Quest

Foglight for PostgreSQL

Cartridge Guide

© 2020 Quest Software Inc.
ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Quest Software Inc.

The information in this document is provided in connection with Quest Software products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Quest Software products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, QUEST SOFTWARE ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL QUEST SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF QUEST SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Quest Software makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Quest Software does not make any commitment to update the information contained in this document.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.

Attn: LEGAL Dept

4 Polaris Way

Aliso Viejo, CA 92656

Refer to our Web site (https://www.quest.com) for regional and international office information.

Patents

Quest Software is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at https://www.quest.com/legal .

Trademarks

Quest, the Quest logo, and Join the Innovation are trademarks and registered trademarks of Quest Software Inc. For a complete list of Quest marks, visit https://www.quest.com/legal/trademark-information.aspx. All other trademarks and registered trademarks are property of their respective owners.

TABLE OF CONTENTS

Introduction to PostgreSQL 7

Foglight for PostgreSQL Overview 8

Foglight for PostgreSQL Requirements 10

Installing and Configuring Agents 11

PostgreSQL Server Pre-Configuration 12

Configuring an Encrypted Connection 14

Cartridge Installation 15

Creating and Configuring Agents 16

Using the Agent Installer Wizard 17

Using the Agent Status Dashboard 18

Agent Properties 19

Upgrading the Agent 23

Removing Monitored Databases 24

Administration 25

Opening the Databases Administration Dashboard 25

Reviewing the Administration Settings 25

Customizing Alarms for Foglight for PostgreSQL Rules 26

Introducing the Alarms View 26

Modifying Alarm Settings 27

Reviewing Rule Definitions 30

Cloning Agent Settings 31

Configuring Email Notifications 32

PostgreSQL Dashboards 35

PostgreSQL Servers 36
Overview 36
Data 36
Actions 36
Admin Actions 37
Server Overview 38
Current Backends 40
Overview 40
Data 40
Actions 40
Admin Actions 40
Current Locks 42
Overview 42
Data 42
Actions 43
Statements 44
Overview 44
Data 44
Actions 45
Statement Details 46
Overview 46
Data 46
Actions 47
Explain Plan Request 47
Replication 49
Overview 49
Data 49

Actions 49
Admin Actions 49
Background Writer 50
Overview 50
Data 50
Databases 52
Overview 52
Data 52
Actions 53
Admin Actions 54
Tables 55
Overview 55
Data 55
Actions 56
Admin Actions 56
Indexes 58
Overview 58
Data 58
Actions 58
Functions 60
Overview 60
Data 60
Actions 60
Admin Actions 61
Tablespaces 62
Overview 62
Data 62

Actions 63

PostgreSQL Reports 64

PostgreSQL Cartridge Rules 65

Approaching Connection Limit 65

Backend Write Percentage High 65

DB Running Out of Disk Space 66

High Percentage of Queries Waiting 68

Index Bloat 69

Low Buffer Hit Percentage – Database 69

Low Buffer Hit Percentage – Table 70

Low Percent of Checkpoints Required 71

PostgreSQL Server Availability 71

Potential Deadlock Issue 71

Replication Lag 72

Slow Connection 72

Introduction to PostgreSQL

PostgreSQL is a powerful, open source object-relational database system originally developed at UC Berkeley by computer science professor Michael Stonebraker, who went on to become the CTO of Informix Corporation. Stonebraker started Postgres in 1986 as a follow-up project to its predecessor, Ingres, now owned by Computer Associates. With more than 15 years of active development, it has earned a reputation for ensuring reliability and data integrity. PostgreSQL supports all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows.

PostgreSQL has evolved into an enterprise class database and boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance. It supports international character sets, multibyte character encodings, Unicode, and it is locale-aware for sorting, case-sensitivity, and formatting. It is highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL systems in production environments that manage in excess of 4 terabytes of data.

Many organizations, government agencies and companies use PostgreSQL. You will find installations in ADP, CISCO, NTT Data, NOAA, Research In Motion, The US Forestry Service, and The American Chemical Society. Today, it's rare to find a large corporation or government agency which isn't using PostgreSQL in at least one department.

Foglight for PostgreSQL Overview

Foglight and PostgreSQL - A Powerful Combination

Foglight for PostgreSQL brings, true enterprise-class monitoring for PostgreSQL to the industry's leading cross-platform database management solution. On its own, Foglight for Postgres is a powerful monitoring solution. When used in conjunction with other components from the Foglight suite, it has the ability to provide true end-to-end transaction insight. Response times can be collected for end user sessions and transactions and can be correlated with metrics from the application and database layers as well as the underlying physical and virtual infrastructure. The ability to collect and correlate information across infrastructure layers further increases the value of the solution, making Foglight for PostgreSQL an invaluable part of this industry-leading Application Performance Monitoring (APM) solution.

From high-level server operations to database and table-level statistics, Foglight for PostgreSQL provides real-time monitoring and alerting, offering users real-time insight into the performance and availability of their PostgreSQL environment. The solution leverages the holistic data analysis and presentation that have made Foglight a leading Gartner Magic Quadrant APM solution. The addition of PostgreSQL monitoring to Foglight provides customers with the ability to monitor PostgreSQL availability and performance alongside Oracle, MS-SQL Server, MySQL, DB2/UDB, and Sybase. Databases can be managed using a common interface, making it the leading cross-platform database management solution on the market today. With the ability to be used as a standalone solution or in conjunction with other solutions available from Dell, Foglight for PostgreSQL is a unique solution offering best-in-class monitoring and an unmatched value proposition.

Ease of Installation

Installation of the solution is straightforward and can be accomplished in minutes. It can be deployed as an agent-based or agentless solution. The agent-based deployment requires that agent code run locally (on the DB host) whereas the agentless configuration allows the agent to be run remotely from another host. Once installed, the solution provides granular control of collection intervals and the option to disable collections for entire databases or database child objects for which monitoring is deemed unnecessary. Within minutes, Information, Dashboards, and Expert advice is available, providing a platform that will ensure the heath and availability of PostgreSQL for years to come.

Data Visualization and Reporting

Foglight for PostgreSQL was designed to provide quick access to vital database statistics. Utilizing rich dashboards and robust data analysis, the solution uncovers critical information on server operations, including the background writer, write ahead logs, connections, locks, and replication status. The solution integrates data from Foglight host monitoring and incorporates these metrics into key dashboards, permitting system data to be presented alongside rich database metrics. Dashboards display server health and availability information alongside vital statistics such as resource allocation and system metrics from the host machine.

Database and table-level statistics are exposed through the use of both standard and derived metrics. These metrics include memory hit rates, read rates, tuple modifications, transaction commits/rollbacks, standby server conflicts, disk sizes, and deadlocks. Additional information is provided for databases, tables, and indexes. Function comparisons show frequency of use and execution time statistics. Information displayed in dashboards can be used to create and schedule reports. Operators then have

the ability to utilize the data as-is, or define their own dashboards and reports using the built-in drag-and-drop functionality.

Administration and Workflow Automation

For those customers who wish to administer their PostgreSQL environment directly from Foglight, some basic administrative actions are available. With properly assigned permissions, these actions can be taken directly from the Foglight console. Actions can also be triggered as a result of a condition, or grouped together in workflows. Customers who wish to identify and store additional actions may do so on demand.

Capacity Planning and Alerting

The solution incorporates functionality for both trending/capacity planning and alerting. To facilitate capacity planning, a tablespaces dashboard exposes growth trends over time, a breakdown of disk space usage on the host machine, and remaining available space. Alerting is based on a robust set of rules and definitions that are included with the solution. Rules can be triggered based on both static and dynamic thresholds. Dynamic thresholds are triggered when the system detects a deviation from normal behavior patterns. The system will automatically create a baseline for usage and alert on environmental anomalies.

Paying Dividends

With feature-rich out-of-the-box dashboards and rules, PostgreSQL for Foglight provides unparalleled insight into your PostgreSQL infrastructure and offers immediate benefits to DBAs and Administrators. Over time, the solution continues to pay dividends by offering historical trend analysis, scheduled reporting, alert management, and the ability to create user-defined dashboards.

Combined, these features bring great insight into database and application availability and ensure improved access to mission-critical data.

Foglight for PostgreSQL Requirements

Foglight for PostgreSQL is compatible with PostgreSQL version 9.1+ and managed database service versions using an equivalent or greater version, such as AWS Aurora/RDS and Microsoft Azure. However, some data may be missing from versions prior to 9.2, most notably I/O timing metric statistics, which are unavailable in those versions.

The agent may be run on a FglAM that is either local or remote to the PostgreSQL server. More information on configuring the PostgreSQL serverr for monitoring can be found in the PostgreSQL Server Pre-Configuration section of this document.

Installing and Configuring Agents

Installation of Foglight for PostgreSQL is covered in the following sections and should be performed in order:

- PostgreSQL Server Pre-Configuration
- Cartridge Installation
- Creating and Configuring Agents

PostgreSQL Server Pre-Configuration

In order to allow full monitoring of the PostgreSQL Server, the agent will require a user with sufficient privilege to execute system queries. Also, several features of the server must be enabled in the server configuration file, typically the postgresql.conf file in a native instance or through the configuration UI in a hosted instance like AWS Aurora or RDS.

PostgreSQL User Permissions

The PostgreSQL agent requires use of a database user with certain minimum privileges for the agent to be able to fully monitor the database.

Create a new user with an existing user that has the appropriate permissions or a utility program of your choice. The new user must have access to all databases you wish to monitor. You must also ensure that any firewalls and the pg_hba.conf file allow the created user access from whichever FglAM host the agent is running from. The agent will only query system tables and views and make use of a few system functions in order to gather operational data.

User privileges required for the PostgreSQL agent:

- LOGIN
- SUPERUSER
- CONNECT (on all databases)
- SELECT (on all databases)

The SUPERUSER privilege is required in order to make use of administration functions through the Foglight console. Usage of the functions is restricted to Foglight users who have been granted the "PostgreSQL Administrator" role.

Example:

CREATE USER user1 WITH PASSWORD 'password1' LOGIN SUPERUSER;

Workload and I/O Metrics

The track_io_timing property must be set to "on". More info is available here: https://www.postgresql.org/docs/current/static/runtime-config-statistics.html.

Statement Monitoring

Installation of the pg_stat_statements module and creation of the extension in a database is required for statement tracking. Basic steps follow below. For more information on how to install this, refer to the relevant PostgreSQL documentation for your version.

1) In the configuration file, set the below sample properties. The pg_stat_statments module allows more advanced configuration, but these are the minimum required.

   shared_preload_libraries = 'pg_stat_statements'
   pg_stat_statements.max = 1000
   pg_stat_statements.track = all

   More info on properties: https://www.postgresql.org/docs/current/static/pgstatstatements.html

2) Restart the server.

3) In PSQL, pgAdmin, or another client program, execute this statement when connected to any database (postgres is fine): "CREATE EXTENSION pg_stat_statements;" The pg_stat_statements view should appear in DB's public schema. If the public schema does not exist, create this first.

4) When configuring the Foglight for PostgreSQL agent properties, make sure to enable statement tracking and provide the database name where the extension was created.

Configuring an Encrypted Connection

The below instructions cover common steps used to configure an encrypted connection from the PostgreSQL Agent client. For full information on secure connections and server-side configuration, refer to the 18.9 Secure TCP/IP Connections with SSL section of the PostgreSQL documentation for your version.

In order to use SSL, the ssl parameter must be set in the PostgreSQL server's postgresql.conf file and the server certificate and private key must exist and be in the default data directory or another directory specified in the configuration.

If the client is required to supply a trusted certificate, root CAs must also be present on the server and the pg_hba.conf file should require the connecting user to provide a certificate.

For more information on client-side SSL configuration, refer to the Using SSL section of the PostgreSQL JDBC Driver documentation. From here, the agent can use two methods to access certificates, covered in the following sections. When finished, restart the FglAM and continue with the agent configuration.

Using the Filesystem

The first is simply accessing them on the filesystem of the FglAM host. File paths are provided to the agent through the file path agent properties in the Setting SSL Connection Parameters section in this document.

If using an SSL Key file, it must be converted to PKCS8 format with the following command:

```
openssl pkcs8 -topk8 -inform PEM -outform DER -in postgresql.key -out postgresql.pk8 -nocrypt
```

The "-nocrypt" argument can be dropped to leave the key encrypted, however a password must be created during encryption and provided in the agent properties.

Using the Java Truststore

The other method requires the certificates to be imported into the Java truststore. The server certificate will first need to be converted into a different format, using this command:

```
openssl x509 -outform DER -in server.crt -out server.crt.der
```

Then, it must be imported into the FglAM truststore. You can use the bundled keytool, which will be located in the Foglight Agent Manager\jre\1.8.0.72\jre\bin directory, or the equivalent on your system, with these commands:

```
keytool.exe -import -file server.crt.der -keystore truststore -alias postgresql
```

If you have not changed the password for the keystore, the default password will be "changeit". Next, edit the baseline.jvmargs.config file in the Foglight Agent Manager\state\default\config directory and add the following parameters with file paths and passwords appropriate for your system. Escape any quotes with a '\'.

```
vmparameter.0 = "-Djavax.net.ssl.trustStore=\"C:/Program Files/Common Files/Dell/Foglight Agent Manager/jre/1.8.0.72/jre/bin/truststore\"";
vmparameter.1 = "-Djavax.net.ssl.trustStorePassword=changeit";
```

Cartridge Installation

1. Open the Foglight Web Console.
2. From the navigation pane, select: **Dashboards** > **Administration** > **Cartridges** > **Cartridge Inventory**. The Cartridge Inventory screen appears. For more information on agents, see the *Foglight User Guide.*
3. Load the ***PostgreSQLAgent-5_9_xx.car*** file by browsing to the location where the .car file exists and then clicking on “Install Cartridge”. Leave the “Enable on Install” check box checked.
4. Once the installation is completed on the Foglight Management Server, the PostgreSQL Agent Cartridge will appear in this list below as an installed cartridge.

Creating and Configuring Agents

Agents can be created in one of two ways:

- Using the Agent Installer Wizard
- Using the Agent Status Dashboard

The Agent Installer Wizard simplifies the agent creation and configuration process and can be accessed from the Databases dashboard. For advanced configuration or modification of agent properties post-creation, use the Agent Status dashboard.

Using the Agent Installer Wizard

Foglight for PostgreSQL provides a graphic, intuitive method for creating and configuring agents, which can be used instead of Foglight's default method for creating agents and editing their properties using the Agent Status dashboard. Foglight for PostgreSQL allows running a wizard that provides a common entry point for adding database instances and then configuring these instances for monitoring.

To run the instance installation wizard:

1. On the navigation panel, click Homes > Databases.
2. Click the PostgreSQL box in the Databases View, and then click Monitor.
3. The Agent Installer Wizard dialog box appears.
4. The first card - Agent Deployment – has two fields:
    a. Agent Name – Provide a name for the agent that will be created. This is not canonical and should be representative of the database instance that this agent will monitor.
    b. Agent Manager - Choose the agent manager on which the agent should run. Considerations for this may include physical or virtual locality to the monitored instance, allocated resources, or grouping with other agents of the same type or monitored environment. If the agent package has not been deployed to this Agent Manager yet, it will be installed when the first agent of this type is created.
5. The second card – Agent Properties – requires a basic set of parameters for connecting to and monitoring the database instance. A full explanation of these properties is available in the Agent Properties section of this document.
6. The third card – Agent Summary – displays a review of the configuration that will be created and an option allowing the agent to be activated after creation. If the configuration looks good, click the Finish button to start the process.
7. When the process completes, a results screen will appear showing the results of agent creation. If the agent was not created, follow the instructions on the results screen. If successful, the database instance should appear in the Databases table within a few minutes.

Note: If the agent was created successfully but data is not appearing, go to the Dashboards > Administration > Agents > Agent Status page and click the icon in the Log File column for the agent you created. In most cases, the reason for the failure will be obvious. You can also refer to the *Foglight for PostgreSQL Installation and Troubleshooting* document for common errors and solutions. If the solution requires reconfiguring the agent properties, follow steps 3-7 of the Using the Agent Status Dashboard section.

Using the Agent Status Dashboard

The Agent Status page can be used to create new agents and configure and manage existing agents. To access the page from the navigation pane, select: Dashboards > Administration > Agents > Agent Status.

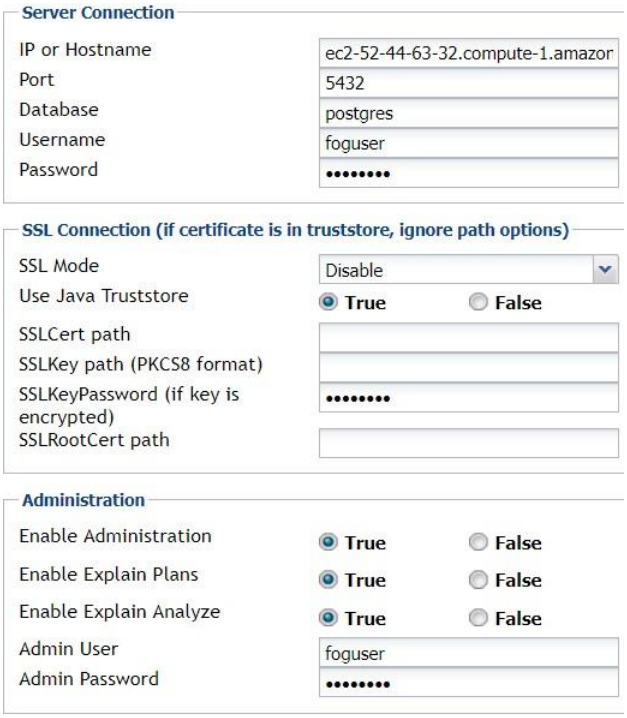
Use the following steps to create a new agent instance:

1. If the PostgreSQL agent package has never been deployed to the FglAM that will be used to host the agent, this must be done before an agent has been created. You can use the Deploy Agent Package button on the Agent Status or Agent Managers page to perform this.
2. Click the Create Agent button and follow the instructions for the cards:
    a. **Host Selector** - Choose the Agent Manager on which the agent should run. Considerations for this may include physical or virtual locality to the monitored instance, allocated resources, or grouping with other agents of the same type or monitored environment.
    b. **Agent Type and Instance Name** – Select the PostgreSQLAgent type. Then, select the Specify Name radio button and provide a name for the agent that will be created. This is not canonical and should be representative of the database instance that this agent will monitor.
    c. **Summary** – Click Finish.
3. Once the agent has been created, click the checkbox next to the PostgreSQL agent.
4. Click the **Edit Properties** button.
5. Select **Modify the default properties for this agent**.
6. Edit the agent properties for the PostgreSQL agent instance:
    - Setting Server Connection Parameters (mandatory)
    - Setting SSL Connection Parameters (optional)
    - Setting Administration Options (optional)
    - Setting Collection Intervals (optional)
    - Setting DB Overrides (optional)
    - Setting Options (optional)
7. Click the **Activate** button.

To modify the properties for an existing agent, skip to step 3 and Deactivate, then Reactivate the agents after changing the configuration.

Agent Properties

This is a full list and explanation of the configurable properties of the Foglight for PostgreSQL agent. The Agent Installer Wizard provides access to the essential subset of available properties. To modify other properties or modify the agent configuration after creation, use the Agent Status dashboard.



Setting Connection Parameters (mandatory)

- **IP or Hostname** – Host where PostgreSQL database server is running. Default is “localhost”. (e.g.<hostname> or <IP address>)
- **Port** – Listening port of the PostgreSQL database server. Default is 5432.
- **Database** – Name of PostgreSQL database for server data collection. The agent will monitor all databases by default, but this is the database that will be accessed for primary data collection.
- **Username** – User that can connect to the PostgreSQL server being monitored
- **Password** – Password of the user that can connect to the PostgreSQL server being monitored.

Setting SSL Connection Parameters (optional)

- **SSL Mode** – Sets SSL mode for JDBC connection and verification level for connection. For more information on mode levels, read the Configuring the Client section of the PostgreSQL JDBC documentation. If you are not using an SSL connection, set to disabled and ignore the rest of this section.
- **Use Java Truststore** – If set to true, the agent will look for the SSL certificates in the Java truststore. The certificates will need to be imported into the truststore first. Read the Using the Java Truststore section for more information. To access the certificates from the filesystem, set to false and use the fields below.
- **SSLCert path** – The full path of the certificate file.
- **SSLKey path** – The full path of the key file.

- **SSLKeyPassword** – If the key file is encrypted, provide the password here.
- **SSLRootCert path** – The full path of the SSL root certificate.

Setting Administration Options (optional)

The Administration options allow for certain server operations to be performed on the monitored server from the FMS console. These operations are restricted to Foglight users with the PostgreSQL Administrator role.

- **Enable Administration** – Allows server operations to be performed from the FMS console.
- **Enable Explain Plans** – Allows explain plans to be requested on demand through the FMS console. Does not require that Administration also be enabled.
- **Enable Explain Analyze** – If this is enabled, the Analyze option in the Explain Plan request form will be selectable. The EXPLAIN ANALYZE command will carry out the command on the server and return actual run times and other statistics. Administrators should be very careful when using this option.
- **Admin User** – User with the SUPERUSER privilege that is able to perform the enabled administrative operations. This user will only be used in order to carry out these operations, not for normal data collection.
- **Admin Password** – Password for the admin user.

Setting Collection Intervals (optional)

The Collection Interval fields in the agent properties are used to set the sample frequencies. You can turn off a collection by setting the interval to 0.

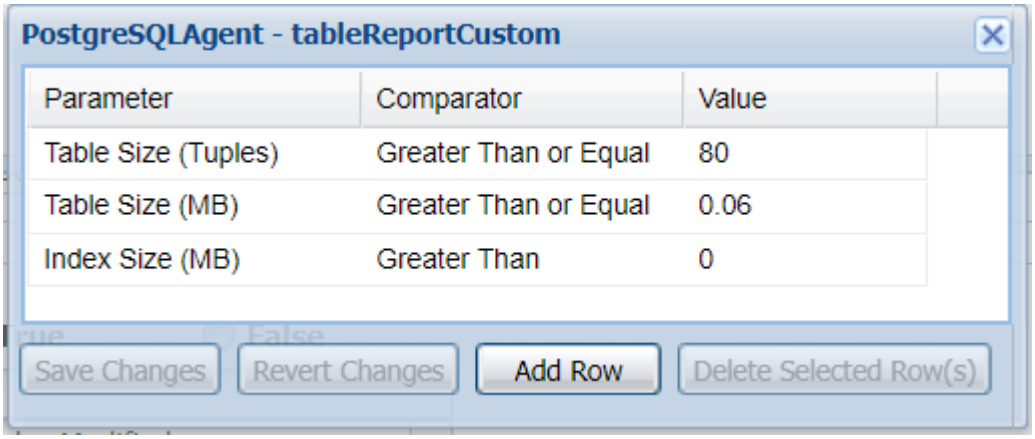
- **DB List Refresh (min)** – Checks for databases on the server in order to schedule collections for those databases. As databases are not often added or removed, this does not have to happen frequently. Set to 30 minutes by default.
- **Availability** – A basic heartbeat collection that checks whether the server is running and how long it takes to connect. Should happen frequently. Set to 60 seconds by default.
- **Server and DBs** – Collects data on server metrics and configuration as well as database-level statistics. Set to 300 seconds by default.
- **Tables, Indexes, & Functions** – Collects table, index, and function statistics from each database. Set to 300 seconds by default.
- **Configuration** – Collects server configuration. Set to 3600 seconds by default.

Top Tables Collection (optional)

The Table Collection section is used to configure collection of the top tables on the server.

- **Enable Top Tables Collection** – Enables the Top Tables collection. Can be disabled on a per-database basis in the DB Overrides section. Set to true by default.
- **# of Top Tables (per DB)** – Sets the number of tables to collect per database. Once a table is collected, it will continue to be collected even if it is no longer within the limit until the agent is restarted. Can be changed on a per-database basis in the DB Overrides section.
- **Sort By** – Sets the parameter that determines which tables to include in the collection. Ex: If this is set to Total Table Size and "# of Top Tables" is set to 100, the 100 largest tables from each database will be collected. Set to Scans (Count) by default.

Tables Report (optional)



The Tables Report section is used to configure a limited collection of tables based on one or several parameters. Tables collected in this manner will not be created as persistent topology objects and will only appear in the PostgreSQL Full Tables Report.

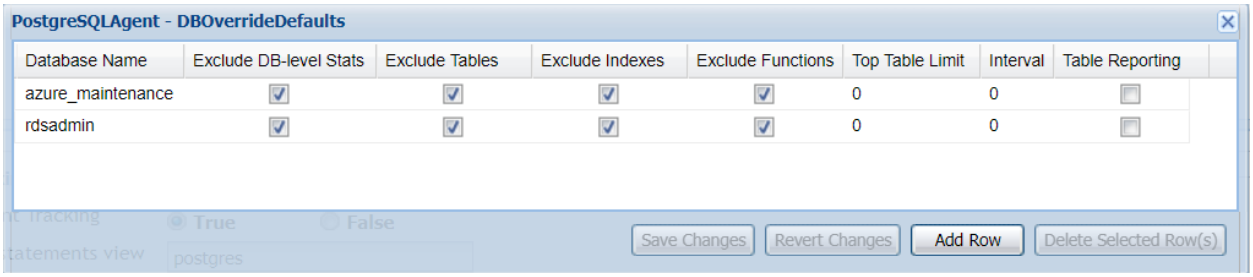
- **Enable Table Reporting** – Enables the Table Report collection. Can be disabled on a per-database basis in the DB Overrides section. Set to false by default.
- **Reporting Interval (hrs)** – Sets the interval to collect information for the PostgreSQL Full Tables Report. Set to 12 hours by default.
- **Required Parameters** – Determines whether collected tables must satisfy all or any of the parameters specified in the Report Parameters setting. Set to All by default.
- **Report Parameters** – Parameters for the collection are set here. In the Parameter column, select one of several available parameters. The value of this for each table will be evaluated with the comparator and value column values. If the expression evaluates to true, the table will be included in the report (depending on whether it meets any or all of the parameters, configured in the previous setting). Ex: Table Size (Tuples) greater than or equal to 80 will match all tables that have at least 80 live tuples (inclusive).

Statement Tracking (optional)

If the pg_stat_statements module and extension is enabled in the PostgreSQL server, this section can be used to configure statement statistics tracking.

- **Enable Statement Tracking** – Enables or disables use of statement tracking.
- **DB w/ pg_stat_statements view** – While statement tracking is performed across all databases, the pg_stat_statements extension is only available in databases where it has been created. If one or multiple databases have the pg_stat_statements view, you may supply the name of any of them in this field.
- **# of Top Statements** – The number of statements for the agent to collect. If you wish to collect all of them, provide the number of the pg_stat_statements.max setting, or anything higher.
- **Sort By** – The statements are sorted in order to gather the # of top statements specified in the previous property. If all statements are being gathered, this field is unimportant. Otherwise, several self-explanatory options are available based on existing or calculated column data in the view.

Setting DB Overrides (optional)



The DB override properties allow you finer-grained control over database collections. If a database is not specified here, it will use the parameters set in the Collection Intervals section.

- **Database Name** – Name of the database where collections are being overridden.
- **Exclude DB-level Stats** – If set to true, database-wide statistics (aggregated from objects belonging to the database) will not be collected for this database.
- **Exclude Tables** – If set to true, statistics for individual tables belonging to the database will not be collected.
- **Exclude Indexes** – If set to true, statistics for individual indexes belonging to the database will not be collected.
- **Exclude Functions** – If set to true, statistics for individual functions belonging to the database will not be collected.
- **Top Table Limit** – Will override the number of tables collected for the Top Tables collection.
- **Interval** – This collection interval will override the default Tables, Indexes, and Functions setting in the Collection Intervals section for this database.
- **Table Reporting** – Will override the Enable Table Reporting. If set to true, the Table Report collection will be enabled for this database. If false, it will not.

Setting Options (optional)

- **Enable Dynamic Memory Allocation** – Enables dynamic memory allocation so that the agent will request additional memory from the FglAM based on the expected amount of data. Should not be used without recommendation from Quest Support. Set to false by default.
- **Agent Host Name** – The hostname property for the PostgreSQL server topology object is by default set to the connection parameter value defined above. However, you can override it with this setting. If you are monitoring the OS, it is important that the hostname provided by the PostgreSQL agent and OS agent match so that they can be linked.

Upgrading the Agent

1. Go to Dashboards > Administration > Cartridges > Cartridge Inventory and click the Install Cartridge button.
2. Locate the .car file on your system and install it with auto-enable selected. If you get a message that a bundled cartridge is of an older version than the one currently enabled on your FMS and will not be enabled, ignore it and continue.
3. Once the cartridge is installed and enabled, go to Dashboards > Administration > Agents > Agent Managers. Agent Managers that can be upgraded with newer agent packages will show “yes” in the Upgradable | Agents column. Select all Agent Managers you wish to upgrade and click the Upgrade button.

Note: If an Agent Manager is not upgradable, check that the Agent Manager version is compatible with the newer agent version. If it is not, the Agent Manager will need to be upgraded first.

Removing Monitored Databases

1. Go to the Databases dashboard.
2. Select the databases you wish to remove.
3. Click the Settings button, then click ok.

Note: Doing this will remove the monitoring agents as well as the historical data already collected. If you wish to delete only the agents, you can do that on the Administration > Agents > Agent Status page. Because the Databases dashboard only shows databases which are being actively monitored, you will only be able to view these databases by going through the PostgreSQL > PostgreSQL Servers dashboard.

Administration

Opening the Databases Administration Dashboard

You can edit agent settings for one or more PostgreSQL instances on the Databases > Administration dashboard.

> **NOTE:** If you attempt to select instances of more than one type of database, such as a PostgreSQL database and an Oracle database, an error message is displayed.

To open the Databases Administration dashboard:

1. In the navigation panel, under **Homes**, click **Databases**.
2. Select the check boxes beside one or more PostgreSQL instances.
3. Click **Settings** and then click **Administration**. The Administration dashboard opens, containing settings for all the selected agents. Settings are broken down into categories, which are organized under a PostgreSQL tree.

Reviewing the Administration Settings

The Databases Administration dashboard allows settings options for collecting, storing, and displaying data, which apply to all the currently selected agents. Click a category of settings on the left (for example: Connection Details) to open a view containing related settings on the right.

To view the full list of selected agents, click the **Selected Agents** button at the upper right corner of the screen. To change the list of agents to which the metrics will apply, exit the Databases Administration dashboard, select the requested agents and re-open the view.

Customizing Alarms for Foglight for PostgreSQL Rules

Many Foglight for PostgreSQL multiple-severity rules trigger alarms. To improve your monitoring experience, you can customize when alarms are triggered and whether they are reported. You can also set up email notifications.

Introducing the Alarms View

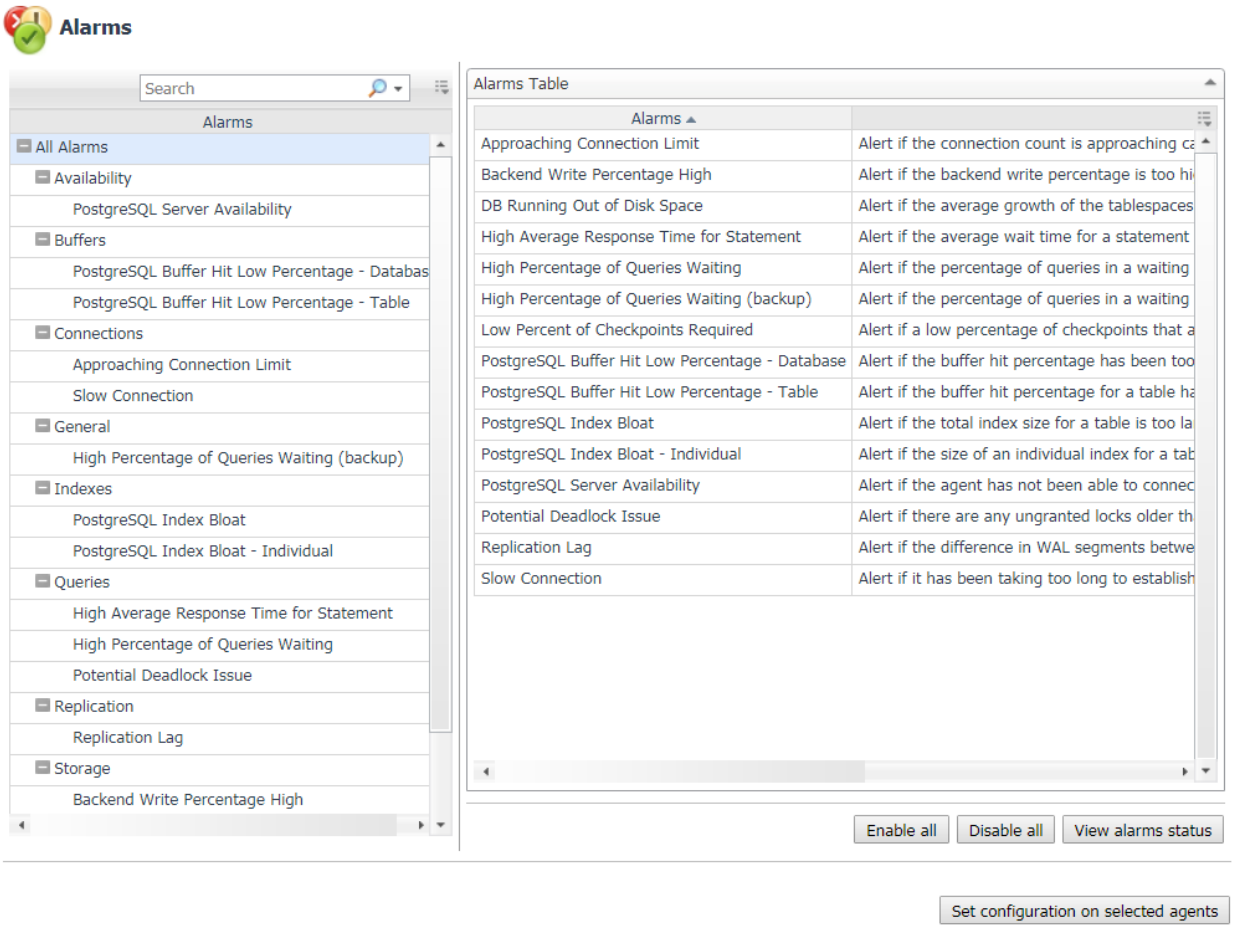
The Alarms view enables you to modify global settings and agent-specific settings for alarms.

To open the Alarms view:

1. Open the Administration dashboard as described in Opening the Databases Administration Dashboard.
2. Select the agents you wish to modify and do one of the following steps:
    a. Select the Settings button and open the Administration dashboard, then click Alarms.
    b. Select the 'Configure Alarm' button.
3. From the Alarms view, you can complete the following tasks:
    a. Modifying Alarm Settings
    b. Reviewing Rule Definitions
    c. Cloning Agent Settings

Modifying Alarm Settings

You can customize how the alarms generated by the default rules are triggered and displayed in the Alarm view. Changes to alarm settings will apply to all selected agents, though thresholds can be customized by individual agent.



The Alarms list controls the contents displayed to the right and the tasks that are available.

- **All Alarms –** Displays all rules with configured alarms and indicates whether alarms are enabled. In this view, you can enable or disable alarms for all the rules at once. You can also set email notifications and define mail server settings.
- **Category of rules –** Displays a set of related rules with configured alarms. In this view, you can enable or disable alarms and also set email notifications for the category of rules.
- **Rule name –** Displays the alarm status for the selected rule. If the rule has multiple severity levels, displays the threshold configured for each severity level. In this view, you can enable or disable the alarm, edit the alarm text, and edit severity levels and their thresholds. You can also set email notifications for the alarm.

You can complete the following tasks:

- Enabling or disabling alarms for selected agents
- Modifying alarm threshold values
- Editing the text of the alarm message

Your changes are saved separately and applied over the default rules. This protects you from software upgrades that may change the underlying default rules.

Enabling or disabling alarms for selected agents

You can override the global alarm sensitivity level setting for the selected agents. You can enable or disable alarms for all rules, a category of rules, or an individual rule.

To see descriptions of the rules, follow the steps described in Reviewing Rule Definitions.

To enable or disable alarms:

1. Navigate to the Alarms view.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:

Scope	Procedure
All alarms	Click **All Alarms**. In the Alarms Settings tab, click either **Enable all** or **Disable all**.
Category of rules	Click a category. Click either **Enable all** or **Disable all**.
Selected rule	Click the rule. In the Alarms Settings tab, click the link that displays the alarm status. Select **Enabled** or **Disabled** from the list and click **Set**.

Modifying alarm threshold values

You can and should modify the thresholds associated with alarms to better suit your environment. If you find that alarms are firing for conditions that you consider to be acceptable, you can change the threshold values that trigger the alarm. You can also enable or disable severity levels to better suit your environment.

When a rule has severity levels, a Threshold section appears in the Alarm Settings tab showing the severity levels and bounds by agent. The threshold values correspond to the lower bounds shown in this table. Many rules do not have severity levels and thresholds.

When editing thresholds, ensure that the new values make sense in context with the other threshold values. For most metrics, threshold values are set so that Warning < Critical < Fatal. However, in metrics where normal performance has a higher value, such as DBSS - Buffer Cache Hit Rate, the threshold values are reversed: Warning > Critical > Fatal.

To change severity levels and thresholds:

1. Navigate to the Alarms view.

2. Click the multiple-severity rule that you want to edit.
3. Click the **Alarms Settings tab.**
4. In the Threshold section, review the defined severity levels and existing threshold bounds for all target agents.
5. Modify the severity levels for one or more agents by following one of the following procedures:

Task	Procedure
Edit severity levels and set threshold values for all agents.	Click **Enhance alarm**. Select the check boxes for the severity levels you want enabled and set the threshold values. Click **Set**.
Change the threshold values for one agent.	Click **Edit** beside the agent name. Set the new threshold values and click **Set**.
Copy the changes made to one agent's threshold values to all other agents.	Click **Edit** beside the agent name that has the values you want to copy. Select **Set for all agents in table** and click **Set**.

Editing the text of the alarm message

For individual rules, you can change the message displayed when an alarm fires. You cannot add or remove the variables used in the message. This is a global setting that affects all agents.

To change the alarm message:

1. In the Alarms view, click the **Settings** tab.
2. Select a rule.
3. Click the **Alarm Settings** tab.
4. Click **Enhance alarm**. A Customize <rule> dialog box opens.
5. In the Message box, edit the message text. To restore the default message, click **Reset message**.
6. Click **Set**.

Reviewing Rule Definitions

If you want to review the conditions of a rule, open the rule in the Rule Management dashboard.

> **IMPORTANT:** Avoid editing rules in the Rule Management dashboard unless you are creating your own rules or copies. These rules may be modified during regular software updates and your edits will be lost.

You can create user-defined rules from the Rule Management dashboard. If you want to modify a rule, we recommend copying the rule and creating a user-defined rule. User-defined rules need to be managed from the Rule Management dashboard; these rules are not displayed in the Alarms view of the Databases Administration dashboard. For help creating rules, open the online help from the Rule Management dashboard.

To open the Rule Management dashboard:

1. On the navigation panel, under **Homes**, click **Administration**.
2. In the Administration dashboard, click **Rules**.
3. Type **PostgreSQL** in the Search field to see the list of predefined rules for PostgreSQL databases. The PostgreSQL rules are displayed. From here, you can review threshold values, alarm counts, and descriptions.
4. To see the full rule definition, click a rule and then click **View and Edit**.
5. In the Rule Detail dialog box, click **Rule Editor**.
6. When you are done with your review, click Rule Management in the bread crumbs to return to the dialog box.
7. Click **Cancel** to avoid changing the rule unintentionally.

Cloning Agent Settings

You may want an agent to have the same settings as another agent. For example, if you add new agents, you may want them to use the same settings as an existing agent. In this case, you can clone the settings from one agent to other agents. This process does not link the agents; in the future if you update the source agent, you also need to update the target agents.

This procedure walks you through selecting the source agent from the Databases dashboard. However, you can also open the Administration dashboard with multiple agents selected. In this case, you select the source agent in Clone Alarm-related Settings to Other Agents dialog box.

To clone alarm-related settings:

1. On the Databases dashboard, select the check box for the agent with the settings you want to clone.
2. Click Settings and then Administration.
3. In the Administration dashboard, click Alarms.
4. Click Set configuration on selected agents. The Clone rule settings across agents dialog box opens.
5. In the Select the source agent drop-down list, you should see the agent you selected.
6. In the Select the target agents table, select the check boxes for agents that should inherit settings from the source agent.
7. Click Apply.
8. When prompted for confirmation, click Yes.

Configuring Email Notifications

We recommend that you set email notifications for the alarms you are most interested in tracking closely. For example, you may want to be notified by email of any Critical or Fatal situation. Or you may want to be informed whenever a key metric is no longer operating within acceptable boundaries.

You can set up email notifications that are generated when an alarm fires and/or on a defined schedule, as described in the following topics:

- Configuring an email server
- Defining Default Email settings
- Enabling or disabling email notifications
- Defining email notifications, recipients, and messages
- Defining variables to contain email recipients

Configuring an email server

You need to define the global mail server variables (connection details) to be used for sending email notifications.

The setting of the email should be configured in Foglight Administration > Email configuration.

Defining Default Email settings

You can define a default email address to be used by every new agent created in the future, by selecting the Default email button when configuring email notification.

The Email addresses entered are applied to all monitored agents not only for the agents that were selected to enter the Alarm administration.

Enabling or disabling email notifications

You can enable or disable email notifications for all alarms, a category of alarms, or a selected rule. Email notifications are sent only if all the following conditions are met:

- The alarm email notification setting is enabled for the affected rule.
- The alarm is triggered by changes in the monitored environment.
- Alarm notification is enabled at the triggered severity level. See Defining email notifications, recipients, and messages.

To enable or disable email notifications:

1. In the Alarms view, click the Settings tab.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:
   - All alarms - Click All Alarms. Click the Define Email Settings button. Select either Enabled or Disabled from the Alarms notification status list. Click Set.
   - Category of rules - Click a category. Click the Define Email Settings button. Select either Enabled or Disabled from the Alarms notification status list. Click Set.

- Selected rule - Click a rule. In the Alarms Settings tab, click the Define Email Settings tab. Click the link that displays the alarm notification status. Select Enabled or Disabled and click Set.

Defining email notifications, recipients, and messages

You control who receives email messages, the subject line, and some text in the body of the email. The body of the email always contains information about the alarm. This information is not editable. You can also control whether an email is sent based on severity levels. You can set different distribution lists for different rules and different severity levels, or set the same notification policy for all rules.

To configure email notifications:

1. In the Alarms view, click the Settings tab.
2. Decide on the scope for the change: all alarms, a category of rules, or a selected rule.
3. Complete the steps for the selected scope:
   - All alarms - Click All Alarms. Click the Define Email Settings button.
   - Category of rules - Click a category. Click the Define Email Settings button.
   - Selected rule - Click a rule. Click the Email Notification Settings tab.
4. If you selected All Alarms or a category, in the Email Notification Settings dialog box, do one of the following:
   - To change the severity levels that warrant an email notification, from the Messages will be enabled for severities box, select the desired levels of severity.
   - To configure the same email recipients and message for all severity levels, click Configure mail recipients for all Severities and then click All severities.
   - To configure different email recipients and messages for each of the severity levels, click Configure mail recipients for the following options and then click a severity level.
5. In the Message Settings dialog box, configure the email recipients and message. Note that you can use registry variables in place of email addresses. Type the variable name between two hash (#) symbols, for example: #EmailTeamName#. For more information, see Defining variables to contain email recipients.
   - To — Type the addresses of the people who need to take action when this alarm triggers.
   - CC — Type the addresses of the people who want to be notified when the alarm triggers.
   - Subject — Optional. Edit the text of the subject line to better suit your environment. Avoid editing the variables, which are identified with the @ symbol.
   - Body Prefix — Optional. Add text that should appear above the alarm information in the body of the email.
6. Click Set to save the message configuration and close the dialog box.
7. If the Edit Notification Settings dialog box is open, click Set.

Defining variables to contain email recipients

You can create registry variables that contain one or more email addresses, and use these registry variables when defining email notifications. This procedure describes how to create a registry value.

To create a registry variable:

1. On the navigation panel, under Dashboards, click Administration > Rules & Notifications > Manage Registry Variables.

2. Click Add. The New Registry Variable Wizard opens.
3. Select the registry variable type String, and click Next.
4. In the Name field, enter a name, for example: EmailTeamName
5. Click Next.
6. Select Static Value.
7. In the Enter desired value box, enter one or more email addresses (separated by commas).
8. Click Finish.

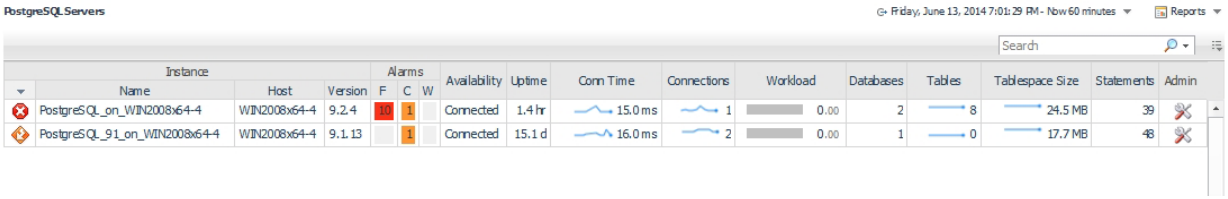
PostgreSQL Dashboards

The installation of the PostgreSQL Cartridge includes the PostgreSQL Dashboards. The PostgreSQL Cartridge offers four principal dashboards:

- PostgreSQL Servers
- Server Overview
- Databases
- Tablespaces

Drill downs from these dashboards are available to expose more granular data. It should be noted that there is a great deal of additional information that is collected that is not displayed in these primary dashboards. Should an operator wish to have that information displayed, Foglight allows for the creation of simple drag and drop dashboards.

PostgreSQL Servers



Overview

The PostgreSQL Servers dashboard displays a row for each PostgreSQL agent instance deployed. This table gives a brief overview of each server's health, availability, response time, and size.

Data

- **Instance**
  - **Health** – Shows the overall health of the deployed instance.
  - **Name** – The name of the PostgreSQL agent. This is the name entered when the PostgreSQL agent instance was created.
  - **Host** – Hostname of the machine where the PostgreSQL server is located.
  - **Version** – The version number of the PostgreSQL server being monitored.

- **Alarms**
  - **Fatal** – Provides a count of the "Fatal" alerts for this agent.
  - **Critical** – Provides a count of the "Critical" alerts for this agent.
  - **Warning** – Provides a count of the "Warning" alerts for this agent.

- **Other Information**
  - **Availability** – Whether the agent is able to connect to the PostgreSQL server.
  - **Uptime** – How long the server has been running for.
  - **Conn Time** – The amount of time needed for the agent to establish a successful JDBC connection to the server.
  - **Connections** – The current number of backends connected to the server.
  - **Workload** – The combined workload of database-level workload metrics, workload is a calculation of time spent reading and writing data file blocks by backends in a sample period, divided by the sample period time. This represents the load on the server and may exceed 1 because of concurrent processing. This is not an estimation of resource usage or availability. Requires track_io_timing to be enabled, available only in v9.2+.
  - **Databases** – The current number of databases contained by the server.
  - **Tables** – The current total number of user tables in the server.
  - **Tablespace Size** – The current size on disk of the server's combined tablespace files.
  - **Statements** – The number of statements gathered in the last sample. This collection may be limited due to the # of Top Statements property in the Agent Properties or the pg_stat_statements.max setting on the PostgreSQL server.

Actions

The PostgreSQL Servers dashboard is conveniently linked to several pages and also contains popup information to provide a natural workflow while investigating server performance. Several fields in the

table have dwell (hovering over the cell value) or drilldown (clicking on the cell value) capability. A list of available interactions with the table is displayed below:

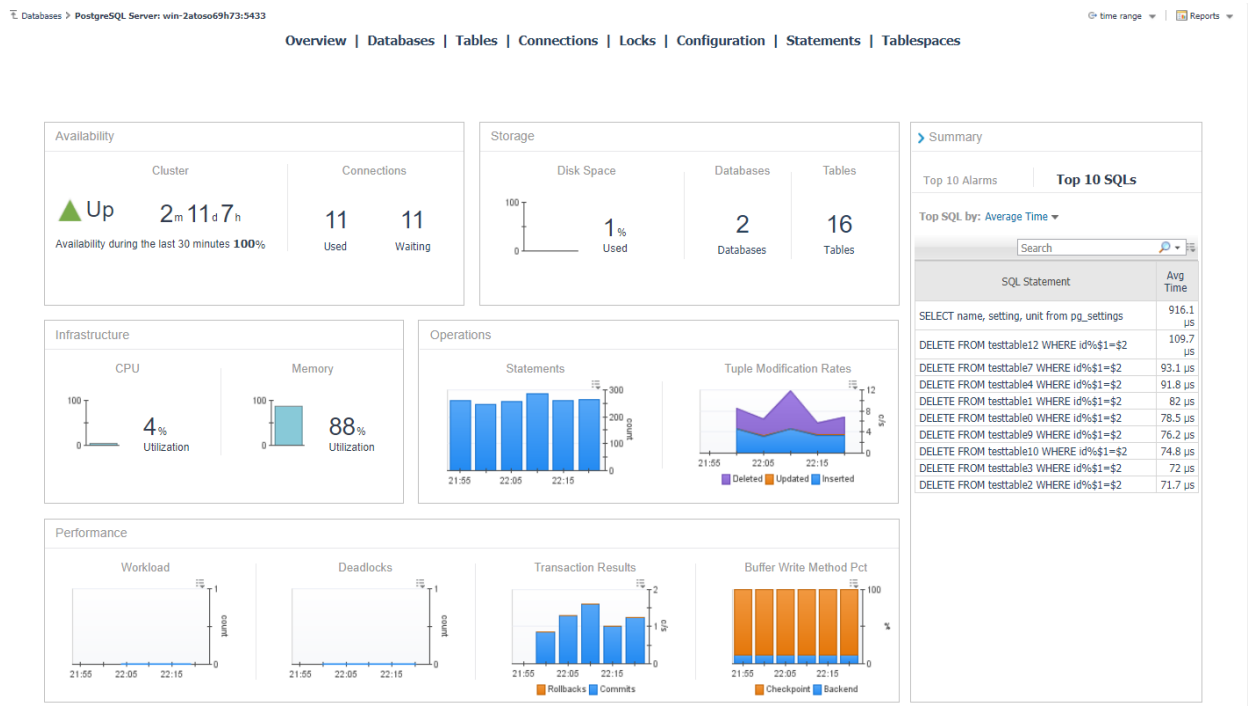
- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Name** (drilldown) – Navigates to the Server Overview dashboard for the selected server.
- **Host** (drilldown) – Navigates to the Host Monitor page for the selected host. This page will only show data if you are monitoring the OS for this host. For more information on this page, see the related documentation for the infrastructure cartridge.
- **Alarms** (dwell or drilldown) – Displays a list of alarms in the selected severity level.
- **Conn Time** (dwell or drilldown) – A popup time plot of the connection time.
- **Connections**
    - Dwell - A popup time plot of connections.
    - Drilldown – Navigates to the Current Backends page.
- **Workload** (dwell or drilldown) – Displays a popup time plot of workload.
- **Databases** (drilldown) – Navigates to the Databases dashboard.
- **Tables**
    - Dwell - A popup time plot of total tables.
    - Drilldown – Navigates to the Tables page.
- **Tablespace Size**
    - Dwell - A popup time plot of the tablespace size.
    - Drilldown – Navigates to the Tablespaces page.
- **Statements** (drilldown) – Navigates to the Statements dashboard.

Admin Actions

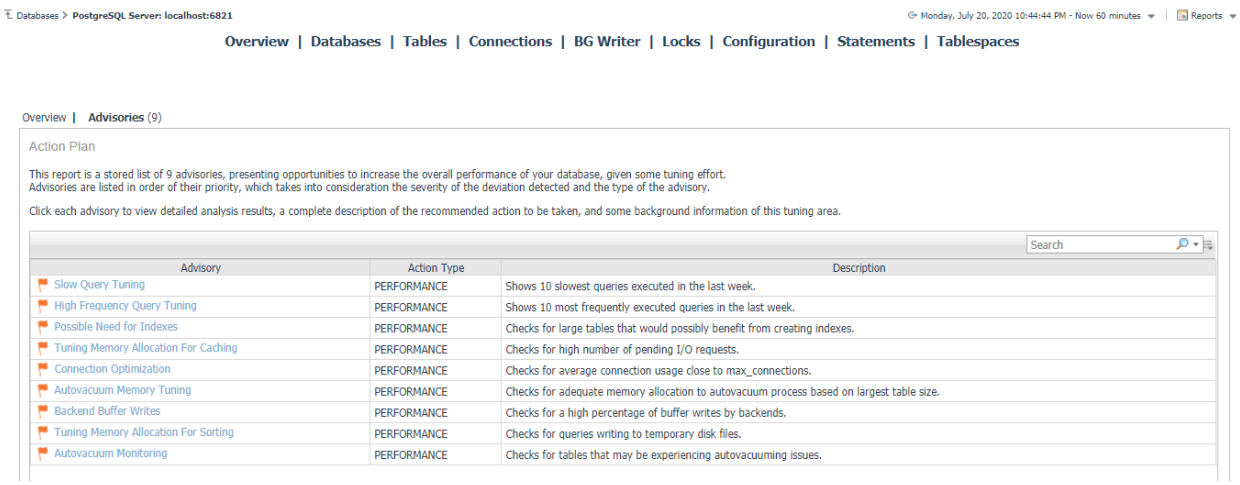
If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Statement Stat Reset** – If using the pg_stat_statements module, this action empties the pg_stat_statements view and resets all statement statistics. The extension will then repopulate the view based on new statements. The next sample by the PostgreSQL agent will collect these as normal.

Server Overview

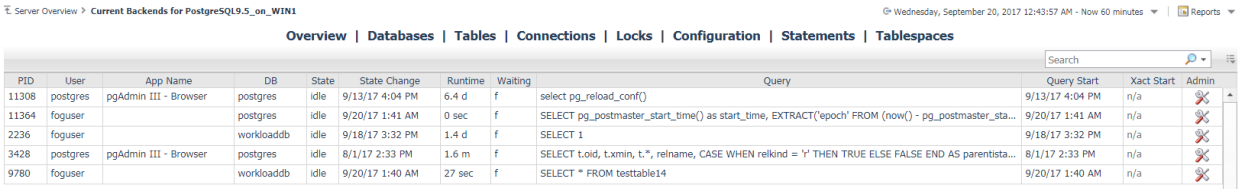


The Server Overview dashboard gives an overview of key segments for the PostgreSQL server, including host metrics. Information includes availability and connection status, storage and host information, and operations and performance metrics. The right side of the page features an expandable summary box that can display the Top 10 SQL executed in the current time range by one of several metrics or the Top 10 Alarms currently active for this server. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.



A link at the top left shows the number of active Advisories and allows switching between the Overview and Advisories pages. Clicking on any of the listed Advisories will launch a popup of the full Advisory, which contains information and advice on improving the performance, security, or other aspects of the monitored server.

Current Backends



PID	User	App Name	DB	State	State Change	Runtime	Waiting	Query	Query Start	Xact Start	Admin
11308	postgres	pgAdmin III - Browser	postgres	idle	9/13/17 4:04 PM	6.4 d	f	select pg_reload_conf()	9/13/17 4:04 PM	n/a	
11364	foguser		postgres	idle	9/20/17 1:41 AM	0 sec	f	SELECT pg_postmaster_start_time() as start_time, EXTRACT('epoch' FROM (now() - pg_postmaster_sta...	9/20/17 1:41 AM	n/a	
2236	foguser		workloaddb	idle	9/18/17 3:32 PM	1.4 d	f	SELECT 1	9/18/17 3:32 PM	n/a	
3428	postgres	pgAdmin III - Browser	postgres	idle	8/1/17 2:33 PM	1.6 m	f	SELECT t.oid, t.xmin, t.*, relname, CASE WHEN relkind = 'r' THEN TRUE ELSE FALSE END AS parentista...	8/1/17 2:33 PM	n/a	
9780	foguser		workloaddb	idle	9/20/17 1:40 AM	27 sec	f	SELECT * FROM testtable14	9/20/17 1:40 AM	n/a	

Overview

The Current Backends page shows active backends in the server from the last server data sample and related information. In the action panel on the right pane, the DB Selector allows you to switch between databases or show backends from all databases.

Data

- **PID** – Process ID of the backend.
- **User** – The user name that owns the backend.
- **App Name** – The application where the backend originated from, if provided in the connection string.
- **DB** – Name of the database the backend is connected to.
- **State** – Current overall state of the backend.
- **State Change** – Time when the last time the backend state was changed.
- **Runtime** – How long the backend has existed.
- **Waiting** – Whether or not the backend is waiting for a lock to open in order to perform its query.
- **Wait Event** – Wait event type if the backend is waiting (9.6+).
- **Wait Event Type** – Wait event name if the backend is waiting (9.6+).
- **Query** – Current or last query executed by the backend. Shown in an abbreviated state in order to fit in the table cell.
- **Query Start** – Time when the currently active query was started or start of the last query if not active.
- **Xact Start** – Time when the backend's current transaction was started or null if no transaction is active.

Actions

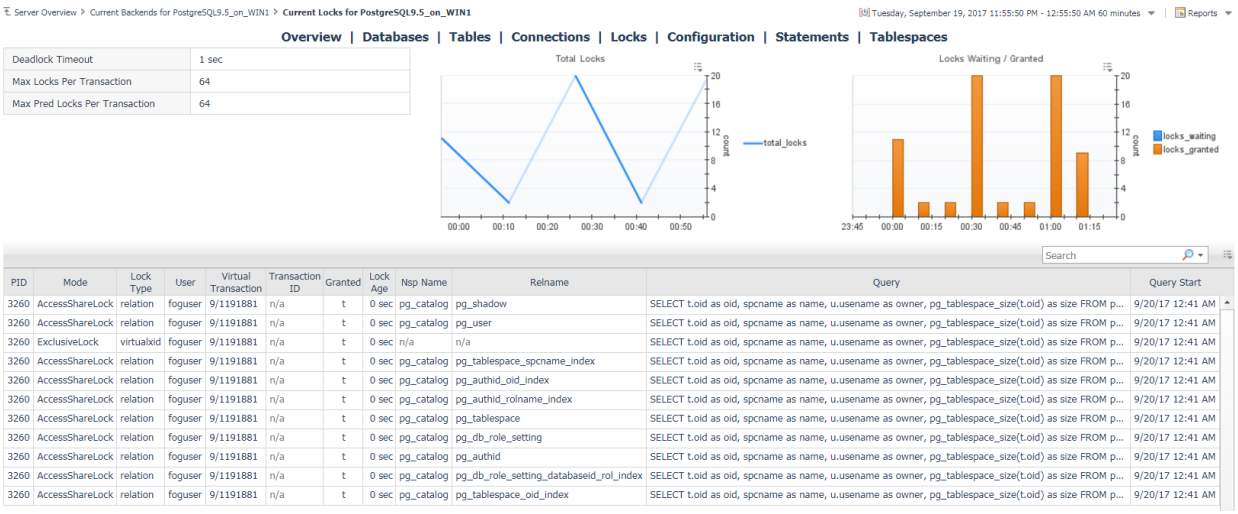
- **DB** (drilldown) – Navigates to the Databases dashboard with DB selected.
- **Query** (dwell or drilldown) - Shows the full query text in a formatted, colored state for better readability.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Cancel** – Cancel the backend's current query.
- **Terminate** – Terminates the backend, i.e. kills the process.

Current Locks



Overview

The Current Locks page shows current locks in the server from the last server data sample and related information. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Data

Top-left Table Section

- **Deadlock Timeout** – The amount of time that the server waits on a lock before checking to see if there is a deadlock condition.
- **Max Locks Per Transaction** – The average number of object locks allocated for each transaction. Individual transactions can lock an infinite amount of objects, but as a whole, the server cannot allow an average of more than this number across all active transactions.
- **Max Pred Locks Per Transaction** - The average number of predicate locks allocated for each transaction. Individual transactions can lock an infinite amount of objects, but as a whole, the server cannot allow an average of more than this number across all active transactions.

Top Graphs

- **Total Locks** – Total locks on the server.
- **Locks Waiting/Granted** – Shows number of locks waiting and granted.

Table

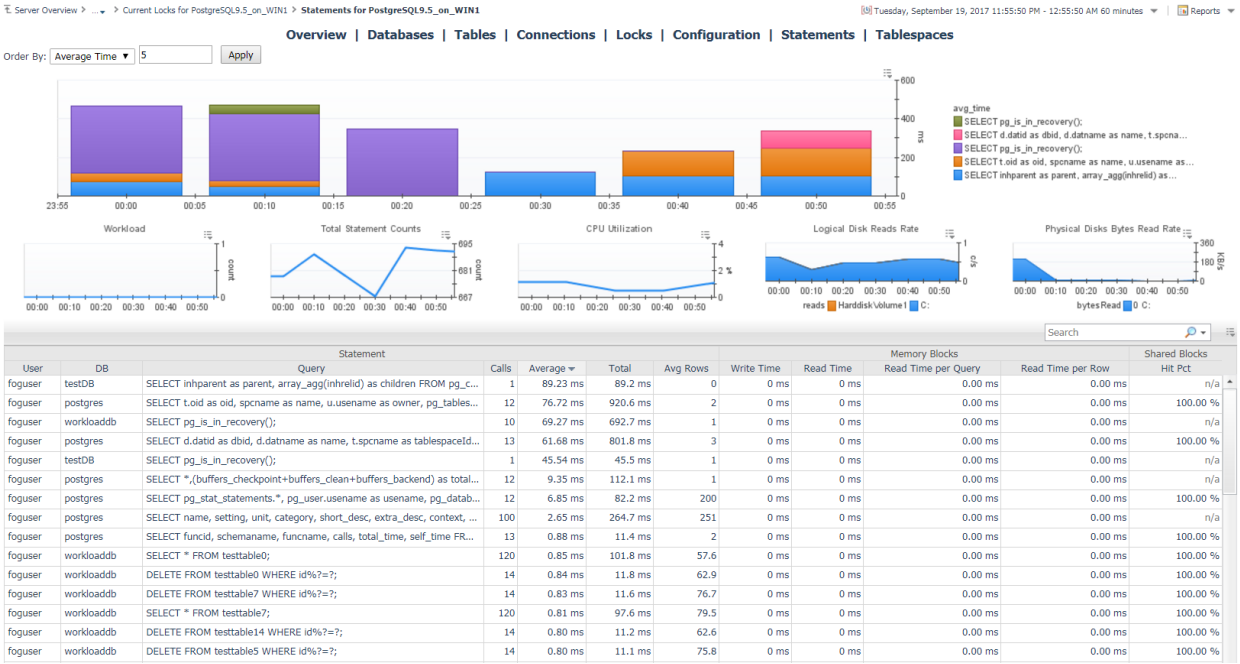
- **PID** – Process ID of the lock.
- **Mode** – Name of the lock mode held or desired by the process.
- **Lock Type** – Type of the lockable object.
- **User** – User controlling the backend that requested the lock.
- **Virtual Transaction** – Virtual ID of the transaction.
- **Transaction ID** – ID of the transaction targeted by the lock or null if target is not a transaction.
- **Granted** – True if lock is held, false if it is waiting to be granted.

- **Lock Age** – How long since lock process started.
- **Nsp Name** – Name of the namespace.
- **Relname** – Name of the object targeted by the lock.
- **Query** – Current or last query executed by the backend. Shown in an abbreviated state in order to fit in the table cell.
- **Query Start** – Time when the currently active query was started or start of the last query if not active.

Actions

- **Query** (dwell or drilldown) - Shows the full query text in a formatted, colored state for better readability.

Statements



Statement							Memory Blocks				Shared Blocks
User	DB	Query	Calls	Average	Total	Avg Rows	Write Time	Read Time	Read Time per Query	Read Time per Row	Hit Pct
foguser	testDB	SELECT inhparent as parent, array_agg(inhrelid) as children FROM pg_c...	1	89.23 ms	89.2 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tables...	12	76.72 ms	920.6 ms	2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	SELECT pg_is_in_recovery();	10	69.27 ms	692.7 ms	1	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT d.datid as dbid, d.datname as name, t.spcname as tablespaceId...	13	61.68 ms	801.8 ms	3	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	testDB	SELECT pg_is_in_recovery();	1	45.54 ms	45.5 ms	1	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT *,(buffers_checkpoint+buffers_clean+buffers_backend) as total...	12	9.35 ms	112.1 ms	1	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT pg_stat_statements.*, pg_user.usename as usename, pg_datab...	12	6.85 ms	82.2 ms	200	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	postgres	SELECT name, setting, unit, category, short_desc, extra_desc, context, ...	100	2.65 ms	264.7 ms	251	0 ms	0 ms	0.00 ms	0.00 ms	n/a
foguser	postgres	SELECT funcid, schemaname, funcname, calls, total_time, self_time FR...	13	0.88 ms	11.4 ms	2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	SELECT * FROM testtable0;	120	0.85 ms	101.8 ms	57.6	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable0 WHERE id%?=?;	14	0.84 ms	11.8 ms	62.9	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable7 WHERE id%?=?;	14	0.83 ms	11.6 ms	76.7	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	SELECT * FROM testtable7;	120	0.81 ms	97.6 ms	79.5	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable14 WHERE id%?=?;	14	0.80 ms	11.2 ms	62.6	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
foguser	workloaddb	DELETE FROM testtable5 WHERE id%?=?;	14	0.80 ms	11.1 ms	75.8	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %

Overview

If Statement Tracking is enabled, the Statements dashboard shows a number of the top statements collected from the server, sorted and limited by fields specified in the Agent Properties. The bar graph can be customized using the fields above it to show top statements by several categories. The graphs below show workload, statement counts, and system utilization metrics (if host is being monitored), in order to allow the user to correlate potentially problematic statements with impact on the system. Below, a table lists all collected statements that have been called in the selected time period. Many of these fields are hidden by default for space considerations, but can be made visible by using the customizer at the top right of the table. A default search filter is set in the top-right section of the table to exclude calls made to system databases, usually internal processes or monitoring queries. This can be cleared or modified to show different results in the table. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between servers without leaving the page.

Data

Statements Table

- **Statement**
  - **User** – Name of the user that executed the statement.
  - **User ID** – ID of the user that executed the statement.
  - **DB** – Name of the database in which the statement was executed.
  - **DB ID** – ID of the database in which the statement was executed.
  - **Query** – Text of a representative statement (up to track_activity_query_size bytes). Statements with the same operations plan may have slight differences in the original text, but are represented by a single statement.
  - **Calls** – Number of times the statement has been executed.

    - **Rows** - Total number of rows retrieved or affected by the query.
    - **Average** – Average time spent executing the statement per call.
    - **Total** – Total time spent executing statements.
- **Memory Blocks**
    - **Write Time** - Total time the statement spent writing blocks (if track_io_timing is enabled, otherwise zero).
    - **Read Time** - Total time the statement spent reading blocks (if track_io_timing is enabled, otherwise zero).
    - **Read Time per Query** - Time the statement spent reading blocks per query.
    - **Read Time per Query** - Time the statement spent reading blocks per row retrieved/affected.
- **Shared Blocks**
    - **Hit Pct** – Percentage of shared block cache hits of total blocks read.
    - **Read** – Number of shared blocks read.
    - **Hit** – Number of shared block cache hits.
    - **Written** – Number of shared blocks written.
    - **Dirtied** – Number of shared blocks dirtied.
- **Local Blocks**
    - **Read** – Number of local blocks read.
    - **Hit** – Number of local block cache hits.
    - **Written** – Number of local blocks written.
    - **Dirtied** – Number of local blocks dirtied.
- **Temp Blocks**
    - **Read** – Number of temp blocks read.
    - **Written** – Number of temp blocks written.

Actions

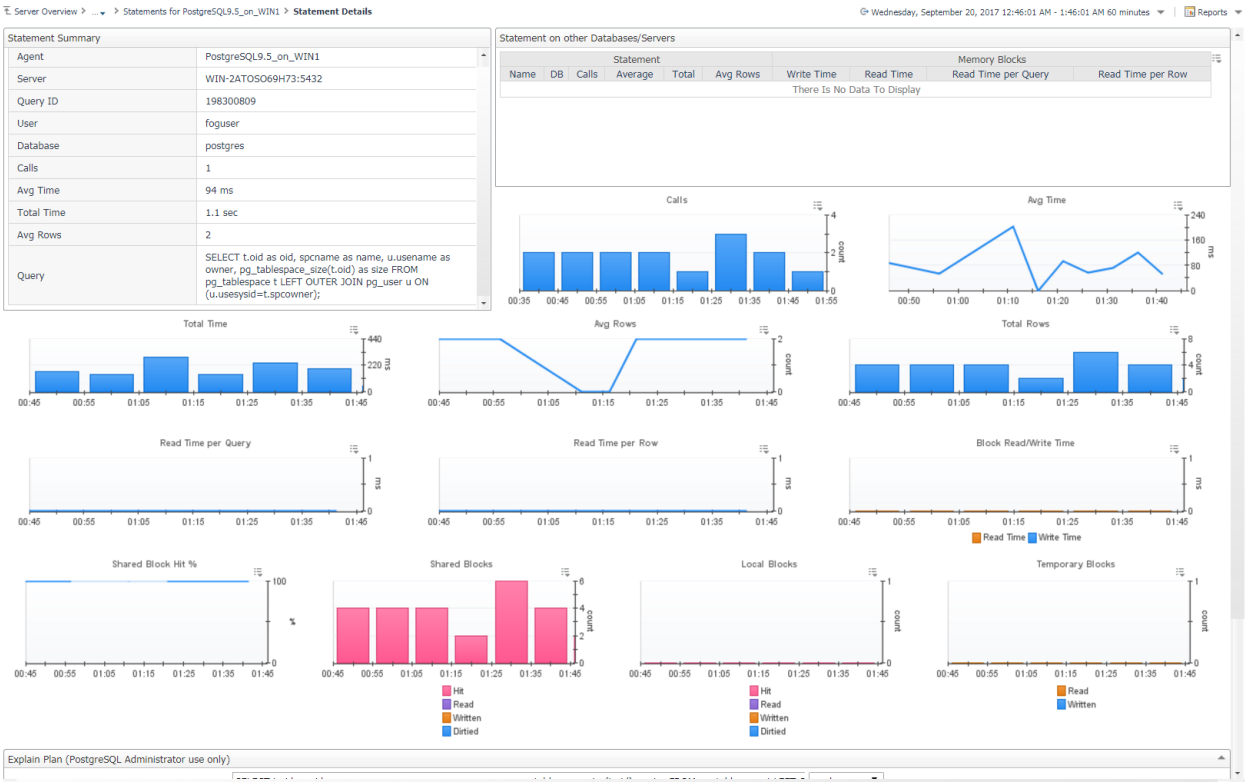
Statements Graph

- **Bar Section** (dwell) – Popup of selected statement details.
- **Bar Section** (drilldown) – Navigates to the Statement Details page.

Statements Table

- **DB** (drilldown) – Navigates to the Databases dashboard with DB selected.
- **Query** (drilldown) - Shows the full query text in a formatted, colored state for better readability.
- **Other metrics** (drilldown or dwell) – Shows a popup time plot for the selected metric.

Statement Details



Overview

The Statement Details page provides a more in-depth look at the operations and history of a particular statement as well as its performance on other monitored servers, if the same queryid has been found. This page also features the Explain Plan Request form, which can be used to request an explain plan for the statement if it has been enabled in the agent properties and the Foglight user has the PostgreSQL Administrator role.

Data

- **Statement Summary**
  - **Agent** – Agent monitoring the PostgreSQL Server.
  - **Server** – The host:port of the PostgreSQL server.
  - **Query ID** – An internally-generated hash calculation. For versions below 9.4, where queryid is not available from the server, a hash code has been generated by the agent based solely on a concatenation of the database name and query text
  - **User** – Name of the first user that executed the statement.
  - **Database** – Name of the database in which the statement was executed.
  - **Calls** – Number of times the statement has been executed.
  - **Average Time** – Average time spent executing the statement per call.
  - **Total Time** – Total time spent executing statements.
  - **Avg Rows** – Average number of rows retrieved or affected by the query.

- **Query** – Text of a representative statement (up to track_activity_query_size bytes). Statements with the same operations plan may have slight differences in the original text, but are represented by a single statement.

- **Statements On Other Servers**
  - This table shows performance of the same query on other monitored PostgreSQL servers on which it may be performed, for comparison to the selected server.

    In order to match the queries across servers, this table relies on the queryid which is not guaranteed to be the same across servers.

    Per PostgreSQL documentation (https://www.postgresql.org/docs/current/static/pgstatstatements.html):

    "As a rule of thumb, queryid values can be assumed to be stable and comparable only so long as the underlying server version and catalog metadata details stay exactly the same. Two servers participating in replication based on physical WAL replay can be expected to have identical queryid values for the same query. However, logical replication schemes do not promise to keep replicas identical in all relevant details, so queryid will not be a useful identifier for accumulating costs across a set of logical replicas. If in doubt, direct testing is recommended."
- **Other Graphs**
  - Represent historical views of metrics listed on the Statements page table.

Actions

Statements On Other Servers Table

- **Name** (drilldown) – Switches the page to represent the same statement as performed on the selected server.

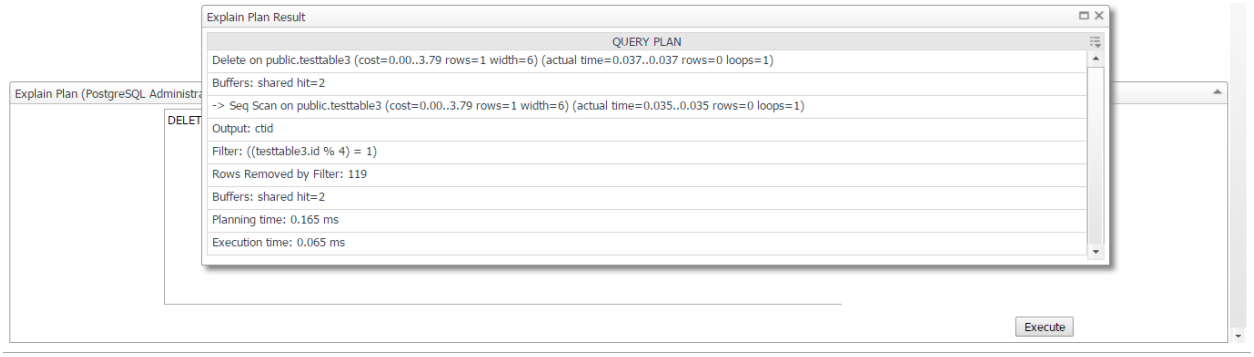
Explain Plan Request

If Enable Explain Plans is set to true in the agent properties, this form allows a user with the PostgreSQL Administrator role to request explain plans. The text area field will be pre-filled with the normalized query text of the statement being investigated. It can (or must be) modified to an appropriate form for the PostgreSQL server to consider it a legitimate statement (? replaced with real numerical or string values). The database is also auto-selected, but may be switched.

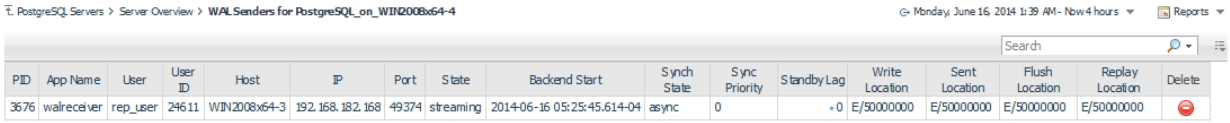
If Enable Explain Analyze has been set to true in the agent properties, the Analyze option in this form will be selectable. If selected, the EXPLAIN ANALYZE command will carry out the command on the server and return actual run times and other statistics. Administrators should be very careful when using this

option. If unset, the server will not actually run the query and will instead return an expected execution plan.

When done, click the Execute button to request the Explain plan and wait. After a short time, a popup should launch with the results of the explain plan request similar to the screenshot below. The Foglight user and statement performed will also be logged to the Agent Log.



Replication



Overview

The Replication page contains a table that displays one WAL sender per row with configuration and current status information.

Data

- **PID** – The process ID of the WAL sender.
- **App Name** – Name of the application connected to the WAL sender process.
- **User** – Name of the user logged into the WAL sender process
- **User ID** – ID of the user logged into the WAL sender process
- **Host** – Host of the connected WAL receiver client.
- **IP** – IP address of the connected WAL receiver client.
- **Port** – Port of the connected WAL receiver client.
- **State** – Current WAL sender state.
- **Backend Start** – Time when the client first connected to the WAL sender.
- **Synch State** – Synchronous state of the standby server.
- **Sync Priority** – Priority of the current standby server client being chosen as the synchronous standby.
- **Standby Lag** – Number of WAL segments the standby server is behind as compared to the primary server. If this number increases, it means that the standby server is not able to keep up and replication may eventually be broken.
- **Write Location** - Last transaction log position written to disk by this standby server.
- **Sent Location** - Last transaction log position sent on this connection.
- **Flush Location** - Last transaction log position flushed to disk by this standby server.
- **Replay Location** - Last transaction log position replayed into the database on this standby server.

Actions

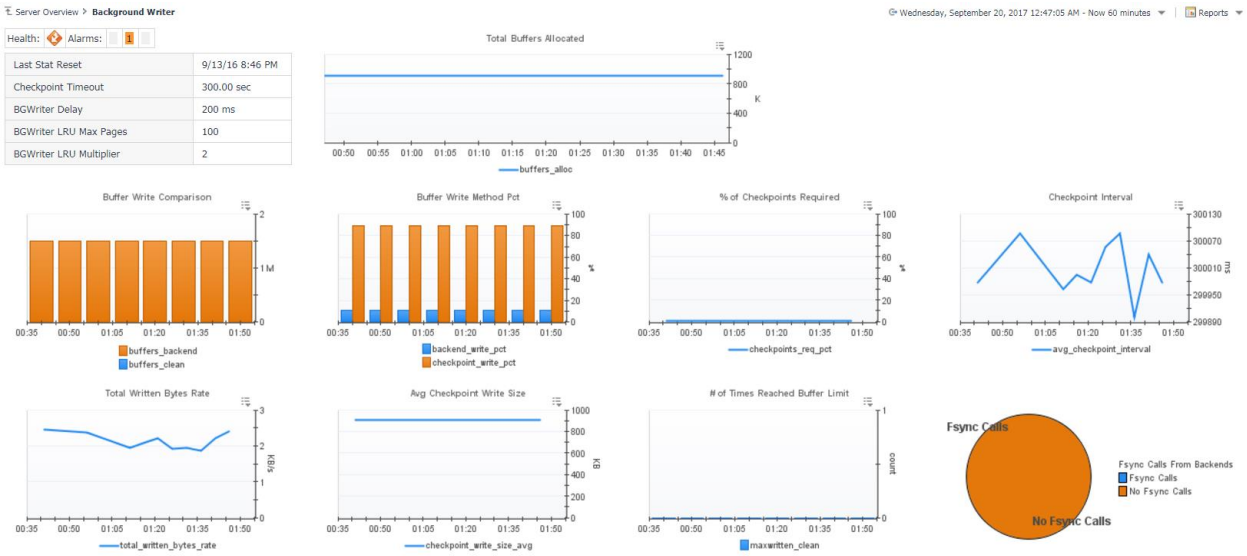
- **Standby Lag** (dwell) - Shows a time plot of the standby lag.

Admin Actions

Click the icon in the Delete table column to perform the following action:

- **Delete** – Delete the WAL Sender topology object if it becomes unnecessary. This will not affect the PostgreSQL server. Because the topology objects are persisted on the FMS, an object may become redundant or unnecessary if the WAL sender is no longer in use. You can manually delete it with this action. If it is still in use, the object will just be recreated.

Background Writer



Overview

The Background Writer page shows important configuration information as well as performance metrics related to the background writer.

Data

Top-left Table Section

- **Health** – Shows the overall health of the background writer.
- **Alarms** – Provides counts of the alarms for this agent, separated by severity.
- **Last Stat Reset** – Time when statistic counters for the background writer were last reset.
- **Checkpoint Timeout** – Maximum amount of time between automatic WAL checkpoint writes. If checkpoint segment limit is not reached before this time, a checkpoint will be triggered.
- **BGWriter Delay** – Delay between activity rounds for the background writer.
- **BGWriter LRU Max Pages** – Maximum number of pages written per activity round.
- **BGWriter LRU Multiplier** –Multiplier used to calculate the number of dirty buffers written for a given activity round

Graphs

- **Total Buffers Allocated** – Number of buffers allocated for use by the background writer.
- **Buffer Write Comparison** – Compares number of buffers written by a backend vs. being written by the background writer.
- **Buffer Write Method Pct** – Compares percent of buffers written by a backend vs. percent of buffers written during normal checkpoints.
- **% of Checkpoints Required** - Percent of checkpoints executed that were required. Low percent indicates system has not been active enough to reach WAL segment limit before reaching the checkpoint timeout limit.
- **Checkpoint Interval** – Average elapsed time between checkpoints.
- **Total Written Bytes Rate** – Rate of total data written into Write Ahead Log.

- **Avg Checkpoint Write Size** – Average amount of data written into WAL during checkpoints.
- **# of Time Reached Buffer Limit** – Number of times the background writer stopped a cleaning scan because it had written too many buffers.
- **Fsync Calls From Backends** – Compares number of times a backend had to execute its own fsync call (normally handled by the background writer even when the backend does its own write).

Databases

Overview

The Databases dashboard contains 2 sections separated by a horizontal splitter: the top section, which displays a row for each database as well as some key information, and the bottom section, which displays metric information related to the database selected in the top section as well as highlighting tables in that database with the highest or lowest value for a particular metric. The size of the two sections can be increased or decreased by dragging the horizontal splitter bar up or down. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Data

Top Section – DB Table

- **Health** – Shows the overall health of the database.
- **Name** – The name of the database.
- **Connections** – Number of connections to this database.
- **Tables** – Number of tables in the database.
- **Functions** – Number of functions in the database.
- **Workload** – A calculation of time spent reading and writing data file blocks by backends in this database in a sample period, divided by the sample period time. This represents the load on the server and may exceed 1 because of concurrent processing. This is not an estimation of resource usage or availability. Requires track_io_timing to be enabled, available only in v9.2+.

- **Blk Hit Rate** – Rate at which disk blocks are found already in the buffer cache, so that a read was not necessary.
- **Blk Read Rate** – Rate at which disk blocks are read from the database.
- **Blk Hit %** - Percentage where a table read is successfully sourced from buffer cache rather than needing physical disk I/O.
- **DB Size** – Disk size of this database.
- **Conflicts Rate** – Rate of queries cancelled due to conflicts with recovery in this database.
- **Tablespace** – Name of the default tablespace for objects created in this database.
- **Statements** – The number of statements gathered in the last sample for this database. This collection may be limited due to the # of Top Statements property in the Agent Properties or the pg_stat_statements.max setting on the PostgreSQL server.

Bottom Section – DB Metrics

- **Tuple Retrieval** – Rate of tuples retrieved with sequential scans (tup_returned_rate) vs. bitmap or simple scans (tup_fetched_rate).
    - **Tuple Modifications** – Rate of tuples modified by deletion, update, or insert operations.
    - **Transaction Results** – Rate of transactions finished with commits or rollback.
    - **Deadlocks** – Number of deadlocks detected in the database.
    - **Memory Reads** – Rate of memory blocks read from buffer cache vs. physical disk.
    - **Memory Operation Time** – Time spent reading or writing data file blocks by backends.
    - **DB Size** – Disk size of the database.
    - **Temp File Writes** – Number of temporary files created by queries in this database and the total data size.
    - **Conflict Issues** – Breakdown of cancelled queries due to conflict by reason.

Bottom Section – Table Highlights

- **Most Blocks Read** – Tables with the most disk blocks read during queries.
- **Lowest Mem Blk Hit %** - Tables with the lowest memory block hit %, i.e. tables where data is being read from disk rather than buffer cache.
- **Most Tuples Modified** – Tables most actively being modified.
- **Largest Size** – Tables with the largest disk size.

Actions

Database Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Connections**
    - Dwell - A popup time plot of connections.
    - Drilldown – Navigates to the Current Backends page filtered for selected database.
- **Tables** (drilldown) – Navigates to the Tables page for the selected database.
- **Functions** (drilldown) – Navigates to the Functions page for the selected database.
- **Blk Hit Rate** (dwell) – A popup time plot of the block hit rate.
- **Blk Read Rate** (dwell) – A popup time plot of the block read rate.
- **Blk Hit %** (dwell) – A popup time plot of the block hit percentage.
- **DB Size** (dwell) – A popup time plot of the database size.

- **Conflicts Rate** (dwell) – A popup time plot of the conflicts rate.
- **Tablespace**
    - Dwell - A popup table with the tablespace's health, OI, owner, and disk size.
    - Drilldown – Navigates to the Tablespaces page.
- **Statements** (drilldown) – Navigates to the Statements dashboard, filtered for statements only performed on the selected database.

Table Highlights
Clicking on an area of the graph reserved for displaying the metric value for a table will drilldown to the Tables page for that database with the selected table highlighted.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Vacuum + Analyze** – Perform a VACUUM ANALYZE operation on the database.
- **Vacuum** – Perform a VACUUM operation on the database.
- **Analyze** – Perform an ANALYZE operation on the database.
- **Stat Reset** – Reset the statistic counts on the database.

Tables

Overview

The Tables page contains 2 sections separated by a horizontal splitter: the top section, which displays a row for each table as well as some key information, and the bottom section, which displays metric information related to the table selected in the top section. The size of the two sections can be increased or decreased by dragging the horizontal splitter bar up or down. In the action panel on the right pane, the DB Selector allows you to switch between databases or show tables from all databases.

Data

Top Section – Tables Table

- **Health** – Shows the overall health of the table.
- **Name** – The name of the table.
- **DB** – The database that contains the table.
- **Schema** – The schema that contains the table.
- **Tuples** – The number of tuples in the table.
- **Indexes** – The number of indexes in the table.
- **Memory Hit %** - Percentage where a read from the table or any related resource is successfully sourced from buffer cache rather than needing physical disk I/O.
- **Idx Mem Hit %** – Percentage where a read from the table indexes is successfully sourced from buffer cache rather than needing physical disk I/O.
- **Idx/Seq Scan Tuple %** – Percentage of tuples returned from index vs. sequential scans.
- **HOT Update %** – Percentage of rows updated by HOT.

- **Idx Size** – Disk size of all index tables associated with the table**.**
- **Table Size** – Disk size of the table.
- **Idx/Table Size** – Ratio of index size to table size.
- **Owner** – Owner of the table.
- **Tablespace** – Tablespace where table data is stored.

Bottom Section – Table Metrics

- **Tuple Modifications Rates** – Rate of tuples modified by deletion, update, or insert operations.
- **Tuple Modification Pcts** – Distribution of modification operations affecting table tuples.
- **Live/Dead Tuples** – Counts of live and dead tuples in table memory. Dead tuples are deleted rows that have not yet been cleaned up.
- **Disk Sizes** – Disk sizes of table, table indexes, and total table size, including related resources.
- **Table Scan Rates** – Rates of index and sequential scans on this table.
- **Buffer Efficiency** – Breakdown of memory block hits by table resource.
- **Stat Calculations** – Counts of vacuum and analyze operations.
- **Last…Operation** – Timestamp of last operation on the table.

Actions

Table Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **DB** (drilldown) – Navigates to the Database page for the selected table.
- **Tuples** (dwell) – A popup time plot of the tuple count.
- **Indexes**
    - Dwell – A popup time plot of the index count.
    - Drilldown – Navigates to the indexes page.
- **Memory Hit %** (dwell or drilldown) - A popup time plot of the memory hit percentage.
- **Idx Mem Hit %** (dwell or drilldown) - A popup time plot of the index memory hit percentage.
- **Idx/Seq Scan Tuple %** (dwell or drilldown) - A popup time plot of the index/sequential scan tuple percent.
- **HOT Update %** (dwell or drilldown) - A popup time plot of the HOT update percent.
- **Idx Size** (dwell or drilldown) - A popup time plot of the index size.
- **Table Size** (dwell or drilldown) - A popup time plot of the table size.
- **Idx/Table Size** (dwell or drilldown) - A popup time plot of the index/table size.
- **Tablespace**
    - Dwell - A popup table with the tablespace's health, OI, owner, and disk size.
    - Drilldown – Navigates to the Tablespaces page.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Vacuum + Analyze** – Perform a VACUUM ANALYZE operation on the table.

- **Vacuum** – Perform a VACUUM operation on the table.
- **Analyze** – Perform an ANALYZE operation on the table.
- **Stat Reset** – Reset the statistic counts on the table.

Indexes

Overview

The Indexes page displays key information on indexes for the selected database table as well as providing comparisons between multiple table indexes in the four graphs at the top of the page.

Data

Top Section – Index Comparisons

- **Indexes v Table Size Comparison** – Compares disk size of table vs. combined table indexes.
- **Index sizes** – Compares disk sizes of table indexes.
- **Memory Hit %** - Compares memory hit percentage of indexes.
- **Average Tuples Returned** – Average number of tuples returned per query.

Bottom Section – Index Table

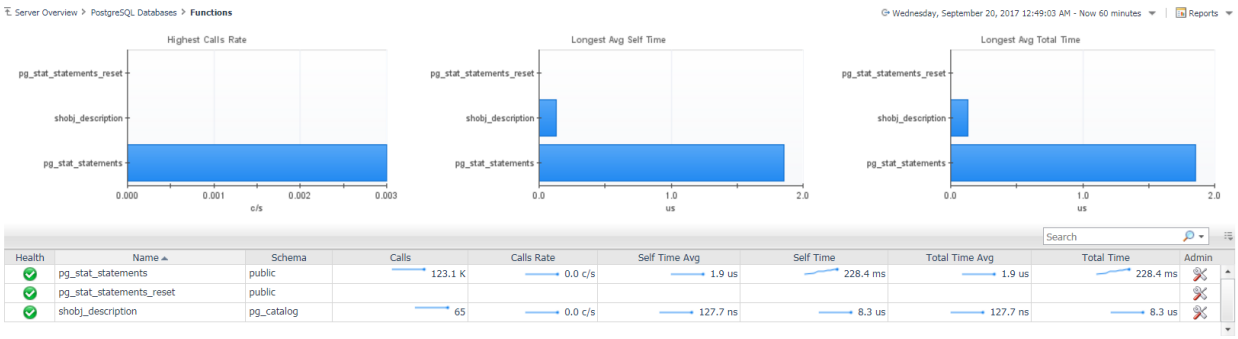
- **Health** – Shows the overall health of the index.
- **Name** – The name of the index.
- **Size** – Disk size of the index.
- **isPrimary** – Whether index is the primary index in the table.
- **isUnique** – Whether index values must be unique.
- **isValid** – Whether index is configured correctly and can be used as an index.
- **Attributes** – The number of columns used in the index.
- **Avg Tuples** – Average number of tuples returned from an index scan.
- **Memory Hit %** - Percentage where a read from the index is successfully sourced from buffer cache rather than needing physical disk I/O.
- **Hit Rate** – Rate of buffer hits in this index.
- **Scan Rate** – Rate of index scans initiated on the index.
- **Tuple Read Rate** – Number of index entries returned by scans on the index.
- **Tuple Fetch Rate** – Number of live table rows fetched by simple index scans using the index.

Actions

Index Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Size** (dwell) - A popup time plot of the index size.
- **Attributes** (dwell) – A popup table of the columns used in the index.
- **Avg Tuples** – A popup time plot of the average number of tuples returned.
- **Memory Hit %** - A popup time plot of the memory hit percentage.
- **Hit Rate** – A popup time plot of the hit rate.
- **Scan Rate** – A popup time plot of the scan rate.
- **Tuple Read Rate** – A popup time plot of the tuple read rate.
- **Tuple Fetch Rate** – A popup time plot of the tuple fetch rate.

Functions



Overview

The Functions page displays statistics on functions for the selected database as well as providing comparisons between functions in the three graphs at the top of the page. Tracking of function call counts and timing requires that track_functions is enabled.

Data

Top Section – Function Comparisons

- **Highest Calls Rate** – Compares call rates of most frequently called functions.
- **Longest Avg Self Time** – Compares self time for functions with highest average time.
- **Longest Avg Total Time** – Compares total time for functions with highest average total time.

Bottom Section – Functions Table

- **Health** – Shows the overall health of the function.
- **OID** – OID of the function.
- **Name** – The name of the function.
- **Schema** – Schema that owns the function.
- **Calls** – Number of calls to the function.
- **Calls Rate** – Rate at which the function is called.
- **Self Time Avg** – Average time spent in the function, not including calls to other functions.
- **Self Time** – Total time spent in the function, not including calls to other functions.
- **Total Time Avg** – Average time spent in the function, including calls to other functions.
- **Total Time** – Total time spent in the function, including calls to other functions.

Actions

Function Table

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Calls** – A popup time plot of the number of calls.
- **Calls Rate** – A popup time plot of the calls rate.
- **Self Time Avg** – A popup time plot of the average self time.
- **Self Time** – A popup time plot of the self time.
- **Total Time Avg** – A popup time plot of the average total time.

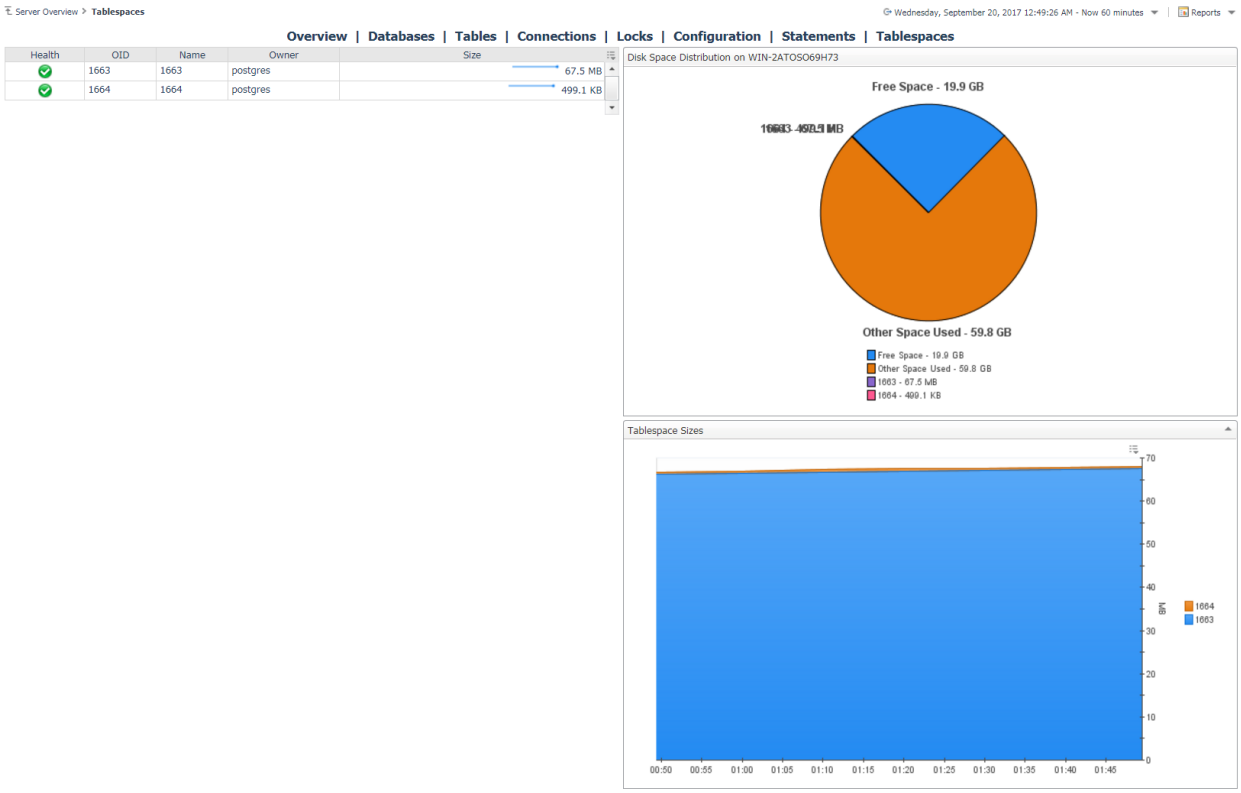
- **Total Time** – A popup time plot of the total time.

Admin Actions

If your user account has the PostgreSQL Administrator role and the PostgreSQL user provided to the agent has the appropriate permissions, you can perform administrative actions on the server from the dashboard. Click the icon in the Admin table column to perform one of the following actions:

- **Stat Reset** – Resets statistics for the selected function.

Tablespaces



Overview

The Tablespaces page shows tablespaces used by the selected PostgreSQL server. The table on the left displays a row for each tablespace while the graphs on the right show a visual distribution of disk spaze. In the action panel on the right pane, the PostgreSQL Agent Selector allows you to switch between PostgreSQL servers.

Data

Table

- **Health** – Shows the overall health of the tablespace.
- **OID** – OID of the tablespace.
- **Name** – Name of the tablespace.
- **Owner** – Owner of the tablespace.
- **Size** – Disk size of the tablespace.

Graphs

- **Disk Space Distribution** – Shows the distribution of disk space on the host between tablespaces, other used space, and free space. If host storage information is not being collected, only tablespace sizes are shown.
- **Tablespace Sizes** – Compares tablespace sizes to each other and shows combined total space used.

Actions

- **Health** (dwell or drilldown) - Shows a breakdown of child objects in a warning, critical, or fatal state.
- **Size** (dwell) – A popup time plot of the disk size.

PostgreSQL Reports

PostgreSQL Executive Summary – Executive summary of PostgreSQL instance with high-level information including connection, workload, availability, and performance metrics, as well as top 10 alarms by severity.

PostgreSQL Full Tables Report - Displays tables in PostgreSQL server that meet conditions specified in the Tables Report section of the Agent Properties. For more information, refer to the Agent Properties section of the Foglight for PostgreSQL guide.

PostgreSQL Health Check – Report showing health indicators for the PostgreSQL instance, including backends, workload, system, and performance metrics, top 10 statement digests by response time and top 10 alarms by severity.

PostgreSQL Server Configuration – Displays variables for the PostgreSQL instance with values at the start and end of the selected time range and when they were modified.

PostgreSQL Server Configuration Comparison – Compares current variable values for all PostgreSQL instances contained in a Service against a selected PostgreSQL instance (template server) to ensure configuration compliance. The report will optionally return all variables or just the variables with at least one difference in values.

PostgreSQL Top Servers by Connections – Displays top *X* PostgreSQL instances by the selected connection metric: availability percent, connection time, total backends, or percent used.

PostgreSQL Top Statements – Displays top *X* statement digests for a PostgreSQL instance by the selected metric: calls, average/total/max time, average rows, or shared blocks hit percent.

PostgreSQL Top Tables – Displays top *X* tables in a PostgreSQL instance by the selected table metric: tuples, indexes, memory hit percent, index memory hit percent, index size, or table size.

PostgreSQL Cartridge Rules

Approaching Connection Limit

Severity	Expression	Message

Fatal	def connUtil = (#total_backends#/scope.monitoringAgent.max_connections) * 100;
	
	if(connUtil>registry("connUtilFatal")){return true;}
		The connection utilization for this server is at @connUtil%. Your max_connections setting is @maxConnections. You may wish to increase this setting in order to ensure there are enough connections available.

Critical	def connUtil = (#total_backends#/scope.monitoringAgent.max_connections) * 100;
	
	if(connUtil>registry("connUtilCrit")){return true;}
		The connection utilization for this server is at @connUtil%. Your max_connections setting is @maxConnections. You may wish to increase this setting in order to ensure there are enough connections available.

Warning	def connUtil = (#total_backends#/scope.monitoringAgent.max_connections) * 100;
	
	if(connUtil>registry("connUtilWarn")){return true;}
		The connection utilization for this server is at @connUtil%. Your max_connections setting is @maxConnections. You may wish to increase this setting in order to ensure there are enough connections available.

Default Registry Variables: connUtilWarn = 85%, connUtilCrit = 95%, connUtilFatal=99%

Backend Write Percentage High

Severity	Expression	Message
Fatal	#backend_write_pct#>registry("backendWriteFatal")	A high backend write percentage means that backends are writing their own WAL buffers to disk rather than waiting for a checkpoint to do this. This may happen normally if your application performs many bulk insert statements. Otherwise, you may need to increase your background writer activity or increase your WAL Buffer allocation size.
Critical	#backend_write_pct#>registry("backendWriteCrit")	A high backend write percentage means that backends are writing their own WAL buffers to disk rather than waiting for a checkpoint to do this. This may happen normally if your application performs many bulk insert statements. Otherwise, you may need to increase your background writer activity or increase your WAL Buffer allocation size.
Warning	#backend_write_pct#>registry("backendWriteWarn")	A high backend write percentage means that backends are writing their own WAL buffers to disk rather than waiting for a checkpoint to do this. This may happen normally if your application performs many bulk insert

		statements. Otherwise, you may need to increase your background writer activity or increase your WAL Buffer allocation size.

Default Registry Variables: backendWriteWarn = 60%, backendWriteCrit = 75%, backendWriteFatal = 90%

DB Running Out of Disk Space

Severity	Expression	Message

Fatal	def ds = server.DataService;
	
	def free;
	try{
	free =
	ds.retrieveLatestValue(scope.monitoredHost.storage,"spaceAvailable").value.avg;
	}
	catch(Exception e){return;}
	if(free==null){return;} //Must be collected
	
	def endTime = new java.sql.Timestamp(System.currentTimeMillis());
	def startTime = endTime.minus(7).toTimestamp();
	
	def metricStart = ds.retrieveEarliestTime(scope,"tablespace_growth_rate");
	
	if(metricStart.after(startTime)){return;} //Metric must be collected for longer than the start time for a representative average
	
	def growthRate =
	ds.retrieveAggregate(scope,"tablespace_growth_rate",startTime,endTime).value.avg;
	if(growthRate<=0){return;}
	
	//Convert to expected daily growth in MB
	def dayGrowth = (growthRate*60*60*24)/(1024*1024);
	
	//Divide free space by day growth
	def daysToFull = free / dayGrowth;
	
	if(daysToFull < registry("dbCapacityFatal")){return true;}
		If the average growth of the tablespaces of this database server over the last week continues at its current rate, it is predicted that the DB host will run out of disk space in @daysToFull day(s).

Critical	def ds = server.DataService;
	
	def free;
	try{
	free =
		If the average growth of the tablespaces of this database server over the last week continues at its current rate, it is

	ds.retrieveLatestValue(scope.monitoredHost.storage,"spaceAvailable").value.avg;
	}
	catch(Exception e){return;}
	if(free==null){return;} //Must be collected
	
	def endTime = new java.sql.Timestamp(System.currentTimeMillis());
	def startTime = endTime.minus(7).toTimestamp();
	
	def metricStart = ds.retrieveEarliestTime(scope,"tablespace_growth_rate");
	
	if(metricStart.after(startTime)){return;} //Metric must be collected for longer than the start time for a representative average
	
	def growthRate = ds.retrieveAggregate(scope,"tablespace_growth_rate",startTime, endTime).value.avg;
	if(growthRate<=0){return;}
	
	//Convert to expected daily growth in MB
	def dayGrowth = (growthRate*60*60*24)/(1024*1024);
	
	//Divide free space by day growth
	def daysToFull = free / dayGrowth;
	
	if(daysToFull < registry("dbCapacityCrit")){return true;}
		predicted that the DB host will run out of disk space in @daysToFull day(s).

Warning	def ds = server.DataService;
	
	def free;
	try{
	free = ds.retrieveLatestValue(scope.monitoredHost.storage,"spaceAvailable").value.avg;
	}
	catch(Exception e){return;}
	if(free==null){return;} //Must be collected
	
	def endTime = new java.sql.Timestamp(System.currentTimeMillis());
	def startTime = endTime.minus(7).toTimestamp();
	
	def metricStart = ds.retrieveEarliestTime(scope,"tablespace_growth_rate");
	
	if(metricStart.after(startTime)){return;} //Metric must be collected
		If the average growth of the tablespaces of this database server over the last week continues at its current rate, it is predicted that the DB host will run out of disk space in @daysToFull day(s).

	for longer than the start time for a representative average
	
	def growthRate = ds.retrieveAggregate(scope,"tablespace_growth_rate",startTime, endTime).value.avg;
	if(growthRate<=0){return;}
	
	//Convert to expected daily growth in MB
	def dayGrowth = (growthRate*60*60*24)/(1024*1024);
	
	//Divide free space by day growth
	def daysToFull = free / dayGrowth;
	
	if(daysToFull < registry("dbCapacityWarn")){return true;}	

Default Registry Variables: dbCapacityWarn = 60 days, dbCapacityCrit = 30 days, dbCapacityFatal = 15 days

High Percentage of Queries Waiting

Severity	Expression	Message

Fatal	def conList = #Current_Backends#;
	
	int total = #Current_Backends#.size()
	if(total==0){return false;}
	int waitNum = 0;
	for(conItem in conList)
	{
	if(conItem.waiting=="t"){waitNum++;}
	}
	
	def waitPct = 100*(waitNum/total);
	if(waitPct>registry("queryWaitingFatal")){return true;}
		In the last 3 out of 5 evaluations, the percentage of queries in a waiting state have exceeded this severity threshold. The percentage of queries in a waiting state when this alarm was triggered is @waitPct%.

Critical	def conList = #Current_Backends#;
	
	int total = #Current_Backends#.size()
	if(total==0){return false;}
	int waitNum = 0;
	for(conItem in conList)
	{
	if(conItem.waiting=="t"){waitNum++;}
	}
		In the last 3 out of 5 evaluations, the percentage of queries in a waiting state have exceeded this severity threshold. The percentage of queries in a waiting state when this alarm was triggered is @waitPct%.

	def waitPct = 100*(waitNum/total); if(waitPct>registry("queryWaitingCrit")){return true;}	

Warning	def conList = #Current_Backends#;
	
	int total = #Current_Backends#.size()
	if(total==0){return false;}
	int waitNum = 0;
	for(conItem in conList)
	{
	if(conItem.waiting=="t"){waitNum++;}
	}
	
	def waitPct = 100*(waitNum/total);
	if(waitPct>registry("queryWaitingWarn")){return true;}
		In the last 3 out of 5 evaluations, the percentage of queries in a waiting state have exceeded this severity threshold. The percentage of queries in a waiting state when this alarm was triggered is @waitPct%.

Default Registry Variables: queryWaitingWarn = 10%, queryWaitingCrit = 20%, queryWaitingFatal = 30%

Index Bloat

Severity	Expression	Message
Fatal	#idx_size_pct#>registry("indexBloatFatal")	The index size for this table is currently @idxPct% of the total table size. You may want to ensure the index table is being vacuumed in order to compact its size.
Critical	#idx_size_pct#>registry("indexBloatCrit")	The index size for this table is currently @idxPct% of the total table size. You may want to ensure the index table is being vacuumed in order to compact its size.
Warning	#idx_size_pct#>registry("indexBloatWarn")	The index size for this table is currently @idxPct% of the total table size. You may want to ensure the index table is being vacuumed in order to compact its size.

Default Registry Variables: indexBloatWarn = 20%, indexBloatCrit = 25%, indexBloatFatal = 40%

Low Buffer Hit Percentage – Database

Severity	Expression	Message
Fatal	if(#blks_hit_pct#<registry("bufferHitFatal")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more

		system resources. The current buffer hit percentage is @hitPct%
Critical	if(#blks_hit_pct#<registry("bufferHitCrit")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%
Warning	if(#blks_hit_pct#<registry("bufferHitCrit")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%

Default Registry Variables: bufferHitWarn = 75%, bufferHitCrit = 50%, bufferHitFatal = 25%

Low Buffer Hit Percentage – Table

Severity	Expression	Message
Fatal	if(#total_blks_hit_pct#<registry("bufferHitFatal")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%
Critical	if(#total_blks_hit_pct#<registry("bufferHitCrit")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%
Warning	if(#total_blks_hit_pct#<registry("bufferHitWarn")){return true;}	Buffer hit percentage for this object has been low for the last three consecutive evaluations. A higher hit percentage means that tuples are being returned from pages in memory rather than being read from disk, which consumes more system resources. The current buffer hit percentage is @hitPct%

Default Registry Variables: bufferHitWarn = 75%, bufferHitCrit = 50%, bufferHitFatal = 25%

Low Percent of Checkpoints Required

Severity	Expression	Message
Critical	#checkpoints_req_pct#<registry("checkReqCrit")	A low percentage of checkpoints that are being performed are required, i.e. the checkpoint_timeout limit has been reached before the checkpoint_segments limit. This may indicate that your settings should be optimized for the level of activity normally seen on this DB server. A lower segment limit can also mean shorter crash recovery time if that is a concern.
Warning	#checkpoints_req_pct#<registry("checkReqWarn")	A low percentage of checkpoints that are being performed are required, i.e. the checkpoint_timeout limit has been reached before the checkpoint_segments limit. This may indicate that your settings should be optimized for the level of activity normally seen on this DB server. A lower segment limit can also mean shorter crash recovery time if that is a concern.

Default Registry Variables: checkReqWarn = 35%, checkReqCrit = 15%

PostgreSQL Server Availability

Severity	Expression	Message
Fatal	if(#availability#.equals("Not Available")){return true;}	if(#availability#.equals("Not Available")){return true;}

Potential Deadlock Issue

Severity	Expression	Message

Critical	def locksWaiting = #locks_waiting#;
	def timeout = scope.deadlock_timeout/1000;
	
	if(locksWaiting==0){return false;}
	
	for(def lockRow in scope.Current_Locks.current.value)
	{
		You may have a potential deadlock issue. Currently, there are @lockNum locks that have not been granted and are

	if(lockRow.granted=="f"){
	if(lockRow.lock_age>timeout){return true;}
	}def locksWaiting = #locks_waiting#;
	def timeout = scope.deadlock_timeout/1000;
	
	if(locksWaiting==0){return false;}
	
	for(def lockRow in scope.Current_Locks.current.value)
	{
	if(lockRow.granted=="f"){
	if(lockRow.lock_age>timeout){return true;}
	}
	}
		older than the deadlock timeout setting of @timeout milliseconds. The oldest waiting lock age is @maxAge seconds and the average age is @avgAge seconds.

Replication Lag

Severity	Expression	Message
Critical	#standby_log_diff#>=registry("repLagCrit")	The difference in WAL segments between your primary's sent_location and standby server's replay_location has exceeded this threshold. If the standby server continues to fall behind, this may cause problems, especially if the primary server goes down in a high-availability situation.
Warning	#standby_log_diff#>=registry("repLagWarn")	The difference in WAL segments between your primary's sent_location and standby server's replay_location has exceeded this threshold. If the standby server continues to fall behind, this may cause problems, especially if the primary server goes down in a high-availability situation.

Default Registry Variables: repLagWarn = 5, repLagCrit = 10

Slow Connection

Severity	Expression	Message
Fatal	#connection_time#>registry("connTimeFatal")	The agent's last two connections to this PostgreSQL server have exceeded this rule's threshold for connection time. The latest connection time was @connTime seconds.
Critical	#connection_time#>registry("connTimeCrit")	The agent's last two connections to this PostgreSQL server have exceeded this rule's threshold for connection time. The latest connection time was @connTime seconds.
Warning	#connection_time#>registry("connTimeWarn")	The agent's last two connections to this PostgreSQL server have exceeded this rule's threshold for

		connection time. The latest connection time was @connTime seconds.

Default Registry Variables: connTimeWarn = 1.5, connTimeCrit = 3, connTimeFatal = 5