Average CPU (master) exceeds [%] for [N] min
- Out of memory errors
- Query runtime exceeds [N] minutes
- Average CPU (segment hosts) exceeds [%] for [N] minutes
- Spill files for a query exceeds [GB]
- PANIC happened on master
- FATAL happened on master

## Segment failure

### What is this alert?

Command Center checks the status of the Greenplum Database segments every 30 seconds and raises an alert if any segments are down or running in their non-preferred roles. This alert will be raised hourly until an administrator has recovered the failed segments.

With segment mirroring enabled, Greenplum Database can tolerate a primary or mirror segment failure as long as there is a working instance for every segment in the cluster. If both the primary and mirror instances fail for any single segment, Greenplum Database cannot run queries. For this reason, it is important to recover the failed segment instance to protect from loss of service.

Segment instances have a "preferred role," either primary or mirror. When all segment instances are in their preferred roles, each segment host has the same number of primary and mirror segments. If a primary instance fails, its mirror instance assumes the primary role and the distribution of primary segments is no longer balanced. This can slow down query processing because a host with more primary segments than others can take longer to complete queries.

### What to do

Restore the failed segments and return the segments to their preferred roles so that the cluster is in balance.

See [Recovering From Segment Failures](#) for steps to recover Greenplum Database segments.

## Average memory (segment hosts) exceeds [%] for [min]

### What is this alert?

This alert warns of high memory consumption on the Greenplum Database segment hosts for an extended period.

Every 15 seconds, the Greenplum Database metrics collector extension samples the memory in use on each segment host. Memory used for kernel cache and buffers is excluded. The average for all segments is calculated. If the average percentage remains above the threshold that is set for the alert for the number of minutes specified, an alert is issued.

If memory consumption is increasing over time, queries could start to fail with out of memory errors.

### What to do?

Check the **Query Monitor** to see if there is unusually heavy query activity.

Look for active queries that perform hash joins or sorts on a large number of tuples. If possible, optimize the queries to eliminate rows earlier so that these memory-intensive operations process a fewer number of tuples.

Adjust resource queues or resource groups to limit the number of concurrent transactions.

Use the `ps` command to identify non-database processes consuming excessive memory. Stop unnecessary processes or move them to another server.

## Memory (master) exceeds [%] for [min]

### What is this alert?

When the master host memory usage is higher than the specified percentage for more than the specified number of minutes, Command Center raises an alert.

Command Center samples the memory in use on the master host every fifteen seconds. Memory used for kernel buffers and cache is excluded from the calculation. An alert is raised if the samples collected during the number of minutes specified are all higher than the specified percentage.

### What to do

Use the `ps` command to identify non-database processes consuming excessive memory and stop them or relocate them to other servers.

If Command Center is running on the master host, restart it on the standby master host.

## Total disk space exceeds [%]

### What is this alert?

This alert is raised when the percentage of segment host disk space in use exceeds the percentage specified in the alert rule. The master disk space is not included in the calculation. The alert is raised once a day until the percentage drops below the percentage in the alert rule.

### What to do

This alert warns you so that you can add disk storage or free up storage in order to prevent a catastrophic disk full error that could interrupt Greenplum Database service.

Here are some suggestions for freeing space on Greenplum Database hosts.

- Archive and remove backup files
- Archive and drop older partitions
- Rotate OS and database log files
- Drop unneeded external tables and their data files
- Vacuum database tables and catalog tables

## Query is blocked for [min]

### What is this alert?

If a query that has started to execute is blocked by another query for the specified number of minutes, Command Center raises an alert.

Queries that are queued, but have not yet started, do not trigger this alert.

### Alert details

The alert contains the Query ID, database name and user, and run-time details.

Qid":

```
{"Tmid":1541113373,"Ssid":6968,"Cnt":3,"Database":"postgres","User":"gpmon","SubmitTime":"2018-11-02T16:10:04+08:00","StartTime":"2018-11-02T16:10:04+08:00","QueryText":""}
```

*Need alert type specific JSON example to write this out*

### What to do

1. Use the **Query Monitor** to locate the blocked query and the query that blocks it.
2. Determine whether the blocking query is executing properly:
  - Is the query also blocked?
  - Is the query blocking a large number of other queries?
  - Is the query creating excessive spill files?

- ♦ Is the query running in the correct resource group or resource queue?
  - ♦ Is the query running longer than usual?
  - ♦ Does the query have excessive data or CPU skew?
3. Determine whether you should allow the query to complete, or cancel the query so that the blocked queries can resume.

## Number of connections exceeds [n]

### What is this alert?

This alert is raised when the number of concurrent connections at the Greenplum Database master instance exceeds a specified number. The number specified should be set lower than the `max_connections` server configuration parameter so that when you receive the alert you can act before Greenplum Database begins to reject client connection requests. For example, if `max_connections` for the master is set to 100, you could set an alert to 80.

### What to do

#### Look for idle or stale connections and terminate them

Users can connect to Greenplum Database using a client such as `psql`, execute queries and remain connected, but inactive, leaving the connection in an idle state. Greenplum Database eventually releases resources used by idle connections, but once the maximum number of allowed connections has been reached, new connection requests are denied.

Use the `pg_stat_activity` system view to find idle connections.

```
SELECT datname, procpid, sess_id, username, current_query from pg_stat_activity;
```

Use the `pg_cancel_backend(<PID>)` function to cancel idle connections.

Determining *how long* a connection has been idle is not possible with just the information in the Greenplum Database 5.x system tables. You can, however, see this information by creating the `session_level_memory_consumption` view in the database. Follow the instructions at [Viewing Session Memory Usage Information](#) to create this view in each database.

After you install the `session_level_memory_consumption` view, a query like the following shows the idle connections with the length of time they have been idle.

```
SELECT a.datname, application_name, a.username, a.ssess_id, procpid,
       now()-idle_start AS time_idle
FROM pg_stat_activity a, session_state.session_level_memory_consumption b
WHERE a.ssess_id = b.ssess_id AND b.idle_start < now();
ORDER BY time_idle DESC;
```

## CPU (master) exceeds [%] for [min]

### What is this alert?

The metrics collector extension checks CPU utilization on the master host every 15 seconds. If the percentage of CPU in use is higher than the percentage specified in the alert rule for the number of minutes specified in the rule, Command Center raises an alert.

The Greenplum Database master uses the greatest amount of CPU at the start of a query, while planning the query, and at the end of the query, while gathering results from the segments. For a large result set it is normal to see a spike in the query's CPU usage on the master during the gather operation. With many queries running concurrently, the CPU spikes should even out.

## What to do

Begin by viewing the Command Center **Query Monitor** and **Query Details** views to see if there are one or more very large queries nearing completion, or if the high usage can be explained by unusual workloads or heavy query activity.

If the query monitor offers no explanation for high CPU usage, you should investigate master host processes using Linux utilities such as `ps` and `top` to identify processes consuming the CPU. If the process ID of an errant process is a `postgres` process, you can query the `pg_stat_activity` system table to find the query and, if needed, use the `pg_terminate_backend()` function to terminate the query.

## See also

[Investigating a Performance Problem](#)

## Out of memory error

### What is this alert?

If a query requests additional memory and is denied, the query fails with an out of memory error and an alert is raised.

## What to do

Greenplum Database has two ways to manage memory resources: resource queues and resource groups. Resource queues deal primarily with fixed quantities of memory, where resource groups deal with portions—percentages—of available memory.

If you use resource groups to manage memory, you can use the **Workload Mgmt** view to adjust them so that more memory is available to queries that are failing due to out of memory errors. If you use resource queues, you use the `CREATE RESOURCE QUEUE` and `ALTER RESOURCE QUEUE` SQL commands to configure them.

There are many factors to consider when allocating memory for queries, including configuring the operating system, allocating a share of memory to Greenplum Database, and configuring a set of resource queues or resource groups to share the memory available to Greenplum Database.

For complete information about how Greenplum Database manages memory and how to configure it, see:



- [Using Resource Groups](#)
- [Using Resource Queues](#)

## Query runtime exceeds [min]

### What is this alert?

An alert is raised if the total runtime for a query is greater than the specified number of minutes. The alert is raised once per query.

Run time is calculated from the time the query begins to execute. The time the query was queued is excluded.

### Alert details

- query ID
- database name
- user name
- time the query was submitted
- time the query started

### What to do

Use the **Query Monitor** to check the execution status of the query.

If the query is blocked, investigate the queries that hold the locks.

## Average CPU (segment hosts) exceeds [%] for [min]

### What is this alert?

Command Center samples CPU usage on all segment hosts every 15 seconds and calculates the average CPU usage for the cluster. An alert is raised if the average CPU usage is greater than the specified percentage for longer than the specified number of minutes.

### What to do

Use the Command Center **Query Monitor** to identify currently running individual queries with high CPU usage.

Use the Command Center **History** view to see CPU load during the period prior to the alert and identify completed queries using too much CPU.

Check CPU usage using operating system utilities such as `ps` and `top` to identify any operating system processes that are consuming excessive CPU, for example backup, restore, or ETL processes.

## Spill files for a query exceeds [GB]

### What is this alert?

Command Center raises an alert if the combined size of spill files for any query exceeds the specified number of gigabytes. This alert is raised just once per query.

Greenplum Database creates spill files to temporarily store data on disk when the data exceeds the amount of memory allocated to the operation. Because memory I/O is much faster than disk I/O, a query that creates spill files takes longer to complete than it would if there was sufficient memory available to avoid creating spill files.

### What to do

Use the Command Center **Query Monitor** to view the plan for the query identified in the alert.

If possible, revise the query so that more rows are eliminated earlier in the plan, eliminating or reducing the size of spill files.

Consider reconfiguring the resource queue or resource group that manages the query to make more memory available to the query. If you use resource groups to manage resources, you can use the Command Center **Workload Mgmt** view to modify resource allocations.

### For more information

- [Managing Spill Files Generated by Queries](#)
- [Using Resource Queues](#) for information about configuring Greenplum Database resource queues
- [Using Resource Groups](#) for information about configuring resource groups