Control Center

1.0 — Last update: 30 October 2024

dbWatch AS

Copyright © 2024 dbWatch AS

Table of Contents

About dbWatch Control Center 16
- Release notes 17
- Find your version 46
- Platforms supported 48

Getting Started 51
- Installing on Windows 52
- Installing on Linux 62
  - Console installation on Ubuntu 64
- Initial Domain Setup 66

Product Overview 71
- Architecture 72
  - Example architecture designs 75
  - Cloud router 79
    - Cloud router security 82
- File Structure 84
- Network and communications 86
- Database operations philosophy 88
- Monitor dashboards and overview screens 90

Installing and using premium packages 91
- SQL Performance package 92
  - Installation of SQL Performance package 93
  - Using SQL Performance package on SQL Server 96
    - Integration with SSMS 103
  - Using SQL Performance package on Oracle 105
  - Using SQL Performance package on PostgreSQL 111
  - Using SQL Performance package on MySQL 117
  - Repository management and monitoring 123
    - Troubleshooting issues in SQL Performance package 127
- Security and Compliance package 129
  - Installation of the Security and Compliance package 130
  - Security and Compliance package on SQL Server 134

Using Control Center 143
- The modules of Control Center 147
- Disabling and Enabling Views 154
- Instance management 156
  - Instance Configuration 158
    - JDBC Properties 162
    - Monitoring Engine 164
  - Adding a SQLServer Instance 165
    - Using OS Authentication 171
    - Adding SQLServer instances with kerberos 174

Adding an Oracle Instance ... 180
Adding a PostgreSQL Instance ... 186
Adding a MySQL Instance ... 191
Bulk installing instances ... 197
Groups Configuration ... 201
Changing Instance Group ... 205
Service groups ... 207
Domain Configuration ... 210
Adding or changing license ... 214
Managing nodes ... 216
Managing roles ... 217
Managing users ... 218
Creating a user ... 220
User integration with Active Directory ... 222
2 factor authentication ... 225
Managing security groups ... 227
Privileges and actions map ... 229
Upgrade dbWatch ... 232
Monitoring ... 234
Tree navigation ... 237
Instance details ... 243
Pause ... 248
Job details ... 252
Job menus ... 257
Job concepts ... 258
Schedule ... 259
Acknowledgement system ... 261
Versioning ... 262
Status ... 263
Report ... 264
Parameters ... 266
Notes ... 267
Job types ... 271
Engine jobs ... 272
No-engine jobs ... 273
Control Center Jobs ... 274
Jobs grouped by platform ... 277
Jobs for Oracle ... 278
Alert log check ... 284
Alert log check (10g) ... 285
Alert log check (9i) ... 287
Alert log check (8i) ... 289
Alert log check (java) ... 290
Alert log check (java for 10g) ... 292
Archive status check ... 294

Backup log check (old style 10g) 295
Backup log check (old style 9i) 297
Backup log check (old style 8i) 299
Dataguard Archive Status 301
Data Guard Archive Status Check (10g & 9i) 302
Database link check 303
Instance alert log 304
Database uptime 305
DBA jobs check 306
DBMS uptime 307
Export log Check (‘old style’ backups) 308
File status check 310
Index status check 311
Invalid objects check 312
Job scheduling check 313
Listener log check 314
Listener log check (10g) 315
Listener log check (11g) 316
Listener log check (9i) 317
Listener log check (8i) 318
Listener log check (java) 319
Listener log check (java 10g) 321
Listener status check 322
Max datafiles check 323
Database mount state 324
Datafile status 325
Max datafiles 326
RMAN backup status (noschema) 327
Password expire 328
PDB status 329
RMAN backup status (10.1) 330
RMAN backup status (10.2) 331
RMAN backup status (11) 332
RMAN backup status (8) 333
RMAN backup status (9) 334
RMAN archivelog backup status 335
RMAN backup status 336
RMAN incremental backup status 338
Test alert db 339
Test alert 340
TNSping check 341
ASM disk statistics 342
ASM diskgroup space 343
Aud$ table size check 344
Autoextensible data files 345

Disk space check 346
Flash Recovery Area Usage 347
Flash Recovery Area Usage (10g) 348
Free extents 349
Max processes 350
ASM diskgroup space (noschema) 351
Extent fragmentation 352
Flash recovery space usage (noschema) 353
Max processes (noschema) 354
SYS.AUD size 355
Tablespace free space (noschema) 356
Segment size collector (all segments -- aggregate) 357
Segment size collector (all segments -- aggregate 9i) 358
Segment size collector (large segments -- detail) 359
Segment size status for old style tablespaces 360
Temporary tablespace free space check 361
Tablespace free space check 362
Blocking detector for RAC 363
Memory session load for RAC 364
MV Refresh Group(s) 365
Dataguard apply time 366
Dataguard archive sequence apply lag 367
Dataguard standby archivelog gap 368
Dataguard standby archivelog gap 369
Dataguard startup time 370
Dataguard transport time 371
RAC instances status 372
Session load RAC 373
Snapshot Log(s) rows count 374
Snapshot Log(s) size 375
Data Guard role switch 376
Restore uptodate status 377
Framework 378
Analyze tables 379
Delete SYS.AUD$ data 380
Rebuild indexes 381
Statistics migation 382
Blocking detector 383
Buffer cache statistics 384
CPU load 385
Database network statistics (SQL*NET) 386
Test DML performance 387
File IO statistics 388
Instance memory statistics 389
Latch statistics 390

Long running queries ... 391
Open cursors ... 392
Disk read statistics ... 393
Redo statistics ... 394
Redo statistics (8i) ... 395
Session load ... 396
Session load (8i and 9i) ... 397
SQL waits ... 398
Table statistics check ... 399
Top user memory usage ... 400
Undo statistics ... 401
User memory statistics ... 402
Wait statistics ... 403
SQL statistics ... 404
SQL statistics (RDS) ... 405
RMAN recovery area backup status ... 406
Tablespace in BEGIN BACKUP mode ... 407
Applied archive log gap status ... 408
SQL statistics (CDB) ... 409
SQL statistics (PDB) ... 411
Jobs for MS SQL Server ... 413
Agent Jobs Check ... 421
Agent Jobs Check (MS2000) ... 422
SQL Server Agent status ... 423
Database backup (MS2000) ... 424
Database backup (system dbs) ... 425
Database backup ... 426
Database backup (SIMPLE) ... 427
Database backup (system databases) ... 428
Database log backup ... 429
Collation check ... 430
Check database recovery mode ... 431
Database status ... 432
Database status (MS2000) ... 433
Database server uptime ... 434
Database Server uptime (MS2000) ... 435
Database server uptime (MS2005) ... 436
Instance error log ... 437
Instance error log (AWS) ... 438
Instance status ... 439
Database backup (noschema) ... 440
Database log backup (noschema) ... 441
Database status (noschema) ... 442
Missing database backup (noschema) ... 443
Missing database log backup (noschema) ... 444

Program status ... 445
Report Server status ... 446
Restricted Enterprise edition features ... 447
Restricted Enterprise edition features (MS2000) ... 448
Restricted Enterprise edition features (MS2005) ... 449
Auto growth event collector ... 450
Data file size check ... 451
Data file size check (MS2000) ... 452
Database growth rate (aggregated) ... 453
Database growth rate (aggregated) (MS2000) ... 454
Database growth rate (detailed) ... 455
Database growth rate (detailed) (MS2000) ... 456
Database disk capacity ... 457
Database disk space usage ... 458
Databases NOT IN USE collector ... 459
Disk space check ... 460
Instance error log file size check ... 461
Filegroups growth rate ... 462
Filegroups growth rate (MS2000) ... 463
Data file size check (noschema) ... 464
Database disk capacity (noschema) ... 465
Disk space check (noschema) ... 466
Transaction log space usage (noschema) ... 467
Object size collector (all databases) ... 468
Temporary database space usage ... 469
Transaction log size check ... 470
Transaction log size check (MS2000) ... 471
Transaction log space usage ... 472
Transaction log space usage (MS2000 / MS2005 / MS2008) ... 473
Truncate transaction log ... 474
Version store space usage (tempdb) ... 475
Database mirroring ... 476
Failover cluster host switch ... 477
Health state ... 478
Log shipping monitor (primary) ... 479
Log shipping monitor (secondary) ... 480
Member state ... 481
AlwaysOn database backup alert ... 482
AlwaysOn transaction log backup alert ... 483
Replica states ... 484
Replication status ... 485
MS SQL Server patch status ... 486
Database statistics ... 487
dbWatch engine alert ... 488
dbWatch Server activity alert ... 489

Autogrow settings 490
Framework 491
Framework (MS2000) 492
Backup All databases 493
Backup All transaction logs 494
Check database and server principal mapping 495
Cleanup MSDB history tables 496
Cycle error log 497
DBCC CHECKDB 498
DBCC UPDATEUSAGE 499
External fragmentation (all databases) 500
Internal fragmentation check 501
SQL Server performance counters 502
Rebuild indexes 503
Rebuild indexes in table 504
Reorganize indexes 505
Reorganize indexes in table 506
Shrink transaction logs 507
Suspect pages 508
Update index statistics 509
Update statistics 510
Statistics migation 511
Blocking detector 512
Blocking statistics 513
Cached query statistics 514
Data cache memory usage 515
Data cache memory usage (MS2005 / MS2008) 516
Data hit ratio 517
Database session load 518
Test DML-DDL performance 519
File IO statistics 520
High activity monitor 521
High activity monitor (MS2005) 522
Index usage statistics (all databases) 523
Instance memory check 524
Instance memory usage 525
Instance memory usage (MS2005 / MS2008) 526
Lazy writer and checkpoint statistics 527
Memory objects statistics 528
Blocking detector (noschema) 529
Instance memory check (noschema) 530
Server memory statistics 531
Sessions per database 532
Session load 533
Transactions log flushed bytes load 534

Transactions log flushed bytes load (MS2000) 535
Transactions load 536
Wait statistics 537
SQL statistics 538
Query store status 539
Jobs for PostgreSQL 540
Backup check – pg_dump 542
Backup check – WAL 543
DBMS uptime 544
dbWatch engine alert 545
Log size statistics 546
Schema growth collector 547
Vacuum alert 548
Schema growth and information 549
Schema growth statistics 550
Tablespace growth and information 551
BDR Replication lag 552
Replication delay alert 553
Replication slave client alert 554
Backup job – pg_dump_all (ssh) 555
Framework 556
Analyze tables 557
Vacuum tables 558
Buffer cache reads per database 559
Buffer cache statistics 560
Disk block hitrate 561
Disk block reads per database 562
Test DML performance 563
Index block hitrate 564
Locks held and statistics 565
Table and index scan collector 566
Session load 567
Table and index scan statistics 568
Transaction statistics 569
SQL statistics 570
SQL statistics 571
Jobs for MySQL 572
DBMS uptime 574
MySQL aborted connects 575
MySQL max connections 576
MySQL mysqld process 577
Database growth rate (aggregated) 578
Database growth rate (detailed) 579
Innodb buffer pool check 580
Innodb cluster status 581

NDB data memory usage check ... 582
NDB data node status ... 583
Innodb cluster switch ... 584
Replica count ... 585
Replica delay ... 586
Replica state ... 587
Binlog cache check ... 588
Key buffer check ... 589
Database load ... 590
Lock statistics ... 591
Memory setup ... 592
Network traffic ... 593
Query cache hitrate ... 594
Session load ... 595
Thread cache hitrate ... 596
Temporary table check ... 597
Jobs for MariaDB ... 598
DBMS uptime ... 599
DBMS uptime (old) ... 600
Binlog cache check ... 601
Binlog cache check (old) ... 602
Database growth rate (aggregated) ... 603
Database growth rate (detailed) ... 604
MariaDB Index Redundancy Checker ... 605
Innodb buffer pool check ... 606
Innodb buffer pool check (old) ... 607
Key buffer check ... 608
Key buffer check (old) ... 609
Database load ... 610
Database load (old) ... 611
Lock statistics ... 612
Lock statistics (old) ... 613
Memory setup ... 614
Memory setup (old) ... 615
NDB data memory usage check ... 616
NDB data node status ... 617
Network traffic ... 618
Network traffic (old) ... 619
Aborted connects alert ... 620
Maximum connections alert ... 621
MySQLd process alert ... 622
Query cache hitrate ... 623
Query cache hitrate (old) ... 624
Replica count ... 625
Replica delay ... 626

Replica state ... 627
Session load ... 628
Session load (old) ... 629
MariaDB Space Optimizer ... 630
Thread cache hitrate ... 631
Thread cache hitrate (old) ... 632
Temporary table check ... 633
Temporary table check (old) ... 634
Jobs for Sybase ... 635
Blocking detector ... 637
Database backup ... 638
Database space check ... 639
Database space check (12) ... 640
Database status ... 641
Data cache monitor ... 642
Data cache statistics ... 643
Database growth rate (detailed) ... 644
DBMS uptime ... 645
Disk activity monitor ... 646
Engine CPU monitor ... 647
Memory statistics ... 648
Procedure cache check ... 649
Procedure cache check (12) ... 650
Session load ... 651
System monitor collector ... 652
Temp cache statistics ... 653
Test DML-DDL performance ... 654
Transaction log space check ... 655
User connections check ... 656
User connections check (12) ... 657
Jobs grouped by category ... 658
Availability ... 659
Capacity ... 664
Cluster and replication ... 669
Compliance ... 672
Consolidation ... 673
Maintenance ... 674
Migration ... 676
Performance ... 677
Management ... 685
Management on SQL Server ... 687
dbWatch alerts ... 689
Configuration ... 691
Parameters ... 692
Advanced dynamic parameters ... 693

Advanced static parameters ... 694
Registry parameters ... 695
Uptime statistics ... 696
[Database name] (database server) ... 697
[Instance name] (instance) ... 699
Maintenance periods ... 700
SQL performance ... 702
Internal repository ... 703
Server activity ... 704
Blocking sessions ... 705
Blocked sessions history ... 706
Blocking sessions history ... 707
Locks ... 708
Active processes ... 709
Performance test ... 710
Sessions ... 711
Sessions history ... 713
Sessions performance ... 714
Disk and Memory usage ... 715
Backups ... 717
SQL Server Error logs ... 718
Databases ... 719
Security ... 720
SQL Server Agent ... 721
Operating System Manager ... 722
Audit ... 723
SQL Worksheet ... 724
Worksheet ... 725
Autodiscover ... 730
Discovering Database Instances ... 731
Defining Autodiscover Ports ... 735
Auto-discover with different default port ... 737
FDL console ... 740
Reporting ... 741
Generate Reports ... 742
Scheduled reports ... 747
Edit Report Templates ... 749

Customization ... 756
Importing resources ... 757
Monitoring ... 759
Editor for engine jobs ... 760
Properties ... 763
Metadata ... 764
Internal objects ... 765
External dependencies ... 766

Configuration parameters ... 767
Installation variables ... 768
Pre implementation ... 769
Implementation ... 770
Main procedure SQL Server ... 771
Post implementation ... 778
Upgrade ... 779
Report ... 780
Example engine jobs ... 789
Job templates ... 790
Customize View using FDL ... 798
No-engine jobs ... 803
Property System ... 810
Metadata properties ... 811
Dynamic properties ... 818
Javascript property ... 819
The property file format ... 821
Management ... 824
Formatting table results ... 825
Result types ... 826
Graphical ... 827
Stacked area chart ... 832
Line chart ... 834
Pie chart ... 836
Category chart ... 838
Category line chart ... 840
Item active ... 842
Reports ... 845
Extensions and 3rd party integrations ... 851
Email and SMS ... 852
Configuring email extension for Service Now ... 860
Web server ... 864
Web configuration ... 865
Dashboards ... 868
Web data export ... 871
Web dashboard editor (beta) ... 874
Anonymous webaccess ... 877
Supported Web Browsers for dbWatch Dashboard ... 878

Advanced Topics ... 879
FDL – Farm Data Language ... 880
Functions ... 892
add function ... 894
sub function ... 895
abs function ... 896
until function ... 897

groupby function ... 898
count function ... 899
sum function ... 900
avg function ... 901
max function ... 902
min function ... 903
upper function ... 904
lower function ... 905
trim function ... 906
concat function ... 907
capitalize function ... 908
distinct function ... 909
mkstring function ... 910
nicedb function ... 911
union function ... 912
intersect function ... 913
difference function ... 914
round function ... 915
div function ... 916
mult function ... 917
replace function ... 918
orderby function ... 919
todate function ... 920
empty function ... 921
nonempty function ... 922
tomillis function ... 923
toperiod function ... 924
containsall function ... 925
containsnone function ... 926
containssome function ... 927
Available properties ... 928
Properties on dbWatch Servers ... 929
Properties on Instances ... 935
Properties on Jobs ... 938
Properties on Internal ... 941
The FString format ... 942
Product Security ... 944
Product Description ... 946
Hardware Specifications ... 947
Operating System ... 948
Network Ports and Services ... 949
Sensitive Data Transmitted ... 950
Sensitive Data Stored ... 951
Certificate infrastructure ... 952
Crypto catalog ... 953

Network Controls 954
Encryption 955
Audit Logging 957
Remote Connectivity 958
Disclaimer 959
Node to Node communication 960
Auditing 962
Control Center Commandline 963
Setting up a CCC node 964
FDL with CCC 967
Add instance with CCC 969
Set metadata with CCC 971
Install jobs with CCC 972
Get instance configuration with CCC 973
Register instance configuration with CCC 974
Instance action script with CCC 975
Start/stop webservice with CCC 976
Exporting configuration from 12 to CC 978
Internal Control Center firewall 982

Troubleshooting Guide 986
Backup of dbWatch 987
Adding MySQL Instance with Super Privilege Error 988
Restore configuration backup 991
Issues and troubleshooting 993
Definition Mapping 994
Resetting dbWatch Control Center Admin Account 998
Fresh Installation of dbWatch Control Center 1001
Remote Support 1006

Additional Resources 1010
Database Farm Management Additional Resources 1011

Licensing terms 1013

About dbWatch Control Center

dbWatch Control Center is a highly scalable software solution that helps enterprises monitor and manage large database servers efficiently, providing total control over all database operations, performance, and resource usage. dbWatch Control Center will give you the complete overview you need to maximize efficiency and minimize the resources you spend on running all your databases.

It is a simple, customizable, lightweight, and easy-to-install database operations tool that provides monitoring, management, reporting, maintenance, security, and integration capabilities. It is a solution that supports cross-platform database servers such as **Oracle**,**Mircosoft SQL Server**, **MariaDB**, **Postgres**, **MySQL**, **MongoDB** and **Sybase** whether the environment is on-premise, cloud, or a hybrid of both.

dbWatch Control Center can generate reports for a single instance, any group of servers, or an entire database farm. These reports can follow a template or be customized to fit your organization's standards. dbWatch monitors instances in real time and compiles historical data for DBA to use in comparative analysis.

All views are under a single solution. From its global view, you can monitor all your instances and dig deeper into specific instances to find more information about them. And, as a user, these views are customizable through the use of Farm Data Language, a built-in query language.

dbWatch Control Center is unique with its combination of closed and open-source code, allowing end-users or external developers to create and extend the product. All code developed by dbWatch for running within the databases, such as monitoring jobs, reports, and management menus and dashboards, are open to read and change to suit customer needs. Development tools come integrated into the application.

To get started with Control Center, go to the section Getting started

For more information, please check out our YouTube channel with many videos covering the product.

Other Existing Documentations

If you are looking for dbWatch's online documentation for Enterprise Manager, you can proceed with the following link:
dbWatch Enterprise Manager Version 12.9

Release notes >>

Release notes

dbWatch Control Center (2024-10-07)

Download

Improvements

- MS SQL Server – Job “Database disk capacity”. New parameter “combine thresholds”. This parameter allows you to combine the conditions of both the “alarm threshold (percent)” and “alarm threshold (MB)” or “warning threshold (percent)” and “warning threshold (MB)” parameters. When set to “YES”, an alarm/warning will only be triggered if both the percentage and MB threshold values are exceeded simultaneously. This setting provides an additional level of control, ensuring that alarms/warnings are only activated when both thresholds are surpassed, minimizing false positives and unnecessary alerts.
- MS SQL Server – Management, added availability group (AG) column to database statistics view and new view for AlwaysOn databases

Bugfix

- MS SQL Server – Job “framework”. Older versions of framework could not be upgraded in version 2.8 (fix in version 2.81). Internal use only.
- Oracle – Job “Extent fragmentation”. Fix for property “length”.
- Oracle – Error when installing engine-jobs with status.

dbWatch Control Center (2024-09-18)

Download

New features

- DBW – Email lost connection reminder
- DBW – FDL todate has format option
- DBW – Installing/Uninstalling jobs now in notes
- DBW – Remote Support
- DBW – Custom Monitoring pause
- DBW – Custom Alerting pause
- DBW – FDL empty/nonempty functions
- Oracle – job “RMAN ARCHIVELOG backup status alert” which can be used if no dbWatch schema exists.
- MS SQL Server – job “Deadlocks event collector”. Collects deadlocks event statistics from Microsoft SQL Server Extended Event Log files. If configured, a warning or alarm can be triggered when a deadlock is detected.
- MS SQL Server – Management, new “Deadlocks history” view.
- Sybase – SQL performance module (LICENSED)
- MS SQL Server – new sub farm view, “tempdb space usage” in “Resource overview”

- Farm – new “Uptime statistics” farm view displaing uptime statistics and maintenance periods statistics.

Improvements

- DBW – Global report item active
- DBW – Global report multi column layout
- DBW – Extension info i separate log
- DBW – Timeout in property fdl steps
- Sybase – Better SQL error reporting
- DBW – Warning when installing/uninstalling jobs when template in use
- DBW – Added uptime to About dialog
- DBW – Network stability
- MS SQL Server – job “MS SQL Server patch status”, ignores “Support end date” and use only “Extended support end date” in Security and Compliance overview.
- MariaDB – Management, added new graphs in SQL Performance module (“Session load”,“Memory usage”, “InnoDB data cache”)
- Sybase – Improved “Test DML-DDL performance” job with multi test series.
- MS SQL Server – job “AG synchronization lag”, new parameter “ignore events”, this parameter makes it possible to “skip” a warning or an alarm (one or more times) during initial synchronization lag detection.
- MS SQL Server – job “SQL statistics” improved procedure execution time.
- MS SQL Server – job “Databases NOT IN USE collector” has two new parameters enabling collecting statistics for additional databases with higher activity.
- Oracle – job “Buffer cache statistics”, new parameter “graph time range”, determines how many days back from the current date the data should be visualized in the report graphs.
- Oracle – job “Framework”, checks and/or removes date records in dbwatch schema if the clock on the database server has been moved forward in time.
- MS SQL Server – job “Framework”, checks and/or removes date records in dbwatch schema if the clock on the database server has been moved forward in time.
- Oracle – job “Database uptime”, two new parameters enabling adding and removing maintenance periods.
- MS SQL Server – job “Database Server uptime”, two new parameters enabling adding and removing maintenance periods.
- MS SQL Server – job “High activity monitor”, new parameter “graph time range”, and improved local report.
- Oracle – job “CPU load”, Better calculation of CPU consumption.

Bugfix

- Sybase – handling of multiple result sets
- DBW – issues in dictionary save
- DBW – Management chart height issue
- DBW – Reset password file privilege issue
- PostgreSQL – Handling of escape characters in password
- DBW/Web – New dashboard issue
- DBW – Compatibility issue for newly imported global report
- DBW – Edit scheduled report, save bug when tree is filtered

- DBW – Rare Monitor hang bug
- Oracle – job “RMAN backup status alert”, wrong date sorting, and now checks only input type “DB FULL” and “DB INCR”.
- Sybase – job “Database growth rate”, report now showing all growth statistics with correct date format.
- Sybase – job “Framework” Upgrade with case sensitives collations fails.
- MS SQL Server – job “Transaction log size check” error: Divide by zero error encountered. Error line: 109. Pos: 04.
- MS SQL Server – job “Database Server uptime” error in report for case sensitive dbwatch database.
- MS SQL Server – job “MS SQL Server patch status”, empty “Details” column if newer patches installed.
- MS SQL Server – Management – node “Uptime statistics”, fix for case sensitive dbwatch database
- Oracle – job “Buffer cache statistics” Divide by zero error encountered. Bether exeption handling.
- MS SQL Server – job “Database Server uptime”, fix for case sensitiv dbwatch database.

dbWatch Control Center (2024-05-06)

Download

New features

- DBW – Editable glabal report name
- DBW – Global report – fdl parameter
- DBW – Http service management through CCC
- DBW – FDL tomillis function
- DBW – FDL set functions containsall/some/none
- DBW – FDL function toperiod
- MS SQL Server/Agresso module – “Agresso reports queue”, “Agresso help and temp tables”, “Agresso blob size collector”, “Agresso framework” (LICENSED)
- Oracle/Agresso module – “Agresso reports queue”, “Agresso help and temp tables”, “Agresso CON$ system table” (LICENSED)
- MS SQL Server – Management. UNIT4 module for Agresso application. (LICENSED)
- Sybase – job “Dump database”, runs “dump database” command. (BETA)

Bugfix

- Mariadb – job “Framework”. Fix for uninstall Framework job.
- Properties – fix for MariaDB properties “grant_all_privs” and “with_grant_option” when command “show grants for current_user()” returns more than one row.
- Sybase – engine, fix for case sensitive database object names (fails when creating dbw_registerNewParameter procedure)
- Sybase – Management, “on-click” security administration fix.
- Oracle – job “AUDIT TABLE”, improved exception handling.
- DBW – Global report timeout handling
- DBW – Bugfix fdl mult function – preserve whole number
- DBW – Bugfix date rendering in reports

- DBW – Bugfix date ordering in graphs
- DBW – Bugfix jdbc handling in Sybase
- DBW – Bugfix in setting of JDBC driver properties

Improvements

- MS SQL Server – Management. Improved Blocking/Blocked session history views.
- MS SQL Server – job “Backup All transaction logs” collects information for every job execution.
- MS SQL Server – job “MS SQL Server patch status”, new cumulative update for SQL Server 2022 (CU12) and SQL Server 2019 (CU26).
- Sybase – job “Database backup” collects backup statistics into history table and checks if there are any missing statistics in master..monOpenDatabases view.
- DBW – Network stability
- DBW – Database related errors in seperate log file
- DBW – Pie chart – removed decimal if whole number
- DBW – Edit job template – can now delete multiple jobs
- DBW – Edit job template – only one category open at a time
- DBW – Sorting when editing multiple metadata entries
- DBW – Only one Connectivity dialog open at a time
- DBW – Test all button in Connectivity dialog
- DBW – Improved traceroute in connectivity dialog
- DBW – Improved loading speed of management
- DBW – FDL improvement for add/sub – preserve whole numbers

dbWatch Control Center (2024-03-11)

Important note

- Linux release is available as an apt repository, see Linux install guide
- All clients and servers need to be upgraded in a multi-server setup before the connection will resume due to network incompatibilites
- Only the server installation is available on Linux in this release (no graphical client or CCC scripting engine)

New features

- MariaDB – Now supported from version 10.2+. For version 10.1 jdbc driver version 2.7.12 must be used for connection insted of 3.3.3. For MariaDB 5.5 only monitoring without local database is supported.
- MariaDB – “SQL statistics” job, collects detailed information about SQL statements executed on the instance. Supported versions: 10.1+. (LICENSED)
- MariaDB – Management – SQL Performance views showing statistics gathered by “SQL statistics” job (LICENSED).
- MariaDB – Monitoring of Galera Cluster; Node state, Node count, Replica queues, Network staistics.
- Oracle/MS SQL Server – New global report template “Uptime statistics”.
- MySQL/MariaDB – New global report template “Health check report”

Bugfix

- DBW – global report download unstability issue
- Oracle – Management. Improved handling of PDB and CDB instances so that statistics are not mixed between them.
- MS SQL Server – “Health state” job, fix when using CS tempdb.
- MS SQL Server – “Autogrowth settings” job. In an earlier version of dbWatch, the job was initially named “Autogrow settings”, but it was subsequently renamed to “Autogrowth settings”. This change has unfortunately led to the confusion and errors.
- MySQL – Engine. Fixed error when upgrading jobs, and fix for jobs with status which appers without status (new framework versjon 1.32).
- MS SQL Server – “MS SQL Server patch status” job, fix for showing wrong cumulative updates version for SQL Server 2019 (CU23/CU24). Last patch for 2019 CU25 5th apr. 2024
- DBW – Fdl now handles escaping in quoted text.

Improvements

- MS SQL Server – “Update index statistics” job. 2 new parameters that make it possible to store statistics per database.
- MS SQL Server – “DBCC CHECKDB” job. Improved local report. If “max elap time” is exceeded during execution of the procedure, the report will contain statistics from previous executions until the last database has been checked.
- MS SQL Server – “Reorganize indexes” job. Improved local report. If “max elap time” is exceeded during execution of the procedure, the report will contain statistics from previous executions until the last database has been checked.
- Oracle – Management. Improved “kill session” menu, so that the session executing “alter system kill session” command does not try to terminate itself.
- MS SQL Server – “Database status” job. Collects statistics for periods where the state of the databases has been different from ONLINE. Two new parameters enable ignoring of databases in a none ONLINE state for a long period of time.
- MS SQL Server – Management. New menu in “Files information” table, showing Drive information (size, free space, used space).
- MySQL – “Maximum connections alert” job. Better feedback when the value of max_connections is not available.
- MS SQL Server – “Backup All databases” job. Two new parameters enable compression and creation of subdirectories for each database.
- MS SQL Server – “Backup All transaction logs” job. Two new parameters enable compression and creation of subdirectories for each database.
- DBW – new framework procedure/function for all database platforms (dbw_registerNewParameter). New framework job for upgrading old installations.
- MS SQL Server – Management. Menu “Create new backup device” automatically creates a file name for the device.
- MySQL – “Network statistics” job, fix for avarage statistics calculation.
- Sybase – “Transaction log space check” job, fix for arithmetic overflow.
- MS SQL Server – “Blocking statistics” job, collects hostname for blocked and blocking sessions.
- MS SQL Serer – Management. Added an new column (hostname) to “Blocked/Blocking sessions history” views (job “Blocking statistics” must be upgraded to version 1.9).
- MS SQL Server – “Database backup” job. New parameter “exclude standby databases”, if set to

YES the alert will exclude all databases in state online as read-only, with restore log allowed (IsInStandBy property set to TRUE).
- MS SQL Server – “Database Log backup” job. New parameter “exclude standby databases”, if set to YES the alert will exclude all databases in state online as read-only, with restore log allowed (IsInStandBy property set to TRUE).
- DBW – Network stability
- DBW – Improved coloring of area graphs in management

dbWatch Control Center (2024-01-18)

Important note

- Linux release is now available as an apt repository, see Linux install guide
- All clients and servers need to be upgraded in a multi-server setup before the connection will resume due to new node-to-node communication with TLS if you are running the 2023-09-27 release or older
- Only the server installation is available on Linux in this release (no graphical client or CCC scripting engine)

Bugfix

- Rare connection issue (unable to connect) between some clients and servers dependent on TLS configuration settings on one of the hosts. The issue is known and the fix is being tested. A new release with this fixed is due out shortly.

dbWatch Control Center (2024-01-12)

Important note

- Linux release is now available as an apt repository, see Linux install guide
- All clients and servers need to be upgraded in a multi-server setup before the connection will resume due to new node-to-node communication with TLS
- Only the server installation is available on Linux in this release (no graphical client or CCC scripting engine)

Known issues

- Rare connection issue (unable to connect) between some clients and servers dependent on TLS configuration settings on one of the hosts. The issue is known and the fix is being tested. A new release with this fixed is due out shortly.

New features

- DBW – Node-to-node communication now TLS
- DBW – 2 Factor authentication
- DBW – Forced wait time when login fails
- DBW – Clock skew detection and alerting

- DBW – Secure Cloud Router
- DBW – Metadata on jobs
- DBW – Div minor security improvements
- DBW – Optional anonymous access to web pages
- MS SQL Server – “Security and Compliance” module, contains a set of 40 jobs, a report, and a set of Management views for maintaining security and compliance standards (licensed).
- MS SQL Server – New job “Database compatibility” checks if databases run under lower compatibility mode than the SQL Server instance.
- MS SQL Server – New job “Table size collector” – collects table (HEAP and CLUSTER type) size statistics.
- Oracle – New management view “Space usage history” under Storage statistics
- MS SQL Server – New Management view “Object size history” under “database”/Disk and Memory usage/Largest tables/indexes (top 100)/

Bugfix

- DBW – Email extension – no engine job issue
- DBW – Delete dictionary topics
- DBW – Management pie chart issue
- Oracle – Engine for Oracle 8i, error when creating local job reports for instance centric jobs.
- Oracle – Job “Session load”, fix for aggregated statistics for max. and avg. sessions.
- Oracle – Job “Framework” for Oracle 8i, fix error ORA-1400 for jobs with status warning/alarm
- MS SQL Server – Job “Update index statistics”, the procedure now continues to analyze indexes on the next execution if “max elap time” is reached.

Improvements

- DBW – Network message forwarding optimization
- DBW – Div GUI improvements
- DBW – SQL result network handling
- DBW – Report table cell OK marking
- DBW – Report chapter text
- MS SQL Server – Job “AG synchronization lag”. Checks AG synchronization lag time.
- MS SQL Server – Job “Query Store status”, new parameter which enables it to react when Query Store stops.
- MS SQL Server – Job “Objects size collector (all databases)”, new parameter “max objects per graph”, improves visualization of object growth rates in graphs, and new parameter “ignore READ ONLY databases” which enables possibility to ingnore read-only databases.
- MS SQL Server – Job “SQL statistics” increased the max-size value of the database files of the dbwatch database
- MS SQL Server – Job “SQL Server Agent status” new parameter “return status”. Return status value (ALARM – 2, WARNING – 1, or OK – 0) when SQL Server Agent is not running.
- MS SQL Server – Management, improved blockig statistics views.
- MS SQL Server – Job “Blocking statistics”, historical tables include SID values for blocked and blocking sessions.
- MS SQL Server – Job “SQL Statistics”, improved cleanup routine for sql similarity statistics.

dbWatch Control Center (2023-09-27)

Important note

- Linux release is now available as an apt repository, see Linux install guide
- Only the server installation is available on Linux in this release (no graphical client or CCC scripting engine)

New features

- DBW – Support for MongoDB (beta)
- DBW – Alert forwarding pause
- DBW – Support for Global Report FDL merge
- DBW – Network layout view (beta)
- DBW – Management charts now have an optional description field
- DBW – Local report can now access task parameters
- DBW – Added FDL difference function
- MS SQL Server – New job “Session statistics”. Collects detailed information about sessions connected to the instance (part of performance module, licensed)
- DBW – Semi-automatic upgrade for dbWatch Server

Improvements

- Oracle – “SQL statistics” job, fix for missing statistics when counters are reset.
- Oracle – Improved blocking sessions view in Management
- MS SQL Server – New blocking sessions view statistics in Management
- MS SQL Server – Job “Backup All transaction logs” supports Case Sensitive instances and parameter “ignore databases” allows spaces when listing database names
- MS SQL Server – Job “Backup All databases” supports Case Sensitive instances, and better exception handling, and parameter “ignore databases” allows spaces when listing database names
- MS SQL Server – Job “Database backup”, new parameter “exclude log shipping databases”
- MS SQL Server – Job “Database Log backup”, new parameter “exclude log shipping databases”
- MS SQL Server – Job “DBCC CHECKDB”, new parameter “return status when exception”
- DBW – Job “Instance restart alert”, better exception handling
- Oracle – Installation – Fix for missing “select any table” privilege for dbwatch database user on Oracle 8i platform.
- DBW – Server down status on click loads connection info
- DBW – Removing a server connection will now close the socket
- DBW – Excludes local specs in the default profile
- DBW – Dynamic title in management views
- DBW – Improved upload resource dialog
- DBW – Domain login select/unselect all
- DBW – Generate report dialog filtered per domain
- DBW – Global report compatibility now fdl
- DBW – Global report SQL compatibility – improved version handling
- DBW – FDL Union handles an unlimited number of sets
- DBW – Better marking of ‘Running’ state for jobs

- DBW – Better Thread handling in javascript execution
- DBW – Category “Unknown” for incompatible jobs
- DBW – Worksheet can show JSON format
- DBW – Engine transfer between instances

Bugfix

- MySQL – Job “SQL Statistics” – error when installing on InnoDB HA-Cluster (fix in version 1.4)
- MySQL – Job “Query cache hitrate” – error when installing on InnoDB HA-Cluster (fix in version 3.1)
- MS SQL Server – Job “Reorganize indexes in table”, parameter “minimum fragmentation” works now
- MS SQL Server – Job “Data cache memory usage”, missing parameter “memory threshold”
- Oracle – Job “Index status check” for Oracle 8i wrong column name when compiling
- Oracle – Job “Disk read statistics” for Oracle 8i to_char compiling error fix* Avoid multiple domain initializations
- DBW – Monitor hang bug
- DBW – Mail format loading
- DBW – Mail multiple ‘lost connection’ messages
- DBW – Access control for web
- DBW – SQL Worksheet – missing results
- DBW – No engine job properties caching issue
- DBW – Report font resolution fix
- DBW – FDL compare number and text fix.

dbWatch Control Center (2023-06-27)

Important note

This release upgrades encryption standards to AES/GCM 256-bit and from Oracle JDK to OpenJDK. When upgrading from a previous release in a multi-server environment or environments where the dbWatch Server and dbWatch Client is on different machines, all dbWatch Servers and Clients must be upgraded before you can connect again.
If you encounter any issues, please get in touch with support@dbwatch.com, and we will sort it out for you.

Bugfix

- DBW – Fix for socket handling of outgoing connections from the server
- DBW – Test connections from “Node connections” dialog.
- DBW – Edit access points from Domain Login dialog.
- DBW – Delete role/security group
- DBW – Local report hang/slow bug
- DBW – Missing styling in “Show report” from task editor
- DBW – Fixed issue when transferring domain
- DBW – Save as new report did not work correctly
- DBW – Generate report, select instance did not always populate

- DBW – Upload resource, select file dialog did not always appear
- DBW – Lost connection message could be sent multiple times
- DBW/PostgreSQL – Session to multiple databases issue
- Oracle – “RMAN backup status” jobs, exception pos 0. Added new parameter and the backup history table.
- DBW – Farm views – waits statistics pie-charts

New features

- DBW – “Hover” on table text and date cells.
- DBW – Instance access control from web dashboard.
- DBW – Privileges on dbWatch Servers.
- DBW – Restart Server functionallity
- DBW – Domain discovery on/off setting
- Oracle – new job “Applied archive log gap status”
- MS SQL Server – Management – Sessions Performance views showing statistics gathered by the “Session statistics” job (LICENSED).
- DBW – new internal view for session statistics
- DBW – new job “Connection alert” – check connection status for property and engine users.
- Oracle – new job “Flash recovery space usage” for Standby instances.
- MS SQL Server – new job “Test alert” – Useful for testing extensions (not intended for use in production environments).

Improvements

- DBW – Switched to OpenJDK
- DBW – Switched to bouncycastle for crypto
- DBW – Email extension queue and retry
- DBW – License expired dialog contains the server name
- DBW – Job parameter description trimmed.
- DBW – Certificate CN contains the username
- DBW – AD config copied when transferring domain.
- DBW – Improved message crypto (AES/GCM 256-bit)
- DBW – Lost connection email contains more info
- DBW – Can not delete domain controller node
- DBW – Node info dialog
- DBW – Improved error handling in charts
- DBW – Group configuration now available for Super DBA role
- DBW – Notifications dialog
- Oracle – Default max tablespace size now 20GB
- MS SQL Server – Default max database size now 20GB
- PostgreSQL – “Session load” – parameter replacement to percentage. New session aggregation table.
- Oracle – “Job Scheduling Check” – improved execution time of the job.
- Oracle – Management – improved blocking statistics views.
- MS SQL Server – “SQL Statistics” – improved deletion of old data so that the transaction log to the dbWatch database does not become full.
- MS SQL Server (v. 2014 +) – “Database Log backup” and “Database backup”, new parameter:

"exclude databases in Always On availability groups"
- MS SQL Server – "AlwaysOn database backup alert" – 4 new parameters, added differential backup check option and uppercase option.
- MS SQL Server – "AlwaysOn transaction log backup" – new parameter – use_uppercase
- MS SQL Server – "AlwaysOn differential database backup" – new parameter – use_uppercase
- MS SQL Server – "Database Server uptime" – improve resource consumption
- MS SQL Server – "Internal fragmentation check" – new parameter "continue on next run", makes it possible to carry out the job over several days
- MS SQL Server – "External fragmentation (all databases)" – new parameter "continue on next run", makes it possible to carry out the job over several days
- MS SQL Server – "Data cache memory usage" – collects execution statistics, new parameter disabling running on instances with high memory usage.

dbWatch Control Center (2023-04-27)

Known bugs

- Problem with driver autodetection, so adding SQL Server 2005 and 2008, requires manually changing the JDBC driver during add instance dialogue to the 9.4.0 version of the driver.
- A problem may occur when installing "RMAN backup status" job on Oracle. Typical error: (Exception, ORA-01403). If this is experienced, please get in touch with support for a patch.

New features

- DBW – Added password policy
- DBW – Job templates can now include other templates in priority order
- DBW – Added 'Resume monitoring' menu to instances
- Oracle – new job 'Tablespace in BEGIN BACKUP mode' – checks if any data files are in BEGIN BACKUP mode.
- MS SQL Server – 'MS SQL Server patch status' – added patches up to 21st Apr. 2023
- MS SQL Server – Management, new menu 'Create snapshot'
- DBW – New Internal FDL Performance view
- DBW – New Internal sessions view

Improvements

- DBW – Improved feedback in test connection for Email extension
- DBW – Better handling of clock skew in certificate validation
- DBW – Job template editor shows parameter description
- DBW – Improvements to session handling
- DBW – More stable connection for Azure
- DBW – Deleting a job template that is in use is no longer allowed
- DBW – Added threshold for thread dump on high cpu usage
- DBW – Improved styling of errors in tables
- DBW – Performance improvements in property loading
- DBW – Improved 'Lost connection error' dialog
- DBW – CSV import now creates subgroups

- DBW – Thread optimization for running engine jobs
- DBW – Added popup when hovering on large table cells
- DBW – Changed firewall settings are enforced at once
- DBW – Added ‘single’ hashtag for forcing single-line fdl result
- MS SQL Server – job “High activity monitor” description change for 4 parameters.
- MS SQL Server – job “Rebuild indexes” has new parameter “ignore tables”
- MS SQL Server – job “Database status” new parameter “ignore standby databases”
- MS SQL Server – job “AlwaysOn transaction log backup alert” improved performance on large AlwaysOn clusters (many nodes)
- MS SQL Server – job “AlwaysOn differential database backup alert” improved performance on large AlwaysOn clusters (many nodes)
- MS SQL Server – job “AlwaysOn database backup alert” improved performance on large AlwaysOn clusters (many nodes)
- Oracle – job “Tablespace free space check” better description if too many tablespaces are in warning/alarm state.
- Oracle – job “Job scheduling check” better exception handling and support for % (percent sign) to represent wild card characters for parameter “ignore job names”
- Oracle – job “RMAN backup status” new parameters and now checks incremental level 0 backup.
- Oracle – job “RMAN archivelog backup status” new parameter.

Bugfix

- DBW – Scheduled report instance list showed guid
- DBW – Delete scheduled report
- DBW – EMail issues for lost connection on no engine instances
- DBW – Jump issue when deleting job in Job template editor
- DBW – Fixed ‘back’ button when creating global report.
- DBW – More stable expanding of nodes in Management
- DBW – Issues with ‘pause monitoring’ for no engine instances
- DBW – Rounding error in fdl div function
- DBW – Issues with setting job template in CSV import
- DBW – ‘Pause monitoring’ was always 30 minutes
- DBW – Buggy adding of metadata in the instance configuration view
- DBW – Buggy ‘Edit dictionary’ dialog
- DBW – Loading of worksheet authentications in multi-server environments
- Oracle – Job ‘SQL statements (Logical reads)’ – better exception handling
- Oracle – Removed RMAN backup jobs for PDB databases
- Oracle – Management fix for Session view for CDB and PDB instances
- Oracle – Job “SQL statistics” – CDB instances do not collect statistics for PDB instances
- MS SQL Server – Management menu fix ‘Change compatibility level’ for 2022
- All DBMS platforms – job “Instance restart alert” better exception handling

dbWatch Control Center (2023-03-10)

Known bugs

- Problem with driver autodetection, so adding SQL Server 2005 and 2008, requires manually

changing the JDBC driver during add instance dialogue to the 9.4.0 version of the driver.

Bugfix

- DBW – Job templates could sometimes drop dbwatch monitoring objects in the database
- DBW – Instance status was not always updated when uninstalling a job
- DBW – Table filter did not always work
- DBW – Delete last metadata fix
- Oracle – Handling of Service name
- MySQL – Global report handling

New features

- MS SQL Server – New job “Agent error log”, Checks for errors in the Agent error log

Improvements

- DBW – Improvements to Job Template GUI
- DBW – Job template deployment more robust
- DBW – Job template copy to Server functionality
- DBW – One time monitoring pause
- DBW – Job deployment messages in a separate log file
- DBW – Improvements to edit metadata GUI
- DBW – Add/remove monitoring database
- DBW – Uninstall job with no installation script
- DBW – Message performance improvement in multi-server, single-domain setups
- DBW – Auditing of domain config changes
- MS SQL Server – New version of “Query Store status” (1.2) job, fix for history data
- MySQL – New version of the “MySQL max connections” (1.2) job, does not change status to OK after WARNING/ALARM (error when using Max_used_connections value)
- MS SQL Server – New version of “MS SQL Server patch status” (1.9), added patches up to 16th Feb. 2023
- MS SQL Server – New version if “High activity monitor” (1.8) job – fix for NULL details when “Page lookups/sec” counter is missing
- Oracle – fix in the local report for “CPU load check” Oracle.

dbWatch Control Center (2023-02-10)

Known bugs

- Problem with driver autodetection, so adding SQL Server 2005 and 2008, requires manually changing the JDBC driver during add instance dialogue to the 9.4.0 version of the driver.

Bugfix

- DBW – Rename instance
- DBW – Access control fix for “Edit connections” in Domain Configuration
- DBW – Refresh bug in “Jobs Status” in Monitoring View

- DBW – Fix for Delete Job Template
- DBW – Escape handling in Job Template config file
- Oracle – Feedback in Worksheet for buggy SQL statements
- MS SQL Server – New version of “High activity monitor” (1.7), Divide by zero error fix
- MS SQL Server – New version of “Objects size collector (all databases)” (1.5), “04 Incorrect syntax near the keyword” fix.
- MS SQL Server – New version of “Transaction log space usage” (2.3), bug fix for invalid usage of DECLARE CURSOR statement
- MySQL – property fix for memory_usage, disk_usage and edition

Improvements

- DBW – Job template mode has been renamed for better readability
- DBW – Improved safeguards when enforcing Job Templates
- DBW – Timeout in SQL Worksheet

dbWatch Control Center (2023-02-07)

Known bugs

- Problem with driver autodetection, so adding SQL Server 2005 and 2008, requires manually changing the JDBC driver during add instance dialogue to the 9.4.0 version of the driver.

New features

- DBW – Job Template Editor
- MySQL – New performance tuning module (this functionality is licensed)
- MySQL – New job “SQL statistics.” Collects statistics for the “SQL performance module” available in Management GUI (this functionality is licensed)
- DBW – License Management View
- DBW – Audit of user actions

Bugfix

- DBW – Fix for polling pause edit
- DBW – Email extension Reconnected message
- DBW – Fix for incorrect TLS handling in email extension
- DBW – Custom tabs in the Monitoring view
- DBW – Handling of duplicate properties when using local definitions
- DBW – Task editor slow issue
- DBW – Minor stuff
- MS SQL Server – Set management auth forgot the database
- Oracle – Support for INTERVALYM / INTERVALS
- MySQL – Fix for job upgrade
- PostgreSQL – Fix for job upgrade
- Azure – Management fix for Azure single db

Improvements

- DBW – New look & feel for Group Configuration
- DBW – More robust CSV import
- DBW – Better logging of adding/removing instances to Server Notes
- DBW – Improved formatting of SMS messages
- DBW – Improvements to Access Control
- DBW – Improved handling of parallel sessions in management
- DBW – Edit Jobs available from the Manage Jobs view
- Oracle – New version of “CPU load” (3.7), new parameter “Time threshold”
- Oracle – New version of “Long running queries” (1.6), fix for lowercase
- MS SQL Server – New version if “SQL statistics” (1.6), new parameters to enable/disable similarity query text calculations.
- MySQL – New job “Network statistics” replaces “Network traffic”
- MySQL – New version if “Query cache hitrate”, now with status, new parameters, and better exception handling
- MySQL – New job “Threads statistics” replaces “Thread cache hitrate”
- MySQL – New job “Statement load” replaces “Database load”
- MySQL – New version of “DBMS uptime”, new parameters, and better exception handling
- MySQL – New version of “Binlog cache check”, new parameters, and better exception handling
- MySQL – New version of “Innodb buffer pool check”, new parameters, and better exception handling
- MySQL – New job “MyISAM key cache check” replaces “Buffer key cache check” and “Key buffer check”
- MySQL – New job “Framework”, for internal use only (added new procedure – dbw_remove_job proc)
- MySQL – New version of “Session load”, better exception handling
- MySQL – New version of “Temporary table check”, new parameters, and better exception handling
- MySQL – Improved resolving of Memory and Edition properties

dbWatch Control Center (2022-12-15)

Known bugs

- Problem with driver autodetection, so adding SQL Server 2005 and 2008, requires manually changing the JDBC driver during add instance dialogue to the 9.4.0 version of the driver.

New feature

- DBW – Firewall
- DBW – Command line interface (CCC)
- DBW – Authentication for Web dashboards
- DBW – Web Dashboard Editor now in beta
- PostgreSQL – New job “Blocking statistics”. Checks for blocked and blocking processes.
- MS SQL Server – New job “Autoshrink settings”. Check if all databases have auto-shrink turned off.
- MS SQL Server – New job “AlwaysOn differential database backup”. Check the backup date (type I, differential backup) for all local databases and databases in the AlwaysOn groups (primary and secondary).

- PostgreSQL – Support for Postgres 15 for «SQL statistics» – Performance module.
- PostgreSQL – New “Blocking view” in Management.
- Linux installer included

Improvement

- DBW – More flexible access control.
- DBW – Improved wizard for adding SQL Cluster.
- DBW – CSV import handles multiple license tags.
- MS SQL Server – New version of “AlwaysOn database backup” job (2.9). Now supports instances on the same host.
- PostgreSQL – New version of “Schema growth and information” job (1.8). Improved synchronization with the “Schema growth collector” job, and better exception handling.
- Oracle – New version of “Long running queries” job (3.5). New parameter “Exclude schemas”.
- MS SQL Server – New version of “Transaction log space usage” job (2.2) for MS SQL Server 2017+. DBCC SQLPERF replaced with sys.dm_db_log_space_usage.
- Oracle – Improved handling of Oracle database link in management.

Bugfix

- DBW – Domain configuration – delete node.
- DBW – CSV import – name collision avoidance for Oracle and MS SQL Server.
- DBW – Instance reconnect issue.
- DBW – Fix for HTML-based global reports.
- DBW – Minor fixes.
- PostgreSQL – New version of “SQL statistics” (1.2) job. Fix for deleting of old statistics.
- PostgreSQL – New version of “Vacuum alert” job (2.7). Spelling mistake.
- Oracle – full version display fix for Oracle 18+.

Known bugs

- Problem with driver autodetection, so adding SQL Server 2005 and 2008, requires manually changing the JDBC driver during add instance dialogue to the 9.4.0 version of the driver.
- Importing instance configuration data from dbWatch Enterprise Manager 12, does not work with Oracle RAC and Dataguard systems

dbWatch Control Center (2022-11-03)

New features

- PostgreSQL – No longer beta. Old engine installations must be reinstalled.
- PostgreSQL – new performance tuning module (this functionality is licensed)
- DBW – AD integration
- DBW – GUI for Web Server initialization
- DBW – Web Dashboard now available

Bugfix

- MS SQL Server – New version (1.1) of the “Query Store status” Job. Handles unavailable databases
- DBW – Bugfix instance reconnect issue
- DBW – Bugfix refresh issue in the Manage Jobs view
- DBW – Bugfix job autoupgrade
- DBW – Bugfix fdl orderby NoValue

Improvement

- MS SQL Server – New version (1.5) of the “SQL statistics” Job. Now handles missing indexes
- DBW – QueryLog view now contains SQL
- DBW – Improved edit node connections dialog
- DBW – Improved handling of misconfigured extensions
- DBW – Improved handling of Job installation failure
- DBW – Improved global report generation

dbWatch Control Center (2022-09-29)

Known issues

- SQL Server Consolidation report is visible in the release but is not working as underlying extra cost features are omitted

New features

- MS SQL Server – new performance tuning module (this functionality is licensed)
- MS SQL Server – new job “Query Store status.” Checks Query Store space usage.
- MS SQL Server – new configuration options for Query Store in Management GUI
- MS SQL Server – new job “SQL statistics.” Collects statistics for the “SQL performance module” available in Management GUI (this functionality is licensed)
- MS SQLServer – Kerberos integration
- Oracle – new job “SQL statistics.” Collects statistics for the “SQL performance module” available in Management GUI (this functionality is licensed)
- Oracle – new performance tuning module (this functionality is licensed)
- MariaDB – Now supported
- PostgreSQL – Support for version 14
- DBW – Management graph range restriction/zoom
- DBW – Management selectors
- DBW – Management tabs
- DBW – Management Studio query plan viewer integration
- DBW – Improved CSV import (JDBC driver & global auth)
- DBW – Logging of instance adding to server notes
- DBW – Domain transfer
- DBW – Feedback when license change will disable jobs
- DBW – Freetext notes
- DBW – Job failure log summary

Improvements

- MS SQL Server – new version 1.7 of “MS SQL Server patch status” —> checks for new patches for MS SQL 2014, 2017, and 2019 released before 2022.09.20.
- Oracle – Performance moved into Server activity in management, making room for a new performance module
- Oracle – Improved Amazon RDS support

Bugfix

- Oracle – Adding data files and changing storage in OMF
- Oracle – Extending and altering tablespaces in RDS
- Oracle – RMAN information didn’t show up correctly in some instances
- Oracle – Instance alert log check, fixed typo in the report
- Oracle – Fixed bug when adding instances with OMF or ASM
- Oracle – Fixed issue on multi-dataguard sites not being possible to add
- Oracle – Multitenant screen in management stuck on loading
- Oracle – Some management screens stuck on loading
- Oracle instance-id handling in management
- PostgreSQL . Postgres 9.6 is not getting detected correctly in the management module
- PostgreSQL – Replication delay sometimes didn’t detect replication
- PostgreSQL – Fixed typo in property, causing some jobs not to show up
- PostgreSQL – BDR replication job installable on non-BDR setups
- PostgreSQL – SSH-based jobs visible without configuring ssh
- MySQL – Fixed many graphs not showing correctly on some versions
- MySQL – Some management screens are stuck on loading
- MySQL – Issue when uninstalling jobs
- Linux – Fixed issue on CPU usage and Disk usage checks not showing up
- SSH connector stability
- DBW – FDL with restricted access to instances
- DBW – ISsues with triggering of no engine jobs
- DBW – Issues with instance property flush
- DBW – Try again when property resolving fails

Note on upgrade
For customers upgrading from previous versions of dbWatch Control Center, be sure to take a backup of the C:\ProgramData\dbWatchControlCenter\config directory. Feel free to contact dbWatch support if you need any assistance. We are happy to help you via zoom or TeamViewer.

Remember that a new “dbWatch Control Center” service is created when you run the installer. If this service has been configured under an account other than the local account, remember to reconfigure it again.

There is also a new license structure. After upgrading, all jobs will execute normally, but installing new jobs and uninstalling already installed jobs may be disabled (depending on your current license). If you experience any issues, contact us, and we will provide you with an updated license.

dbWatch Control Center (2022-05-25)

Bugfix

- DBW – Fix alert triggering with different timezones.
- DBW – Monitor/Server connection issue fix.
- DBW – Fix for lack of automated refresh in management view.
- MS SQL Server – new version 1.3 of “Sessions per database.” Fixed NOT NULL constraint error, improved exception handling
- MS SQL Server – new version 1.2 of “Database growth rate (aggregated).” Fix for arithmetic overflow (error converting expression to data type int) for databases larger than 2.1 TB
- MS SQL Server – new version 1.8 of “Database growth rate (detailed).” Improved exception handling
- MS SQL Server – new version 2.1 of “Transaction log size check.” Fix for arithmetic overflow (error converting expression to data type int) for databases larger than 2.1 TB
- MS SQL Server – new version 1.7 of “Blocking statistics.” Better handling of NULL values in details.

dbWatch Control Center (2022-05-05)

Improvements

- DBW – Added timeout setting for no engine jobs
- DBW – Performance improvements in monitoring for large environments
- DBW – Engine job triggering is more robust
- DBW – No engine jobs use less disk space in the property store
- Oracle – New version 2.8 of “Session load Rac.” Transformed to an alert with eight new parameters (need dbWatch Server restart after upgrade).
- Oracle – New version 3.3 of “Session load.” Transformed to an alert with four new parameters (need dbWatch Server restart after upgrade).
- Oracle – Improved history in session graph
- MS SQL Server – New version 2.2 of “Database disk capacity.” Supports now log list of drive specifications.
- MS SQL Server – New version 1.6 of “MS SQL Server patch status.” Checks for patch CU16 for MS SQL 2019, RTM CU28, and RTM CU29 for MS SQL 2017.
- Postgres – Improved history in session graph, logical reads, and transactions per second
- Oracle – Extent fragmentation job – added parameter for skipping tablespaces
- Oracle – Tablespace free space for noschema installation, now supports excluding tablespaces and including unallocated free space
- Postgres – BDR replication lag, new parameter max histr interval, allowing old historical records to be removed automatically
- MS SQL Server – New version 1.61 of “Check database recovery mode.” Excludes mirrored and Availability Groups databases
- MS SQL Server – New version 1.4 of “Database backup (system databases). Added to new parameter for better configuration flexibility
- MS SQL Server – New version 1.72 of “Index usage statistics.” More trace info when an exception occurs. It also excludes the tempdb database.

- MS SQL Server – New version 2.54 of “DBCC CHECKDB.” Added new parameter “ignore errors” to exclude irrelevant exceptions
- MS SQL Server – New version 1.3 of “Transactions log flushed bytes load.” Better exception handling

New features

- DBW – Added views for insight into dbWatch routing and connection info
- DBW – Instance config in separate files
- DBW – Left side tree in monitor now minimizable
- DBW – Single domain multiserver now in beta
- DBW – Lots of changes to internal views
- DBW – Restart the node network in the server’s context menu.
- DBW – Webserver
- DBW – New vmoptions settings for debug output
- Healthcheck report added for Oracle, Postgres, and MySQL
- Oracle – RMAN archivelog backup job, monitoring archivelog backup sets
- MS SQL Server – New alert 1.1 “Shrink transaction logs” – shrinks tr. logs detected by “Transaction log size check” alert

Bugfix

- DBW – Lots of fixes for general stability
- DBW – Fixed memory leak for no engine jobs
- DBW – Job template handling
- DBW – Lots of fixes for network stability
- DBW – Changes to group defaults sometimes caused the config file to be corrupted
- DBW – Management specification loading more robust
- DBW – Now handles Oracle timestampz type
- DBW – Unregister instance now works
- DBW – License handling when the license is almost depleted
- MySQL – Uninstall job now works on MySQL 6 and newer
- MySQL – Bugfix for performance-related jobs that did not get correct values
- Oracle – Create user menu option moved to correct location
- Postgres – Bugfix in management security view that was not showing
- MySQL – NDB jobs are now not included in the default install
- Oracle – Alert log check, a minor adjustment in the description
- Oracle – RMAN backup job, now ignoring archivelog backup sets
- Oracle – Autoextensible files job improved compatibility query
- Oracle – Extent fragmentation – Improved query for better database response
- Oracle – Top user memory usage – New default threshold values
- MS SQL Server – New version 1.8 of the “framework” task. Engine fix for SQL Server 2000 (database-centric alerts).
- MS SQL Server – New version 1.2 of “Shrink transaction log.” Bug when deleting history records. (CC)
- MS SQL Server – New version 1.7 of “Database growth rate (detailed).” Missing parameter when a new installation
- MS SQL Server – compatibility fix in Management views

- MS SQL Server – cosmetic fix for local report for [Test DML-DDL performance] job
- MS SQL Server – missing [activity_dtu_test] property. Fix of [Performance test] in the Management view

dbWatch Control Center (2022-02-14)

New features

- MS SQL Server – New version (1.4) of “MS SQL patch status.” Checks patch level of SQL Server instance.
- MS SQL Server – New job “Shrink transaction logs.” Shrinks tr. logs detected by “Transaction log size check” alert
- DBW – New fdl topic “iospeed.”

Improvement

- MS SQL Server – New version (1.61) of “Check database recovery mode.” Excludes mirrored and Availability Groups databases.
- MS SQL Server – New version (1.4) of “Database backup (system databases).” Added to new parameter for better configuration flexibility
- MS SQL Server – New version (1.72) of “Index usage statistics.” More trace info for exceptions (excludes tempdb database)
- MS SQL Server – New version (2.54) of “DBCC CHECKDB”. Added new parameter “ignore errors” to exclude irrelevant exceptions
- MS SQL Server – New version (1.3) of “Transactions log flushed bytes load.” Improved exception handling
- DBW – Improved stability in multi Server/multi Client setups
- DBW – New implementation of job triggering.
- DBW – More stable property loading
- DBW – Job templates can be triggered for a single instance
- DBW – Improved network logging functionallity
- DBW – More stable Domain Controller startup

Bugfix

- MS SQL Server – Bugfix in Management database compatibility menu
- Oracle – Fixed bug in management. Results not showing
- Oracle – Disk space check did not displace space correctly on Windows Server 2019
- Oracle – Rebuild indexes report not displaying correctly
- DBW – Bugfix for initial loading of authentications in SQL Worksheet

Some changes to the config file layout. The server will move the files at the first startup. Some users have experienced needing to restart the server again for the changes to have an effect.

dbWatch Control Center (2022-01-14)

Bugfix

- DBW – Bugfix in the handling of property storage policy
- DBW – Removed false, lost connection messages
- DBW – Metadata dictionary can now delete values
- DBW – Buggy license counting fixed
- DBW – Minor stability issues fixed
- MS SQL Server – Bugfix for “External fragmentation (all databases)” job
- MS SQL Server – Bugfix for “[Check database recovery mode” job
- MS SQL Server – Bugfix for “Index usage statistics (all databases)” job
- MS SQL Server – Bugfix for “Filegroups growth rate” job
- Oracle – Bug fix in compatibility tag that made some jobs disappear from the manage jobs overview
- Oracle – Added pre-implementation steps for some scenarios where jobs didn’t install
- Oracle – Typo fixes in multiple reports

New features

- DBW – Job templates – deploy jobs in bulk
- PostgreSQL – Added job BDR Replication lag

Improvements

- DBW – Added metadata dictionary key syntax check
- DBW – Lost connection handling Monitor/Server

Known issues

- Oracle – Job templates do not work correctly on RAC

dbWatch Control Center (2021-12-22)

Bugfix

- Oracle – RMAN 11g Backup check, intermittent exceptions due to lack of privileges
- Oracle – RMAN backup check for 12 and newer, intermittent exceptions due to lack of privileges
- Oracle – Job scheduling check, intermittent exceptions due to lack of privileges
- Oracle – ASM disk statistics check, intermittent exceptions due to lack of privileges
- Oracle – ASM diskgroup check, intermittent exceptions due to lack of privileges
- Oracle – Autoextensible files check, intermittent exceptions due to lack of privileges
- Oracle – Free extents check, intermittent exceptions due to lack of privileges
- Oracle – Tablespace free check, intermittent exceptions due to lack of privileges
- Oracle – Job Scheduling check, improved handling of large job names and multiple errors
- Oracle – Tablespace free space check, improved handling for ignored tablespaces
- Oracle – new version: Delete SYS.AUD$ data

- Postgres -new version: Session load
- DBW – handling of engine scripts with multiple versions
- DBW – Bugfix in task editor – save as new

Improvements

- Oracle – Adjusted history thresholds to a minimum of 14 days for jobs
- Oracle – Improved graphs Management, Better history for Logical IO
- Oracle – Improved graphs Management, Better history for Network throughput
- Oracle – Improved graphs Management, Better history for Transactions per sec
- DBW – sub/add function now has optional timeunit argument
- DBW – improvements to job template handling
- DBW – improvements to monitor/server socket handling
- DBW – improvements to the internal initialization sequence
- DBW – filtering dates in tables
- DBW – improved caching of management specifications on the monitor
- DBW – improved handling of Oracle RAC
- DBW – improved handling of Oracle Dataguard
- DBW – improved handling of service groupings
- DBW – improved handling of exhausted license
- DBW – improved property loading
- DBW – number alignment in tables
- DBW – improved default mail configuration

New features

- MySQL – new job: Innodb cluster status
- MySQL – new job: Innodb cluster switch
- MySQL – new job: Replica count
- MySQL – new Job: Replica delay
- MySQL – new Job: Replica state
- MySQL – New job for NDB cluster, NDB data node status
- MySQL – New job for NDB cluster, NDB data memory usage
- MS SQL Server – added Kerberos support
- MS SQL Server – new task: Database session load
- MS SQL Server – new version: Database growth rate detailed
- MS SQL Server – new version: Database disc capacity
- MS SQL Server – new version: Filegroups growth rate
- Oracle – Adding ssh based jobs for AIX and Linux
- DBW- task parameter default value now available in fdl
- DBW – instance import now supports job templates
- DBW – added instance menu to status tables
- DBW – added ssh support
- DBW – added compacting of static property store

dbWatch Control Center (2021-10-19)

New features

- Basic job template support
- Added new Farm Maintenance views for Microsoft ERROR log statistics
- New job for MS SQLServer, “Database session load,” which collects session statistics per database

Bugfix

- Pie chart color rendering
- Fixed issues in PostgreSQL reconnect

Improvements

- Improved speed for Oracle connect
- More robust socket handling
- Cosmetic changes to the Domain Login dialog
- Server performance improvements
- Improved Internal views

Misc

- Removed backup statistics from Farm views

dbWatch Control Center (2021-09-09)

- Improvements to domain handling
- Improvements to network stability
- Improvements to FDL handling in multi-domain setups
- Improvements to eval license handling

- New dialog for initial domain setup
- New Login Dialog
- New E-Mail configuration dialog

- Added ‘Clone & Detach’ feature for charts and tables

- Bugfix: for Copy Cell
- Bugfix: You can now uninstall upgradable no-engine jobs
- Bugfix: Local reports for no engine jobs
- Bugfix: Missing labels on some charts
- Bugfix: Configuration backups
- Bugfix: Countless minor bugfixes

- New version of MS SQLServer “Agent Jobs Check” 2.7

- New version of MS SQLServer “framework” 2.2
- New version of MS SQLServer “Database disk capacity” 2.0
- New version of MS SQLServer “Wait statistics” 1.6
- Backup overview (for MS SQL Server) – Beta
- Maintenance overview (for MS SQL Server) – Beta

- Oracle Dataguard apply time, a new basic report added
- Oracle Dataguard archive sequence apply lag, a new basic report added
- Oracle Dataguard gap alert, a new basic report added
- Oracle Dataguard startup time, a new basic report added
- Oracle Dataguard transport time, a new basic report added
- Oracle Dataguard apply time, adjusted default schedule to run less frequent
- Oracle Dataguard archive sequence apply lag, adjusted default schedule to run less frequent
- Oracle Dataguard startup time, adjusted default schedule to run less frequent
- Oracle Dataguard transport time, adjusted default schedule to run less frequent
- Oracle Dataguard datafile status, adjusted default schedule to run less frequent
- MySQL/MariaDB, Improvements in engine scripts for improved cloud support
- Oracle DB Uptime, Fixes for report issues
- Postgres Locks held and statistics, Bugfix on job reinstall issues
- Postgres Transaction statistics, Bugfix on job reinstall issues

dbWatch Control Center (2021-05-28)

Bugfixes

- Metadata storage fixed
- Improved stability in message routing
- Incorrect refresh in the Manage Jobs view
- Version handling when importing no-engine jobs
- Missing label in report charts
- Adding instances with MySQL 5.0 – 5.6 on AWS or on-premise could fail with privilege issues
- Oracle database network statistics improved housekeeping
- Server selection now works in the Extension view

Improvements

- Support for AlwaysOn cluster
- Added time topic for server
- Improved “Domain Login” dialog
- Improved “Connected users” view
- New “Internal Notes” views
- New “Internal engine files” views (MS SQL Server, Oracle)
- Can now “delete” auto-discovered instances

Known issues

- AlwaysOn
    - You can get an error the first time you execute the “AlwaysOn database backup alert” and “AlwaysOn transaction log backup alert.”

    - The databases in Availability groups are not shown in management.
    - You must select “Cluster” as a grouping in the tree. This selection is forgotten on Monitor restart.

dbWatch Control Center (2021-05-06)

Bugfixes:

- Connection stability issues monitor/server
- Issues with job upgrade
- Issues in add instance wizard
- Issues in Connect/Disconnect to all instances
- Duplicates in upload resource view
- Sort order in the acknowledgment history view
- Fixed “Show error log,” for instance
- Stability issues in chat
- Lots of minor fixes in stability and user interface

Improvements

- Improved dialog for initial domain setup
- Improved installation wizard
- Added “busy” indicator in domain login dialogs
- Login on entering in domain login
- New version for MS SQL Server “Database growth rate (detailed)” job
- New version for MS SQL Server “Filegroups growth rate” job
- New version for MS SQL Server “Transaction log space usage” job
- New version for MS SQL Server “Database disk capacity” job

dbWatch Control Center (2021-04-23)

Improvements

- More user-friendly flow for initial installation
- Improved stability
- Improved Access control

Bugfix

- Generation of global reports

Limitations

- Multi-domain is not stable
- Postgres drivers can only support up to version 9.6 since the available driver only applies to 9.2.1003, which does not have SCRAM-SHA-256.
- dbWatch server on Linux and Ubuntu are not yet supported
- Same MySQL installation error as RC 3
- PostgreSQL unstable (Work in progress)

dbWatch Control Center RC 4.0

RC 4.0 focuses on UI stability and additional features from RC 3.0

New Features

- Recommended to install to a small environment with instances no more than 100 instances
- Upon installation, dbWatch will ask for the license file
- SMTP extensions for emails
- UI stability where the connection does not prematurely die
- Brent Ozar module.
- Farm Module Views are split into three separate files; this is a fix to RC3's single instance view problems when there are more than 20 instances registered

Known Issues:

- Postgres drivers can only support up to version 9.6 since the available driver only applies to 9.2.1003, which does not have SCRAM-SHA-256.
- dbWatch server on Linux and Ubuntu are not yet supported
- Same MySQL installation error as RC 3
- Recovery options for MSSQL are only limited to SIMPLE recovery (FULL backup restore)
- In the "Add Instance" Wizard, UI locks the "NEXT" button when returning from the job installation process
- Limited Security Set-up – only admin credentials are available with six instances
- Bulk Installation is still riddled with bugs when installing databases

dbWatch Control Center RC 3.0

RC 3.0 integrates a few new features from RC 2.0

New Features

- Can support up to 100 databases for now; will keep on expanding to accommodate more
- Autodiscover feature
- dbWatch internal is added as a view

Known Issues:

- Connection dies after 30 minutes of no activity
- MySQL installation for version 8 and above encounters an error during installation
- dbWatch server on Linux and Ubuntu are not yet supported
- More than 20 instances will cause a bug for single instance views; you can still see the overview but clicking on an instance will
- No stress testing is done with multiple servers housing hundreds of databases
- In the "Add Instance" Wizard, UI locks the "NEXT" button when returning from the job installation process
- Limited Security Set-up – only admin credentials are available with six instances at max
- Bulk Installation is still riddled with bugs when installing databases

dbWatch Control Center RC 2.0

RC 2.0 is an improvement from RC 1.0 regarding UI stability and dbWatch server connection.

New Features

- Can support up to 100 databases for now; will keep on expanding to accommodate more
- Bulk Instance Installation
- Supports up to **Postgres 12** and **MySQL** database engines

Known Issues:

- No comprehensive testing done with **Sybase**, **MySQL**, **MariaDB**, **Ingres** and **Postgres**
- dbWatch server on Linux not yet supported
- No stress testing is done with multiple servers housing hundreds of databases
- In the “Add Instance” Wizard, UI locks the “NEXT” button when returning from the job installation process
- Limited Security Set-up – only admin credentials are available with six instances at max
- Bulk Installation is still riddled with bugs when installing databases

dbWatch Control Center RC 1.0

Control Center takes your monitoring solutions to new heights. Similar to **dbWatch 12.7**, it covers all your monitoring needs.
More importantly, it builds on its functionality and user-friendly design with the following:

- Extensive dashboards and view to monitor and manage larger database server farms
- new and easy-to-use UI
- Expand the job list of dbWatch 12.7 monitoring solution for SQL and Oracle even further
- Supports Oracle 12, 18 and 19, and MSSQL 2012, 2014, 2017 (Mysql)
- Farm Data Language – an improved version of Dbwql

Tested and supported:

- **Oracle** 8, 9, 10, 11, 12, and 9; both standard and enterprise editions
- **MSSQL** for 2000, 2005, 2008, 2012, 2016, 2017, and 2019; both for Standard and Enterprise Editions
- dbWatch Server installed on Windows 10

Known Issues:

- No full testing done with **Sybase**, **MySQL**, **MariaDB**, **Ingres** and **Postgres**
- dbWatch server on Linux not yet supported
- Incoherent error detection, such as character limit that prompts error: illegal characters inputted
- No Max and Min hours of scheduling.
- Repeated configuration in installing jobs can produce bugs
- No stress testing done with multiple servers housing hundreds of databases
- In the “Add Instance” Wizard, UI locks the “NEXT” button when returning from the job installation process

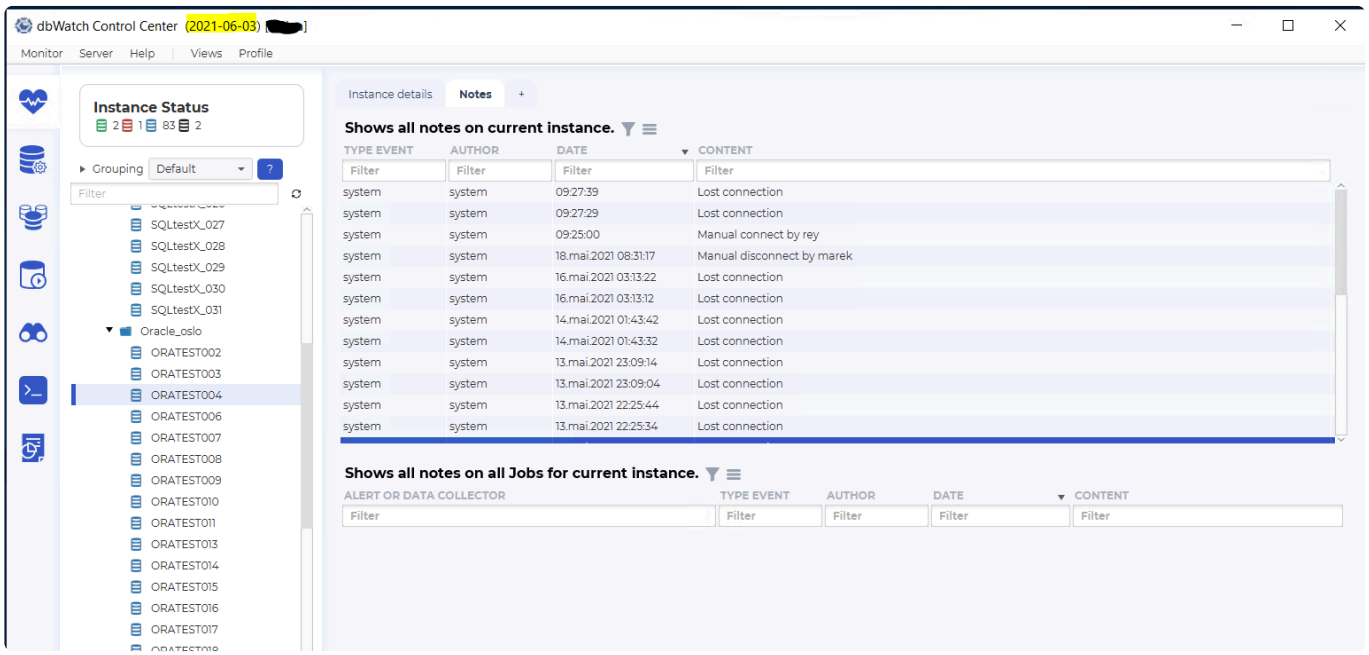
- Limited Security setup – only admin credentials are available
- Stability Issues when connecting to several servers

[<< About dbWatch Control Center] / [Find your version >>]

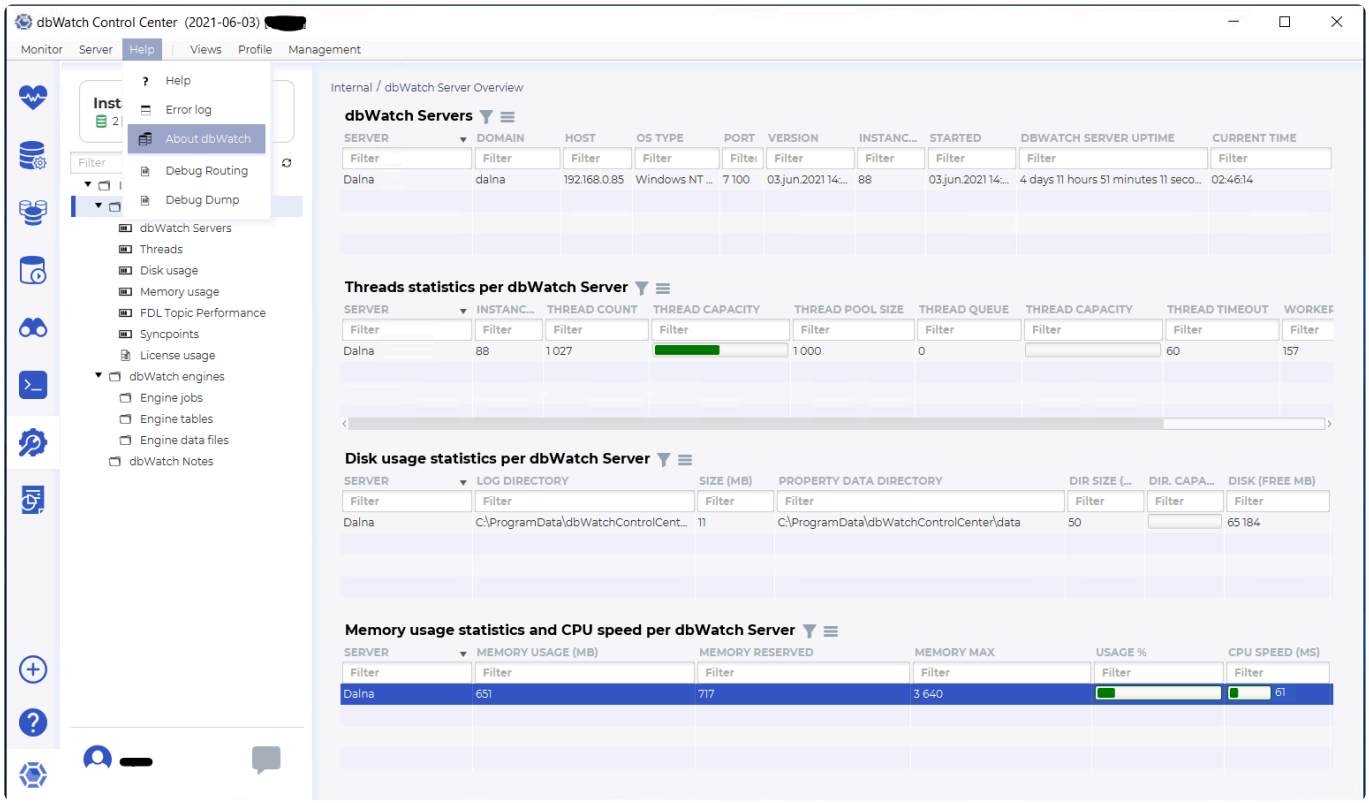
Find your version

Checking dbWatch Version

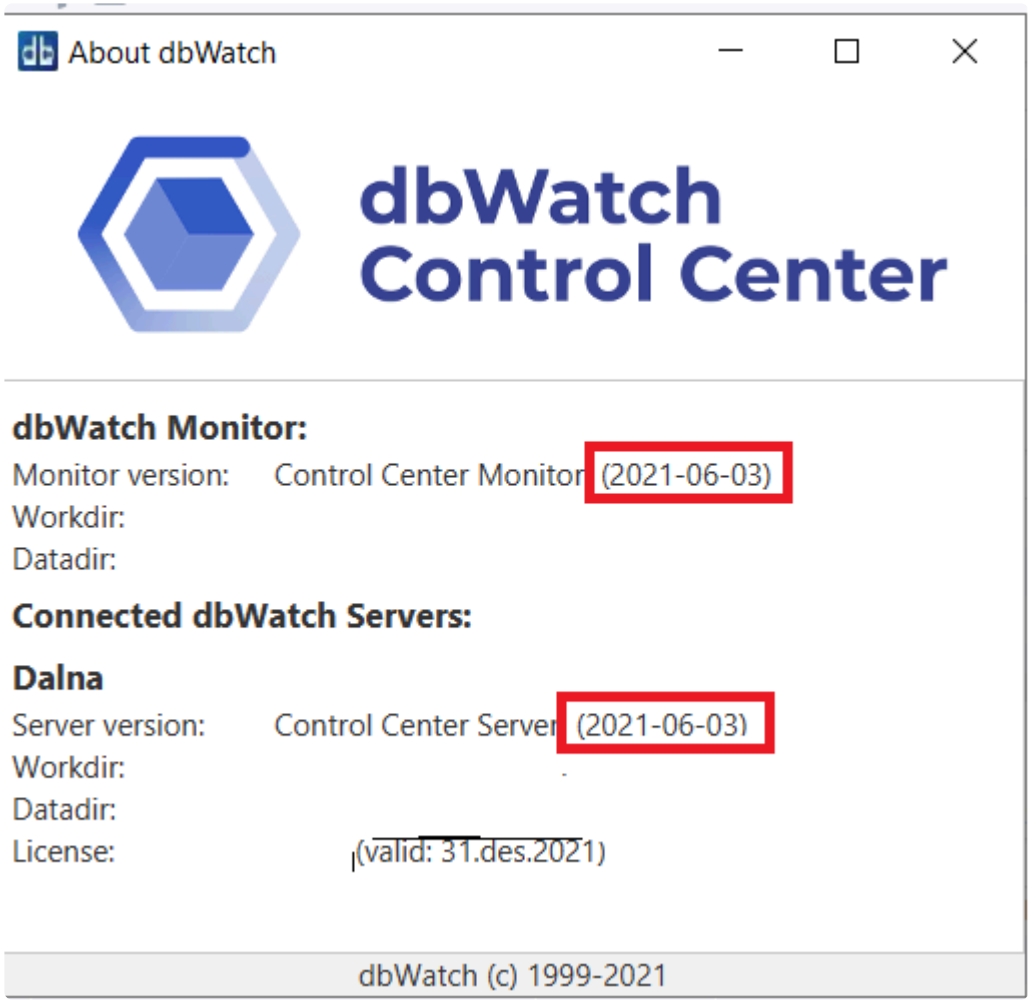
To check your current dbWatch version, you can:
1. See it in the upper portion of your application.



2. Open Help > About dbWatch or click on the dbWatch Icon:



You'll see a window showing you the current version of dbWatch Control Center.



[<< Release notes](#) / [Platforms supported >>](#)

Platforms supported

Platforms supported for the main product.

Current release supports the following platforms, versions, and setups:

On-premise support

Supports* Express edition, Standard edition, Standard edition one, Standard edition two, and Enterprise edition of the following versions:
Oracle 8i
Oracle 9i
Oracle 10g
Oracle 11g
Oracle 12c
Oracle 18c
Oracle 19c
Oracle 21c
*) Not all editions exist for all releases

Microsoft SQL Server 2000
Microsoft SQL Server 2005
Microsoft SQL Server 2008
Microsoft SQL Server 2008 R2
Microsoft SQL Server 2012
Microsoft SQL Server 2014
Microsoft SQL Server 2016
Microsoft SQL Server 2017
Microsoft SQL Server 2019
Microsoft SQL Server 2022

PostgreSQL 8.2
PostgreSQL 8.3
PostgreSQL 8.4
PostgreSQL 9.0
PostgreSQL 9.1
PostgreSQL 9.2
PostgreSQL 9.3
PostgreSQL 9.4
PostgreSQL 9.5
PostgreSQL 9.6
PostgreSQL 10
PostgreSQL 11
PostgreSQL 12
PostgreSQL 13
PostgreSQL 14

PostgreSQL 15
PostgreSQL 16

MySQL 5.1
MySQL 5.5
MySQL 5.6
MySQL 5.7
MySQL 8.0

MariaDB 5.1
MariaDB 5.2
MariaDB 5.3
MariaDB 5.5
MariaDB 10.x
MariaDB 11.x

Sybase ASE 12
Sybase ASE 15
Sybase ASE 16

MongoDB 4.x (beta)
MongoDB 6.x (beta)
MongoDB 7.x (beta)

Cloud support

Microsoft Azure SQL single database
- Supported but monitoring/management limited due to possibilities in underlying solution
Microsoft Azure SQL managed instance
Microsoft Azure Database for MySQL
Microsoft Azure Database for MariaDB
Amazon RDS Microsoft SQLServer 2014
Amazon RDS Microsoft SQLServer 2016
Amazon RDS Microsoft SQLServer 2017
Amazon RDS Microsoft SQLServer 2019
Amazon RDS Microsoft SQLServer 2022
Amazon RDS Oracle 19c
Amazon RDS MySQL 5.7
Amazon RDS MySQL 8.0
Amazon RDS MariaDB 10.3
Amazon RDS MariaDB 10.4
Amazon RDS MariaDB 10.5
Amazon RDS MariaDB 10.6
Amazon RDS PostgreSQL 10
Amazon RDS PostgreSQL 11
Amazon RDS PostgreSQL 12
Amazon RDS PostgreSQL 13
Amazon RDS PostgreSQL 14

Amazon EC2 – Same support as on-premise support

Platforms supported for specific modules.

SQL Performance module for MS SQL Server

Microsoft SQL Server 2014
Microsoft SQL Server 2016
Microsoft SQL Server 2017
Microsoft SQL Server 2019
Microsoft SQL Server 2022

SQL Performance module for Oracle

Oracle 11g
Oracle 12c
Oracle 18c
Oracle 19c
Oracle 21c

SQL Performance module for PostgreSQL

PostgreSQL 9.4
PostgreSQL 9.5
PostgreSQL 9.6
PostgreSQL 10
PostgreSQL 11
PostgreSQL 12
PostgreSQL 13
PostgreSQL 14
PostgreSQL 15
PostgreSQL 16

SQL Performance module for MySQL

MySQL 5.7
MySQL 8.0

SQL Performance module for MariaDB

MariaDB 10.2-10.11
MariaDB 11.x

Getting Started

Downloading the latest version of Control Center

You can get a download link to the latest version of dbWatch Control Center by registering for a free evaluation copy

Wait for dbWatch's Confirmation Email. Follow the steps and link in that email.

There are guides will help you install on different platforms:
Installing dbWatch on Windows
Installing dbWatch on Linux

There are tutorial videos, that go through the steps of installation and using the product on this Youtube playlist.

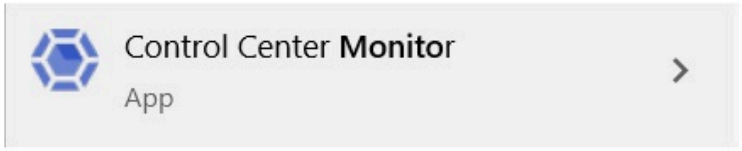
No Confirmation Email was received. What do I do?

Send us an email at sales@dbwatch.com to get your free evaluation email.

I already installed dbWatch Control Center. What to do next?

After installing dbWatch Control Center, click on the dbWatch Control Center Icon. You can either open the shortcut found on your desktop or search for it in your search bar.

dbWatch Control Center Shortcut



dbWatch Control Center in Search bar

After opening the application, two windows will appear.
Proceed with the guide on How to set up the domain controller.

<< Platforms supported / Installing on Windows >>

Installing on Windows

dbWatch Server Prerequisites

- Windows Server (VMWare virtual server or AWS EC2 supported)
- Windows Server 2008, 2012, 2016, 2019, and 2022
- Windows 10 or Windows 11
- 8 GB RAM
- 4 CPU cores
- 10 GB hard drive space

dbWatch Engine Prerequisites

- 500 Mb free space in each database instance
- SQL Performance module (extra cost) requires additional space, around 5 GB
- Engine – Server communication determined by each supported database platform.
- Bulk install for large database environments
- Installs in under 2 minutes per instance

dbWatch Client Prerequisites

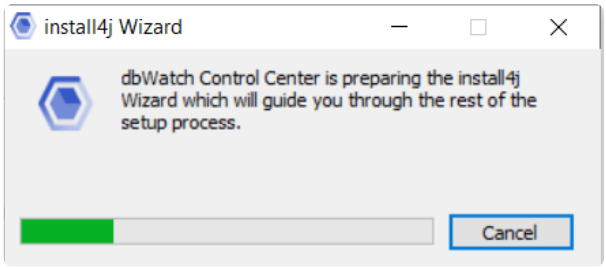
- Windows operating system with a graphical interface
- Windows Server 2008, 2012, 2016, 2019, and 2022
- Windows 10 or Windows 11
- 10 GB hard drive space
- 8 GB RAM
- 4 CPU cores
- Mouse
- Java support (Java runtime is included in the installation)
- Client-Server communication requires a single port only
- Client is installed automatically during Server installation

Download

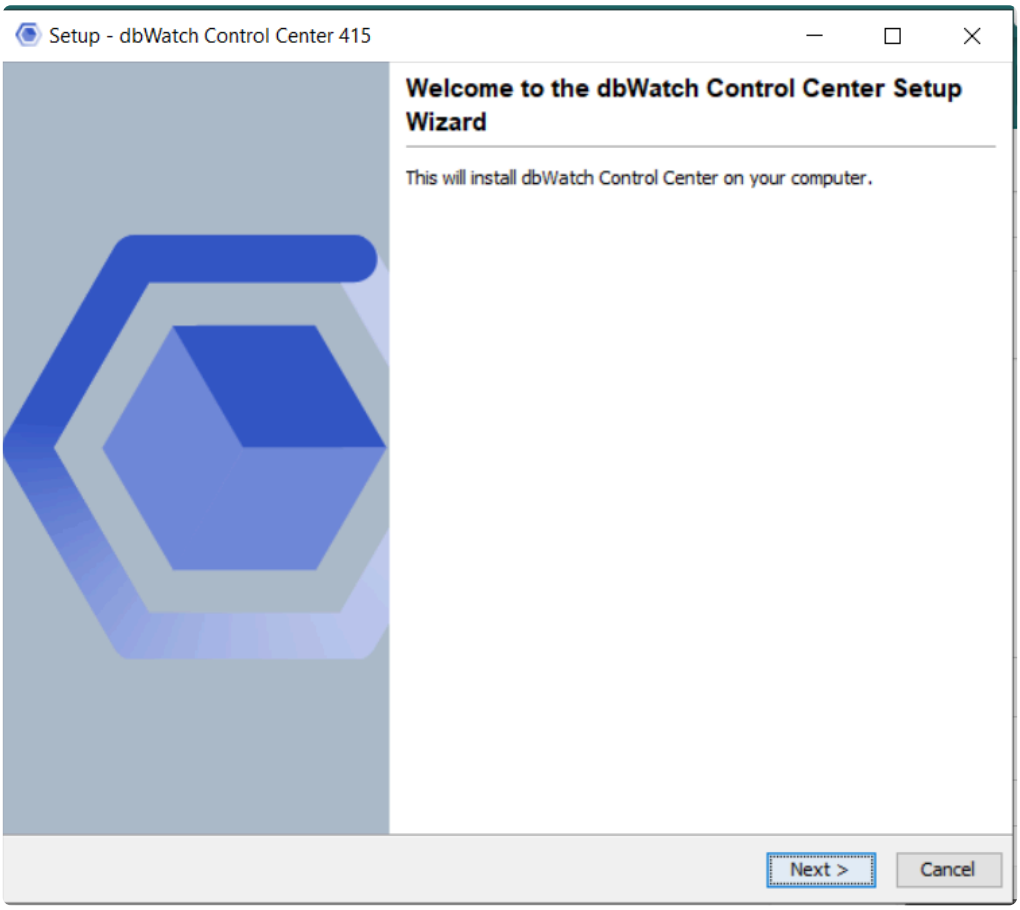
To download Control Center, register your email, and you will be redirected to our landing page. Click the link to be redirected to the Control Center's installer page: "dbWatch Control Center Release": Download the latest dbWatch Control Center by registering here

Step-by-step installation

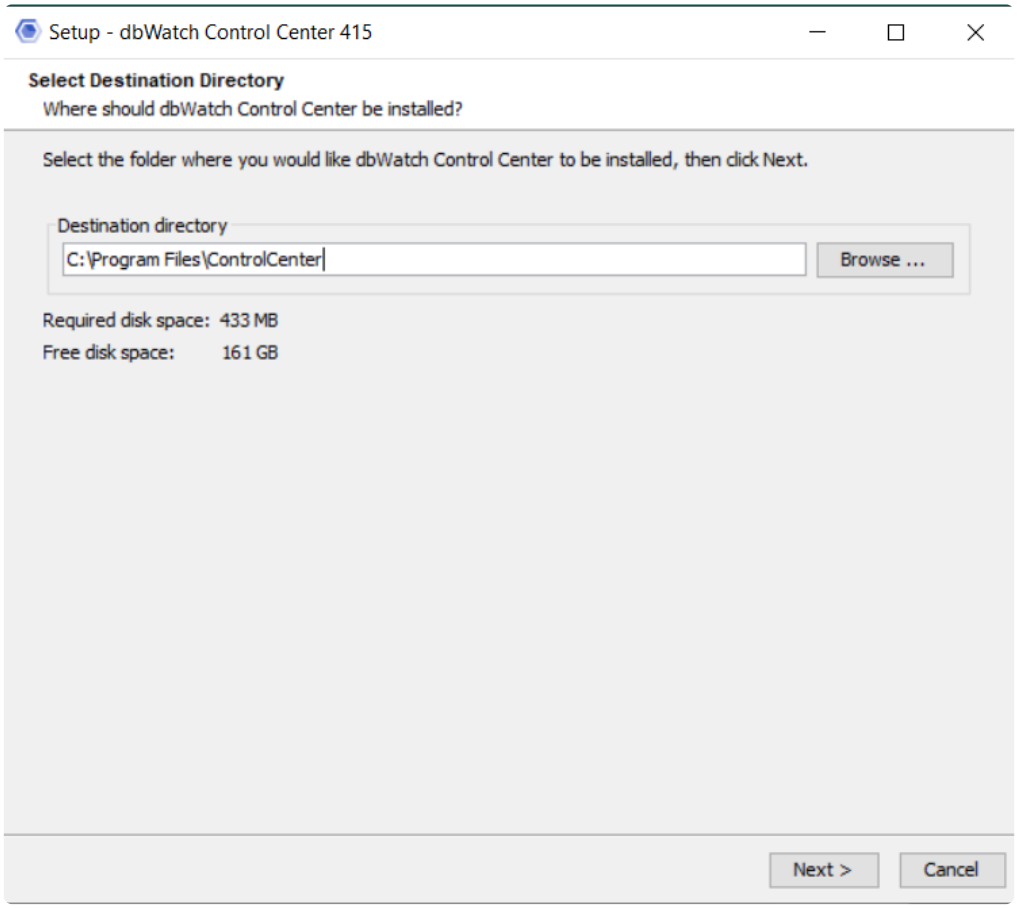
Open the download file and run the program. You will be greeted by the start screen of the Installation Wizard. Make sure you are running as Admin otherwise, this may block your system from installing the application. Alternatively, you can run the file via the command line.



After you do so, you will get the following window, where you see the opening page of the installer. Click Next.

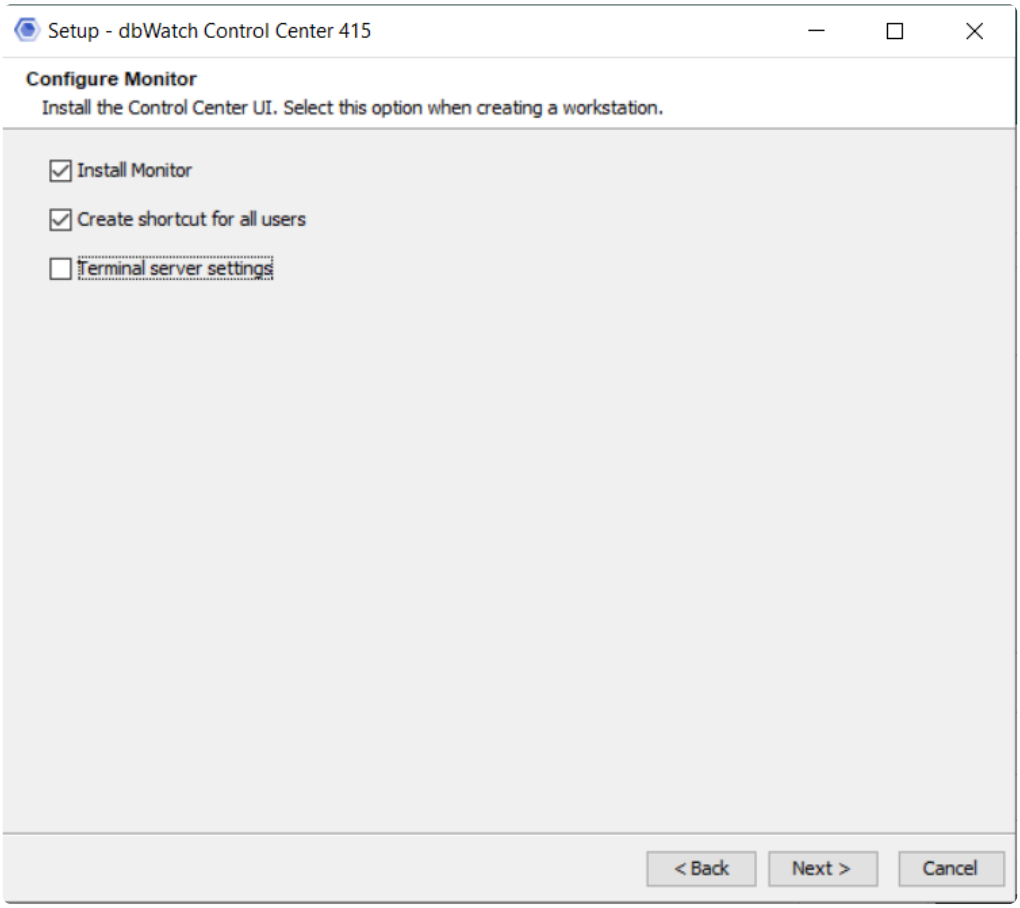


You'll arrive at the page where you can set the destination directory.



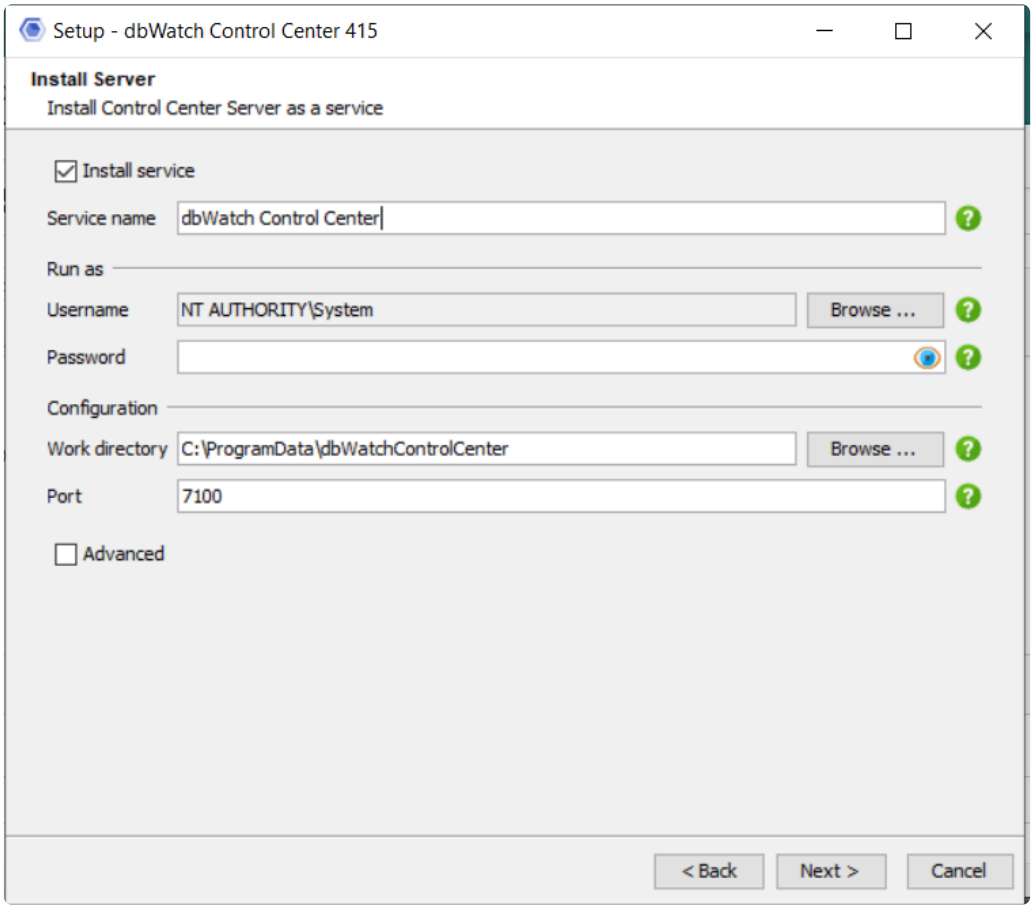
The previous installation of the Control Center will be replaced by the newer version.

After clicking “Next,” you will be directed to the next page of the installation process. Here, you can tick the selection you need.



- **Install Monitor** – Install the latest file path for *Control Center* monitor.
- **Create shortcut for all users** – Short cut for monitor will be available in your desktop
- **Terminal server settings** – Define domain and port for *Control Center*. Skip this when you don't need to define a port.

Select the installation options you want and click "Next".

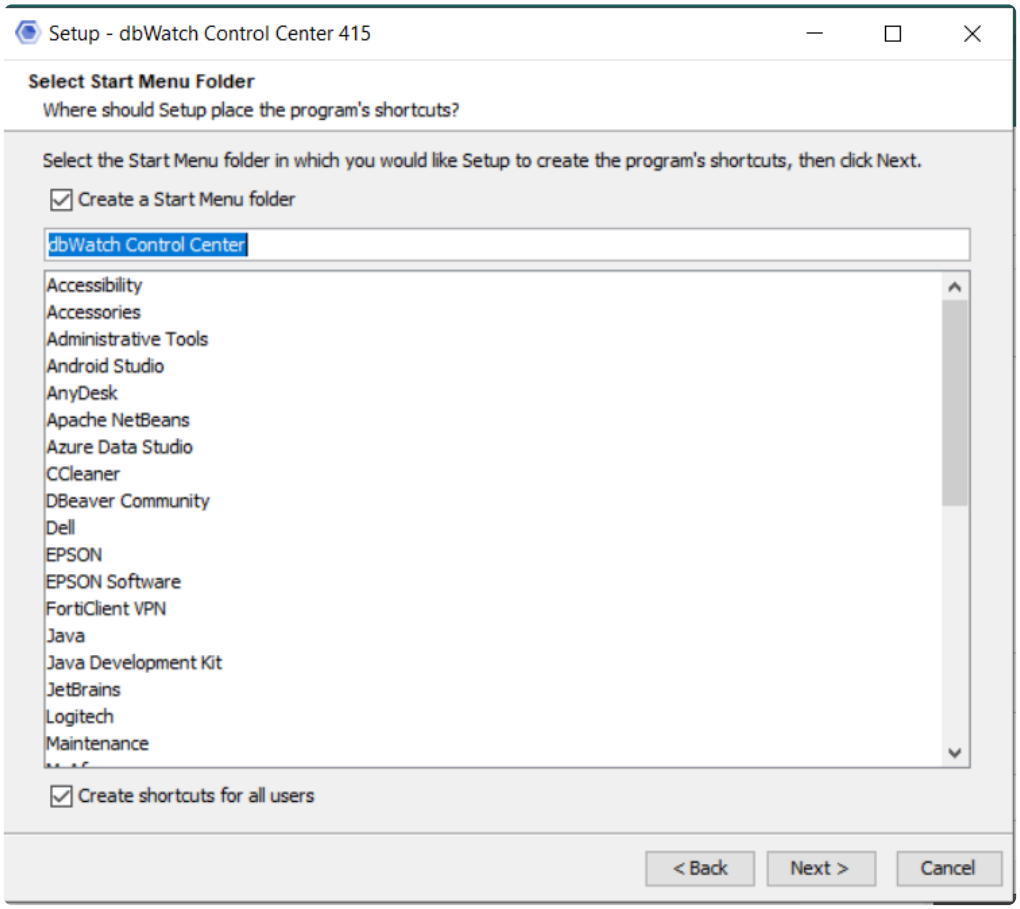


This is the window where you specify the credentials that will be used to run your dbwatch service. The dbwatch service account requires membership in the Performance Monitor Users group or equivalent privileges to access system performance data.

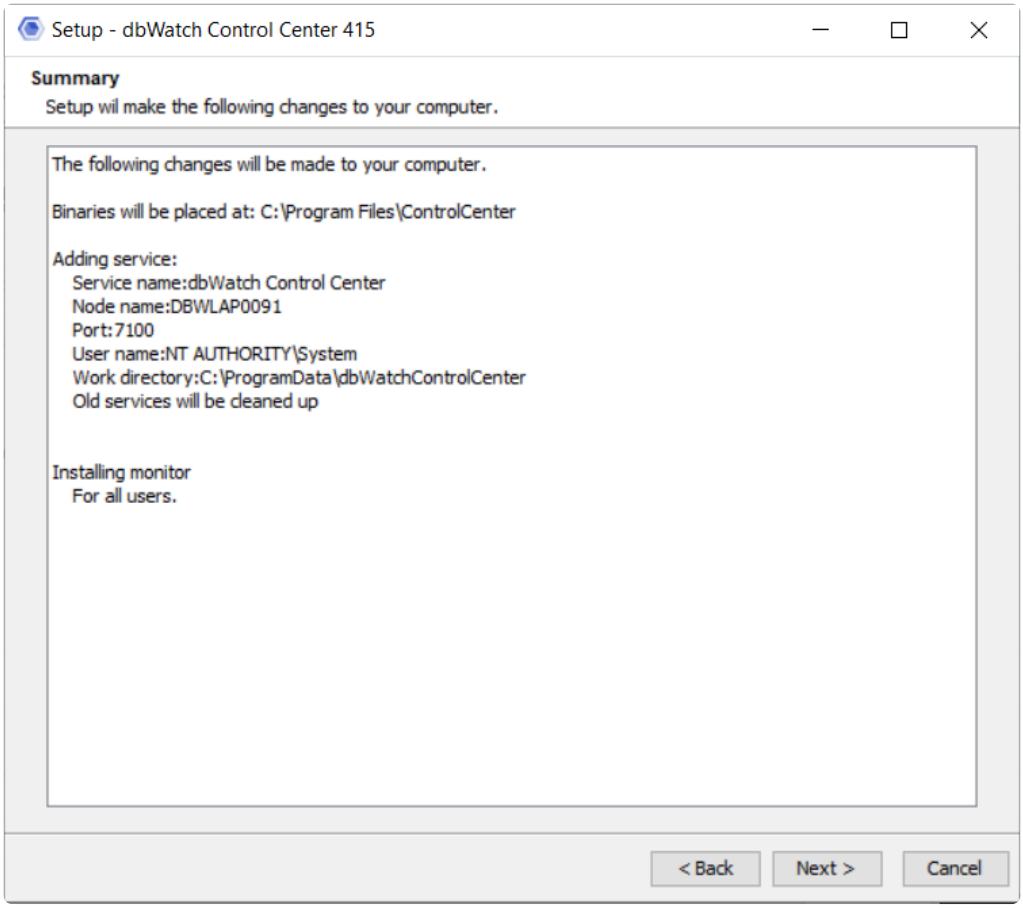
If you just want to install the client software, Control Center Monitor, you can untick “Install Service” here. It’s also possible to change the default port for ControlCenter, and configure the service for Control Center. By ticking “Advance”, two additional boxes and input fields show up. If you don’t need to specify the Service ID, you can skip that segment. If you don’t want to replace the old service running or start the Dbwatch Control Center, you can untick them.

> ! Always remember to keep the previous version of the monitor closed, as it causes problems when installing the service.

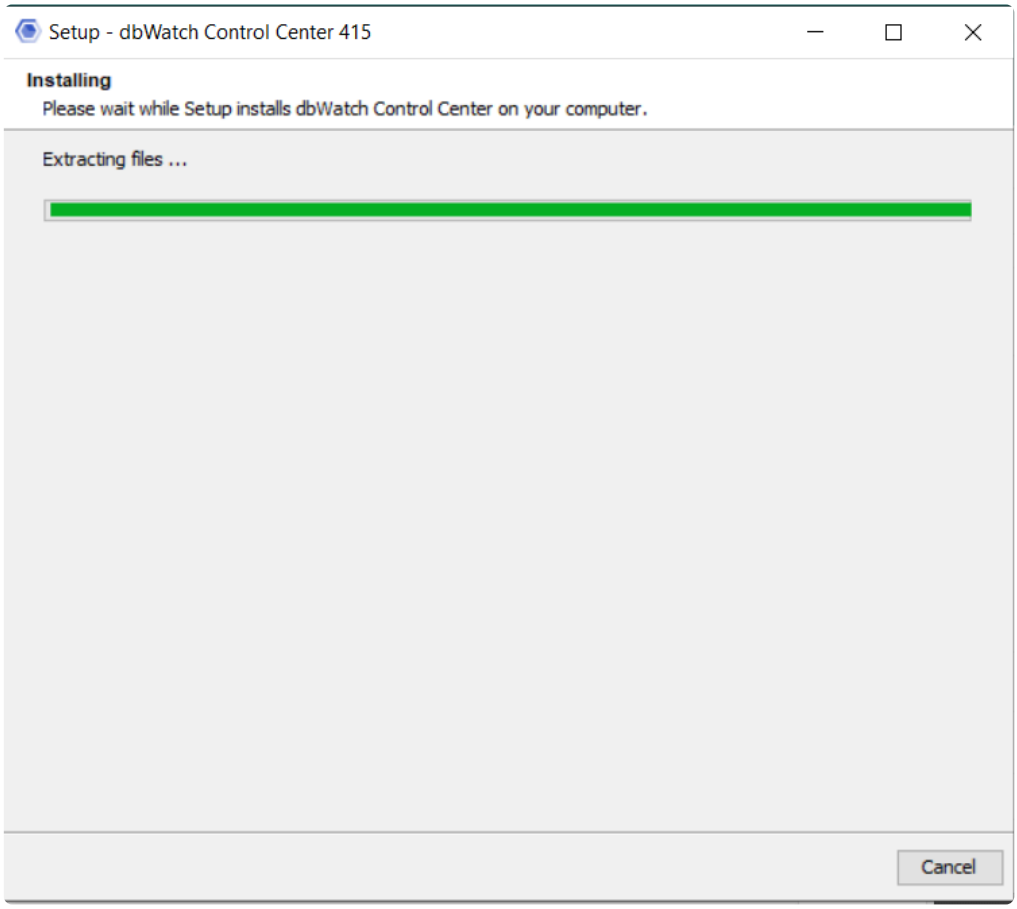
If you do not choose to create a service, the Server can still be started in console mode. Click “Next” to continue.



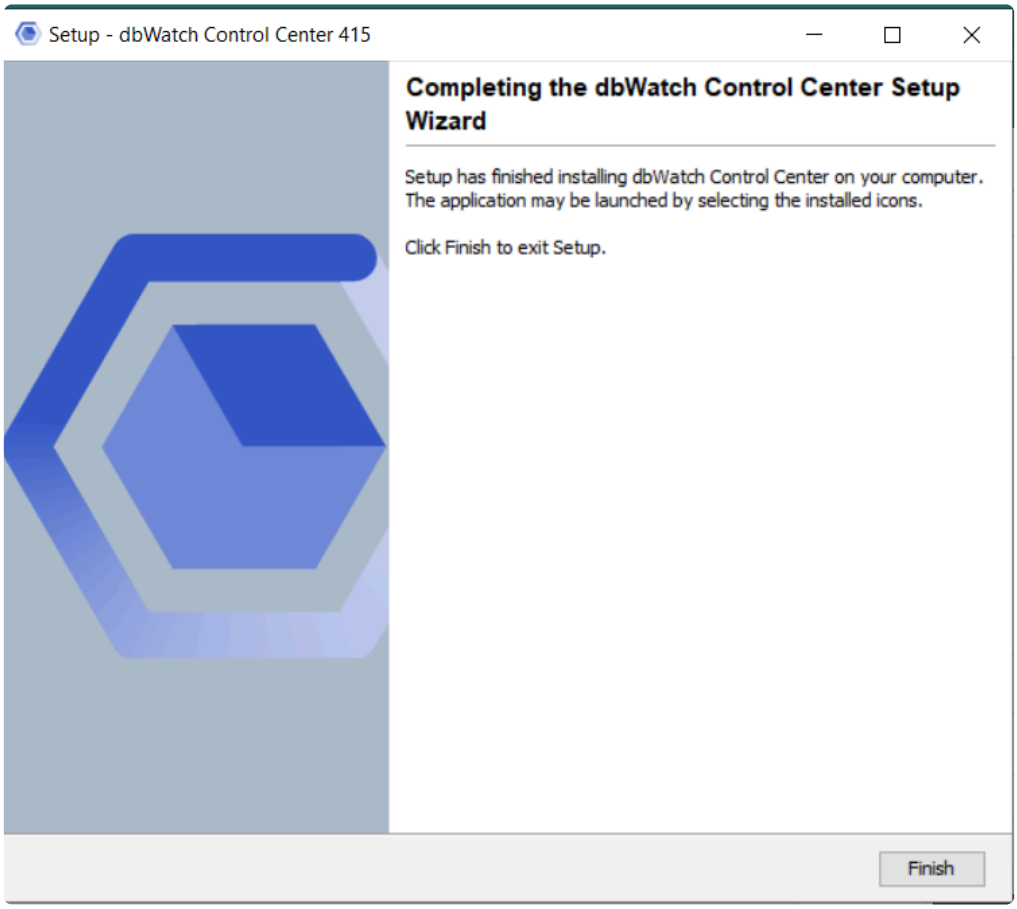
This page gives you the option to create a shortcut to your start menu. Untick the box if you don't want to create a shortcut. Click "Next" to proceed with the summary.



You are now ready to install dbWatch. Check the information displayed and click “Next” if they are correct.



The installation should take less than a minute on a typical computer.



The installation has now been completed successfully.

What To Do Next

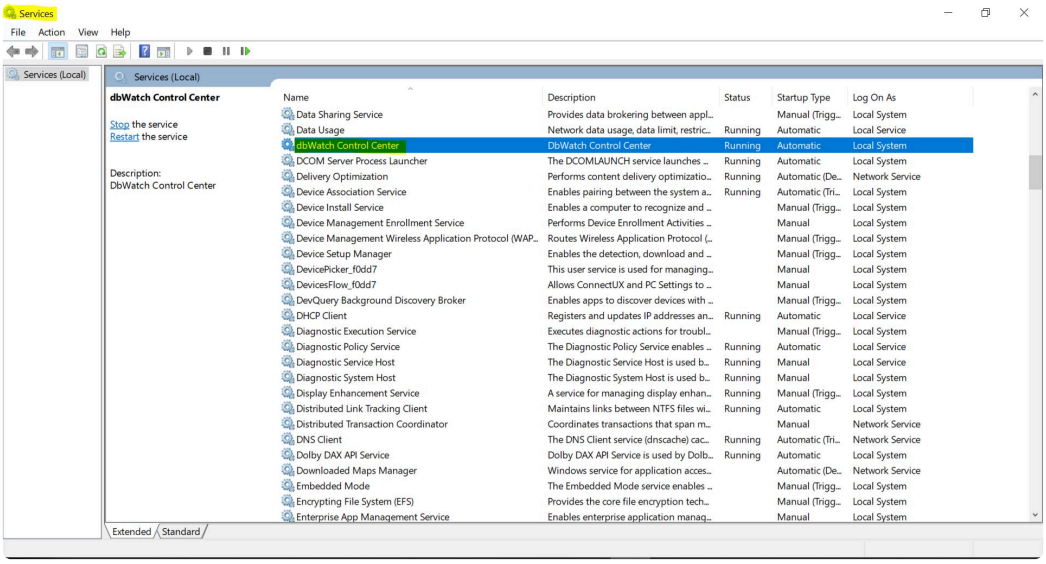
If you did not start the Monitor directly, you need to start it manually. This is located under “All Programs”->”dbWatch Control Center”->”Monitor ” in the start menu on Windows.

When the Monitor starts for the first time, it will launch the “DomainLogin” dialog and try to find a Server at localhost port 7100. If your Server is at a different host or port, enter the correct URL. When it is able to connect, it will display the “Configure Server” dialog.

When you start your new dbWatch installation, you will be prompted with the “Add database” wizard. For information about how to add a new database, see the “Instance management chapter under “Using Control Center”.

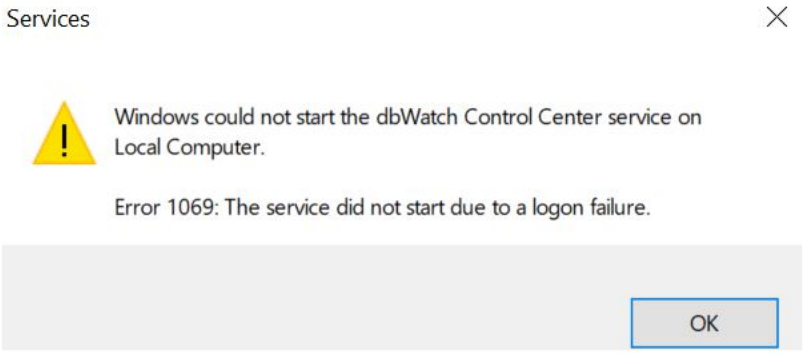
Quick Fixes

If you’re encountering service problems that relate to credential errors, here’s a quick way to deal with them. Open “Services” on your computer. You can type “Services” in the search bar.

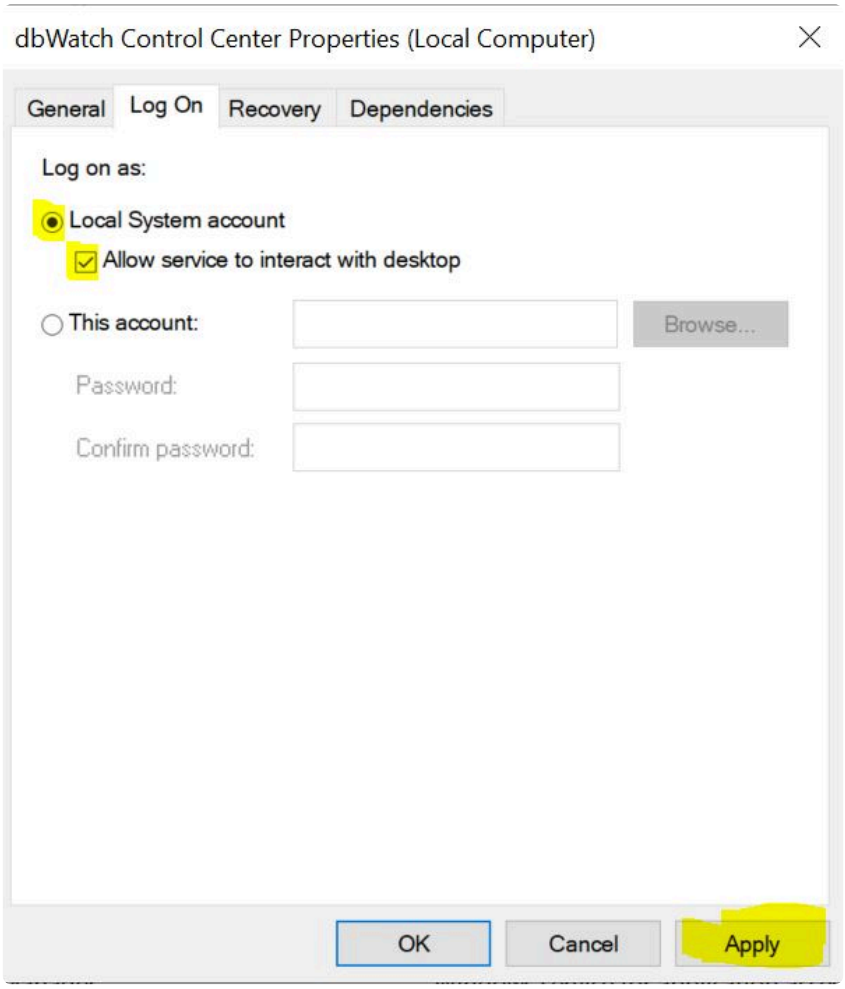


Start running the service. Highlight the equivalent of “Dbwatch Control Center”. This was defined earlier in your installation process.

Right-click and click “Start.



If this error above appears, the credentials you inputted were incorrect. To fix this, right-click on the highlighted services again and select properties. A window of properties will show up. Select the tab “Log On” then select Log on as Local System account. This only works if CC is installed locally.



Click “Apply” and re-run the services. You should be able to start your dbWatch services without a problem.

Once it is installed, the next step is Initial domain setup

<< Getting Started / Installing on Linux >>

Installing on Linux

dbWatch Server Prerequisites

Installation of dbWatch on Linux requires X11 for the graphical installation. Installation executable will fallback to console installation if X11 is not available, otherwise console installation is available with -c switch after executable.

- Linux Server (VMWare virtual server or AWS EC2 supported)
- Ubuntu 20.04 LTS (other versions and types may work)
- 8 GB RAM
- 4 CPU cores
- 10 GB hard drive space

dbWatch Engine Prerequisites

- 500 Mb free space in each database instance.
- SQL Performance module (extra cost) requires additional space, around 5 GB
- Engine – Server communication determined by each supported database platform
- Bulk install for large database environments
- Installs in under 2 minutes per instance

dbWatch Client Prerequisites

- Linux operating system with a graphical interface
- Ubuntu 20.04 LTS (other versions and types may work)
- 10 GB hard drive space
- 8 GB RAM
- 4 CPU cores
- Mouse
- Java support (Java runtime is included in the installation)
- Client – Server communication requires single port only
- Client is installed automatically during Server installation

Download

To download Control Center, register your email and you will be redirected to our landing page. Click the link to be redirected to the Control Center's installer page: "dbWatch Control Center Release": Download latest dbWatch Control Center by registering here
h2. Step by step installation

Installation steps

Special considerations on some Linux distributions

If you are not using IPV6 and want to use IPV4, you will need to disable IPV6 on the Linux server

running dbWatch Server.

To disable ipv6 on Ubuntu 20.04 LTS:
Edit /etc/default/grub and append ipv6.disable=1 to GRUB_CMDLINE_LINUX and GRUB_CMDLINE_LINUX_DEFAULT.

```
GRUB_CMDLINE_LINUX_DEFAULT="ipv6.disable=1"
GRUB_CMDLINE_LINUX="ipv6.disable=1"
```

and then update-grub

Then add the following lines in /etc/sysctl.conf

```
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6=1
```

And reboot.

You can also disable ipv6 in netplan with the entry link-local: [ipv4]: cat /etc/netplan/50-cloud-init.yaml

```
# This file is generated from information provided by
 # the datasource.  Changes to it will not persist across an instance.
 # To disable cloud-init's network configuration capabilities, write a file
 # /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
 # network: {config: disabled}
 network:
     ethernets:
         ens160:
             link-local: [ ipv4 ]
             dhcp4: true
     version: 2
```

Once it is installed, the next step is Initial domain setup

<< Installing on Windows / Console installation on Ubuntu >>

Console installation on Ubuntu

Before You Start

Before installing the dbWatch Control Center, note that the installer requires **root** privileges. For simplicity, this documentation demonstrates steps executed as the **root** user. You may use ‘sudo’ as an alternative.

This guide is for Ubuntu Server 20.04.5 LTS with a minimal setup, including an OpenSSH server. OpenSSH is optional unless remote access is required.

Adding Apt Repositories

Two trusted repositories must be added:

1. **dbWatch Release Repository:** Provides the latest dbWatch Control Center.
2. **Bellsoft Repository:** Hosts the required Java version.

Adding the Bellsoft Repository

Bellsoft’s installation guide is available here. For quick setup, execute the following commands:

```
wget -q -O - https://download.bell-sw.com/pki/GPG-KEY-bellsoft | sudo apt-key
add -
echo "deb [arch=amd64] http://apt.bell-sw.com/ stable main" | sudo tee /etc/ap
t/sources.list.d/bellsoft.list
```

Adding the dbWatch Release Repository

Execute these commands to add the dbWatch repository:

```
wget -q -O - https://download.dbwatch.com/release/archive.key | sudo apt-key a
dd -
echo "deb [arch=amd64] https://download.dbwatch.com/release/ stable main" | su
do tee /etc/apt/sources.list.d/dbwatch.list
```

Installing the Software

Update packages and install the dbWatch Control Center:

```
sudo apt-get update
sudo apt-get install dbwatch-controlcenter
```

Starting the Service

Start the dbWatch Control Center service with:

```
sudo service dbwatch-controlcenter start
```

After installation, proceed to Initial Domain Setup.

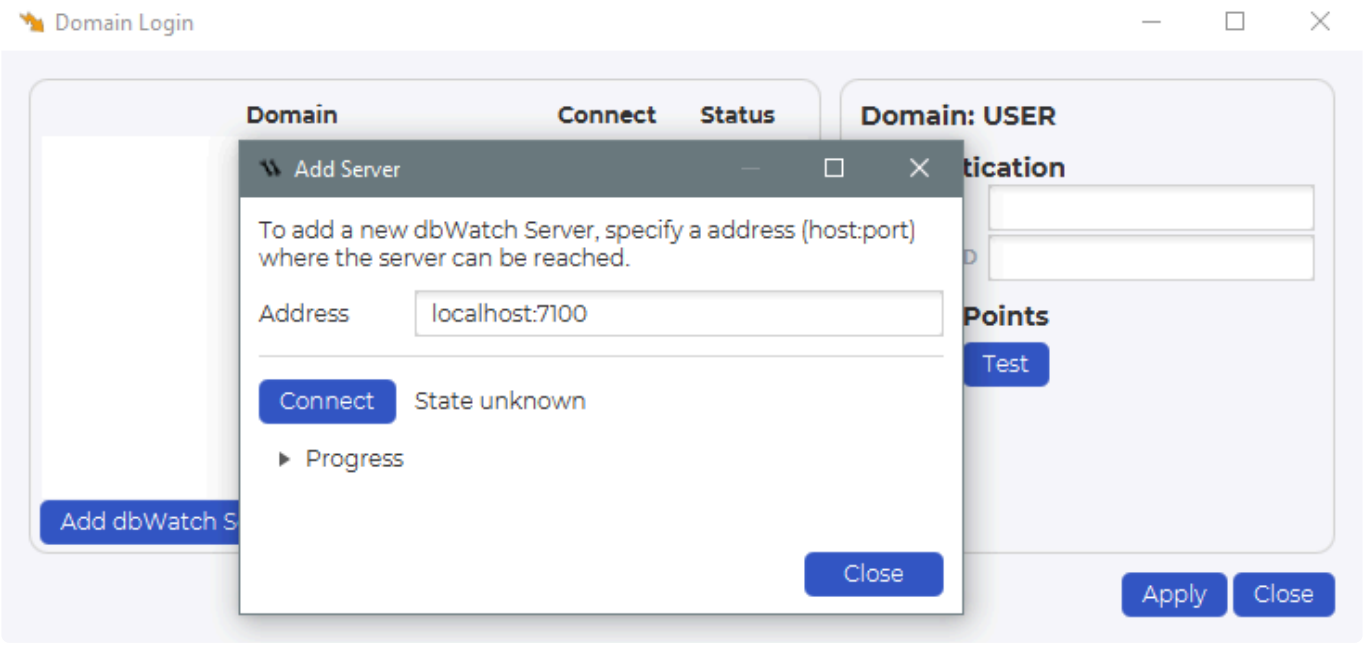
<< Installing on Linux / Initial Domain Setup >>

Initial Domain Setup

All dbWatch Servers are part of a domain.

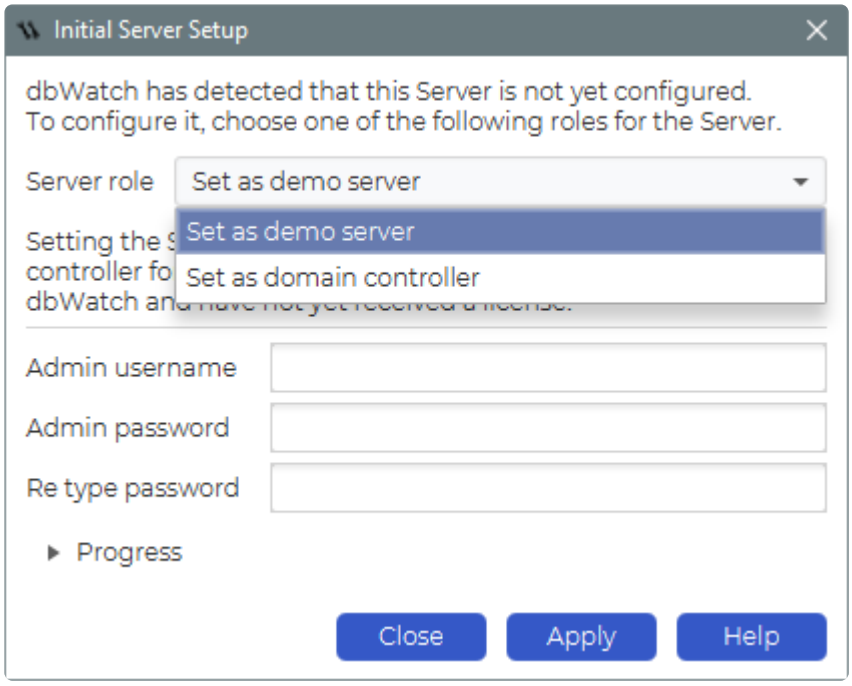
This domain is unique for each customer and it is specified in the dbWatch license. Every dbWatch installation must have one of the dbWatch Servers specified as the Domain Controller.

The first time a you start the dbWatch Monitor you are prompted for the address where the dbWatch Server is listening. The default port is 7100.
Enter the address and click “Connect”.

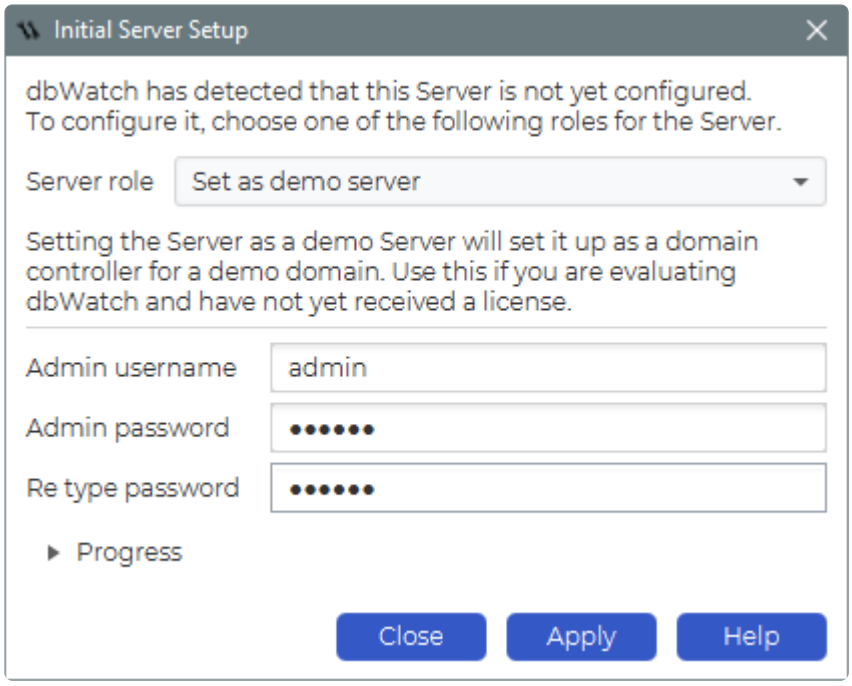


If the dbWatch Monitor is able to connect to the dbWatch Server, you are given 2 choices to configure the Server.

- Set as demo server
  Use this if you are evaluating dbWatch, and have not yet recevied a license. This will create a demo license for you.
- Set as domain controller
  Use this if you have a license and want to configure a production domain controller.

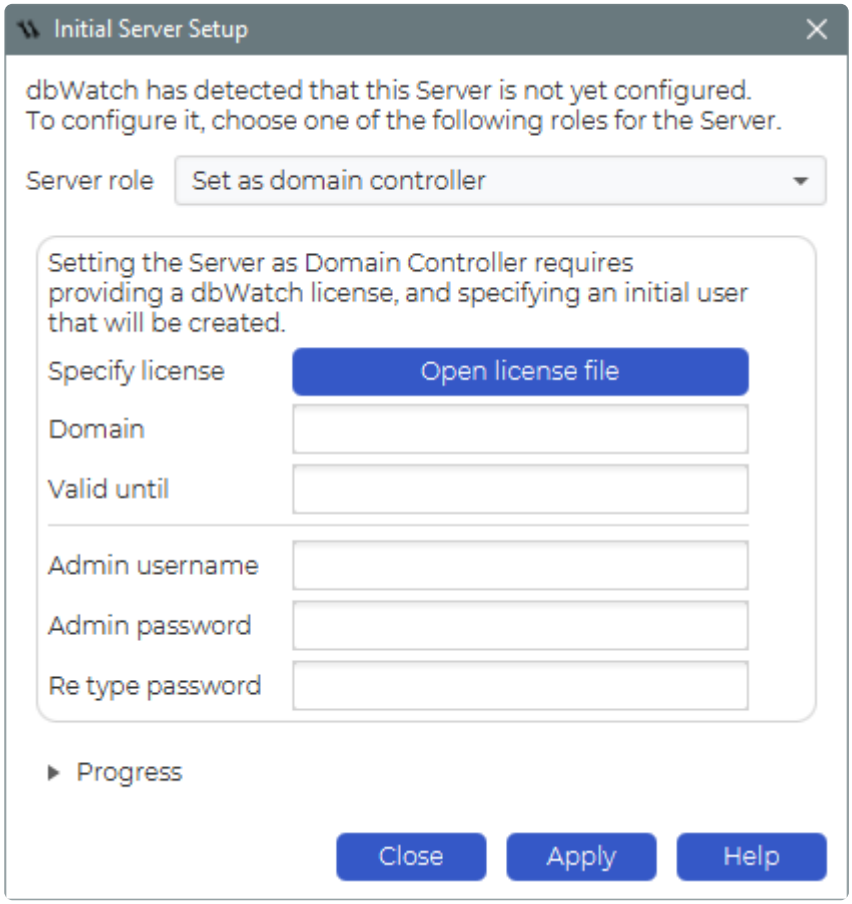


Set as demo server



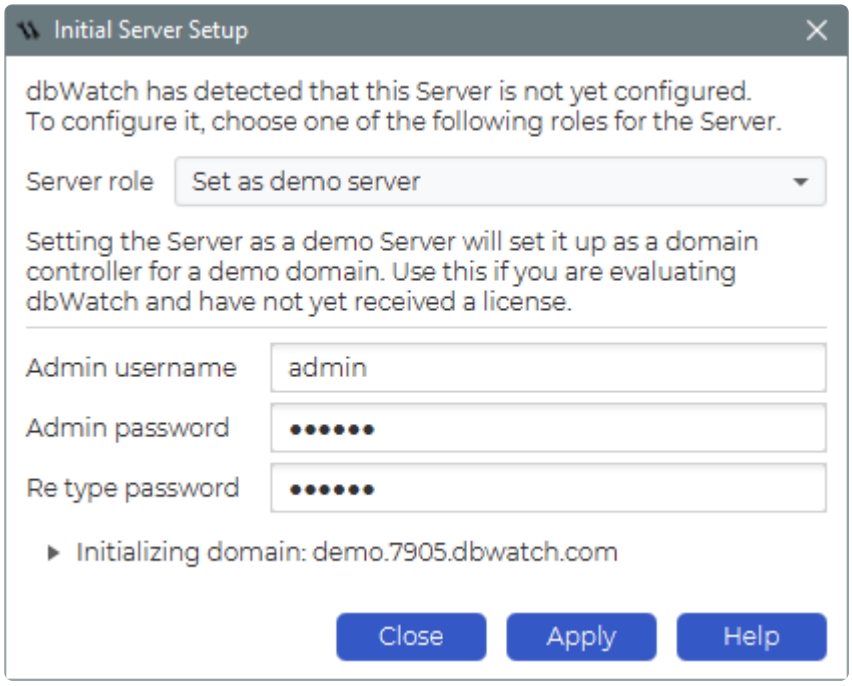
If you want to setup a demo server, select this option, and fill inn a username and password. This will be used to create an admin user in dbWatch.
This user will have the “Domain Admin” role, wich means that the user will be allowed to edit the security settings and define new users.

Set as domain controller

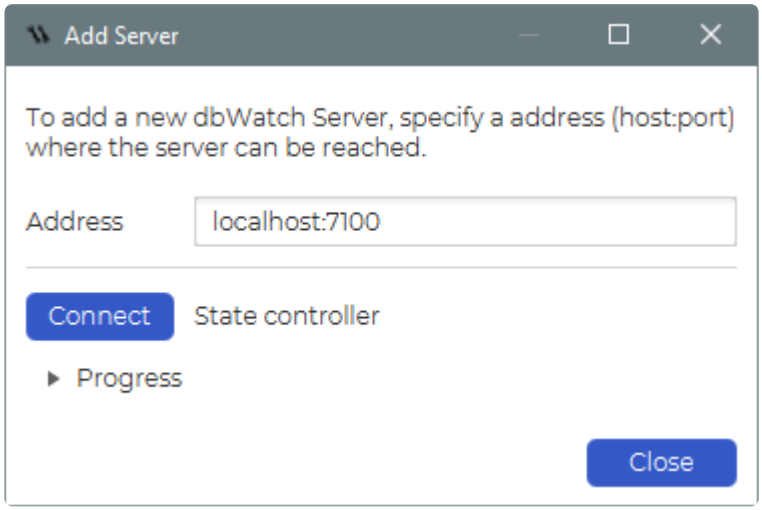


If you have a license and want to setup a domain controller, select this option.
Click on “Open license file” and select your license. If the license is valid the “Domain” and “Valid until” fields will be filled.
Then specify a username and password, this will be used to create an admin user in dbWatch. This user will have the “Domain Admin” role, wich means that the user will be allowed to edit the security settings and define new users.

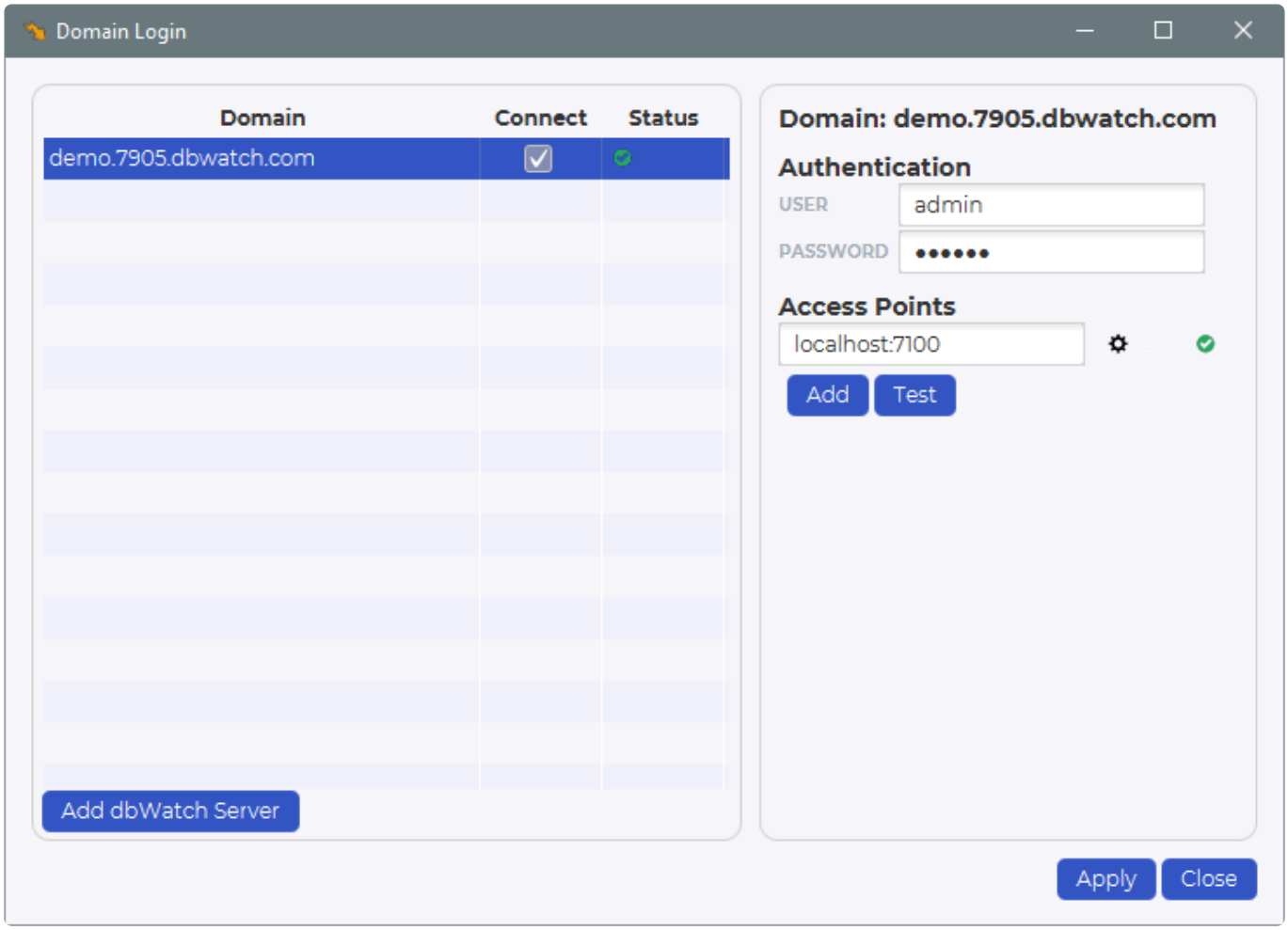
Next click on “Apply”. The dbWatch Server will be initialized as either a demo server or a licensed server, depending on your choice.



When the initialization process is complete, wait untill the “Add Server” dialog has a “State controller” status.



Close this window.



When everything is ready, the “Status” icon for the domain should turn green.

You are now ready to register your database instances in dbWatch Control Center.

<< Console installation on Ubuntu / Product Overview >>

Product Overview

Product Overview

In Product Overview, we will look into architecture and structure and give an overview of how the application operates.

Chapters:

- Architecture
- File Structure
- Network and communications
- Database Operations Philosophy
- Monitor Dashboards and Overview Screens
- Dashboards and Overview Screens

[<< Initial Domain Setup] / [Architecture >>]

Architecture

Architecture components

The dbWatch installation consists of three components:

1. **dbWatch Monitor/Client** serves as the front-end GUI
2. **dbWatch Server** is the core business logic and central hub of the system
3. **dbWatch Framework** contains stored objects in the registered database instances.

dbWatch Monitor

The dbWatch Monitor is the GUI front end for interacting with one or several dbWatch Servers. The Monitor is an installed application.

dbWatch Server

The dbWatch Server is the program responsible for all communication with the configured database instances. It is responsible for triggering the monitoring procedures (if monitoring is enabled), generating reports, and performing the actual administration tasks. The dbWatch Server can also communicate with 3rd party products through extensions. In more advanced environments, the dbWatch Server can be configured to have one or more roles configured. The roles available depend on the license and features enabled.

dbWatch Server roles

When configured in more advanced environments, role separation is possible. There are several potential roles and a dbWatch Server can have one or more of them:

Instance hub

This role is for connecting to and monitoring database instances. A configuration of multiple dbWatch Server nodes can have one or more nodes with the role of instance hub to monitor multiple security zones, aid scalability, or provide fail-over assistance.

Domain CA

This role is typically the first to be configured, as the domain CA is the node that handles the configuration of other nodes and the security.

Storage

This node is assigned for storage, typically for USL (Unified Scripting Language). This node will hold the actual data if a script needs to temporarily or permanently store data.

Cloud router

This node is used to route secure communication between dbWatch Server nodes. Typical usage is when a dbWatch node can only make outbound connections, and you need a common connection point for the GUI client.

Host node

This node is used for USL to perform scripts interacting with the underlying host. It is used for integrations and USL scripting but can also work as an agent to allow for host monitoring.

Webserver

This node provides web dashboards and data exports populated with data from FDL (Farm Data Language) queries across the environment.

Update

This node provides updates for the automatic update functionality for other nodes in the environment. May have internet access to download updates automatically or provide manually downloaded updates for the other dbWatch nodes.

Database Instances

There are four dbWatch modules that can be enabled for each database instance, the following briefly describes their impact on the instance;

Monitoring

When you enable monitoring for a particular instance a dbWatch Engine is installed on the instance. The Engine is a set of database objects (Procedures, Tables & Functions) written in the native language related to that instance. This code provides the interface used by the dbWatch Server when triggered by the dbWatch Tasks and Alerts. It is also used by Tasks and Alerts to report back to the dbWatch Server. All engine codes are open for end users to view and are included with the installation of the dbWatch Server. The dbWatch Server maintains a pool of up to 4 sessions that perform the monitoring of each instance.

Management

The dbWatch Management module consists of a set of specifications that define the management interface your user has for a particular instance. A specification can optionally define a set of “helper” objects on the database instance. This will usually entail several tables used for temporary storage and functions that perform codes that are used often. If the selected management specification contains such objects, they are installed on the instance the first time the management interface is used and this then is Management.

Framework

The dbWatch Server has a pool of sessions for each instance that will grow on demand, based on the user commands issued in the Management interface. This will hardly ever grow beyond 1 or 2 sessions and when the user is inactive for a while (30 seconds default) sessions are terminated.

SQL Worksheet

The dbWatch SQL Worksheet allows you to perform custom SQL statements on any database instance. It has no associated database objects. The dbWatch Server has a pool of sessions for each instance that will grow on demand based on the SQL statements issued in the SQL Worksheet. This will hardly ever grow beyond 1 or 2 sessions. When the user is inactive for a while (30 seconds default) sessions are terminated.

Reports

The dbWatch Reports allows you to generate reports based on any data on any database instance. It has no associated database objects.
The dbWatch Server will create a session for each (global) report it generates and terminates it afterward. On a normal system, this will result in no more than a handful of sessions being generated per day.

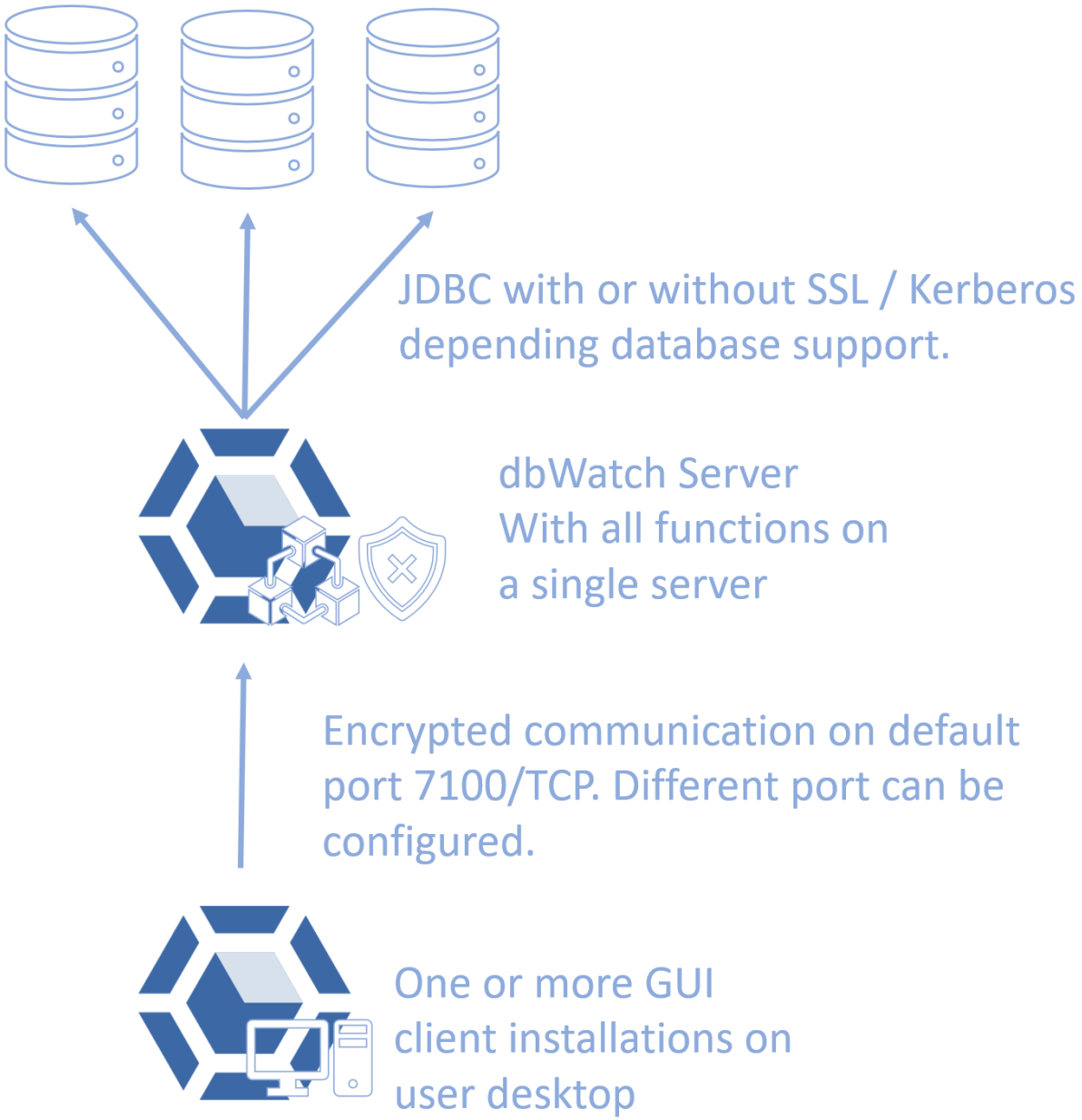
<< Product Overview / Example architecture designs >>

Example architecture designs

Basic setup

The most basic setup is where a dbWatch Server service is configured with all functions in the same node, and the client is either local or installed on a separate computer.

Example drawing:



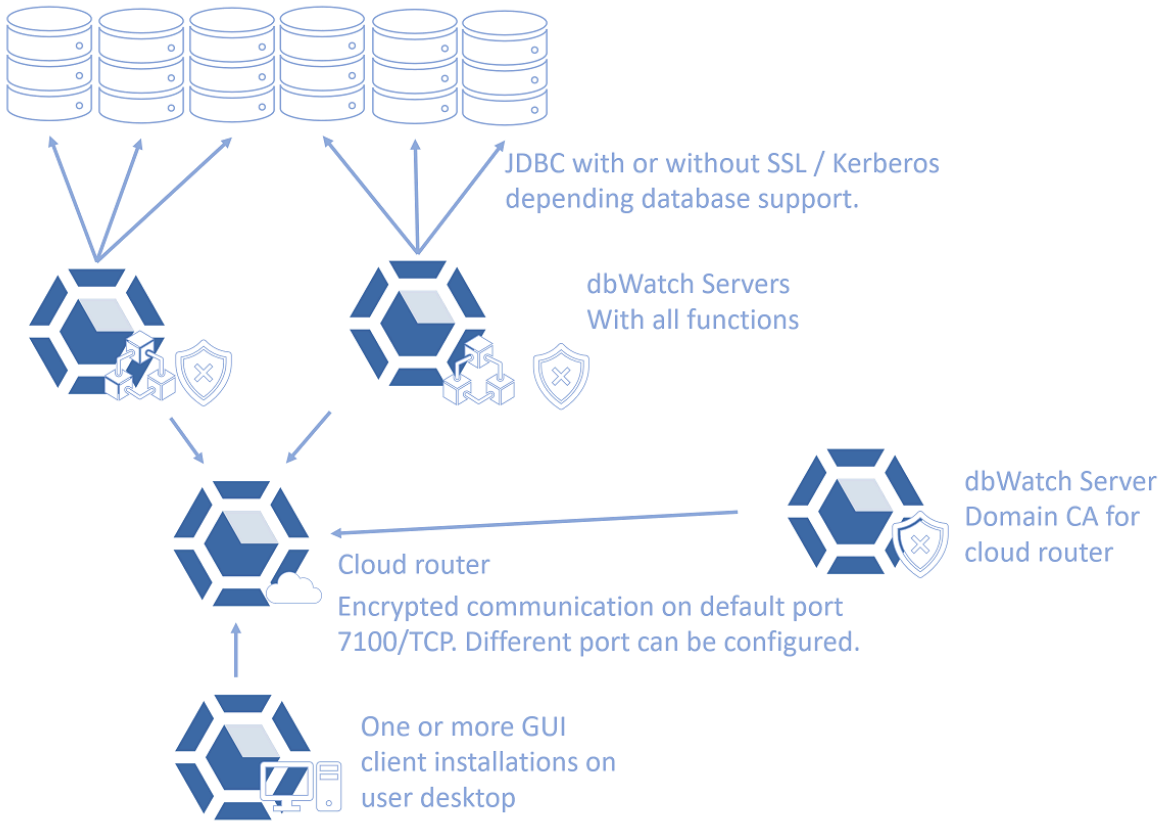
The pictured architecture is similar to the design and setup of an environment using dbWatch 12 and should be a drop-in replacement. Multiple clients can connect, suitable for configurations where databases are in a single security zone and network connections are uncomplicated.

Multiple dbWatch security domains with shared connection

hub

The pictured architecture is a setup where multiple setups, like the basic setup, are, combined with a shared connection hub, called a cloud router, to give clients one common connection point to reach all environments. The initial connection direction is reversed, so the dbWatch Servers connect outbound from their network to the cloud router. No firewall opening is needed to allow communication to the dbWatch Server. This setup is intended for service providers that service many smaller customers or a company with separate data centers or security zones.

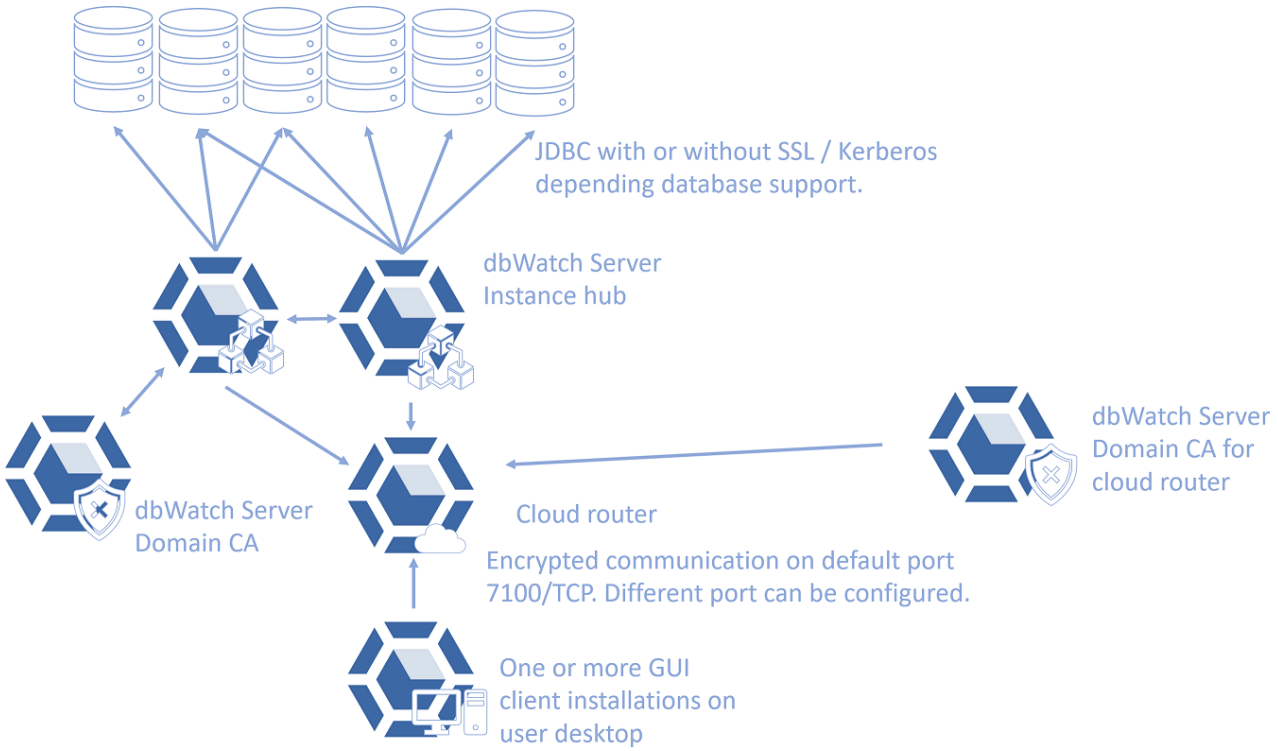
Example drawing:



Multiple monitoring servers in the same security domain

Suppose the environment consists of more than 250 database instances, or you want load balancing between dbWatch Servers monitoring database instances (Instance hubs). In that case, you might want to separate the instance hub node into multiple dbWatch Server installations. This setup requires the domain CA node role. This node controls the security and configuration of the security domain to be separate from the instance hubs, the systems that monitor database instances. You can add more instance hubs to this setup when you require additional capacity. A cloud router can facilitate a common connection point but is not needed, as any dbWatch server in a security domain configured for incoming connections can serve as a connection point.

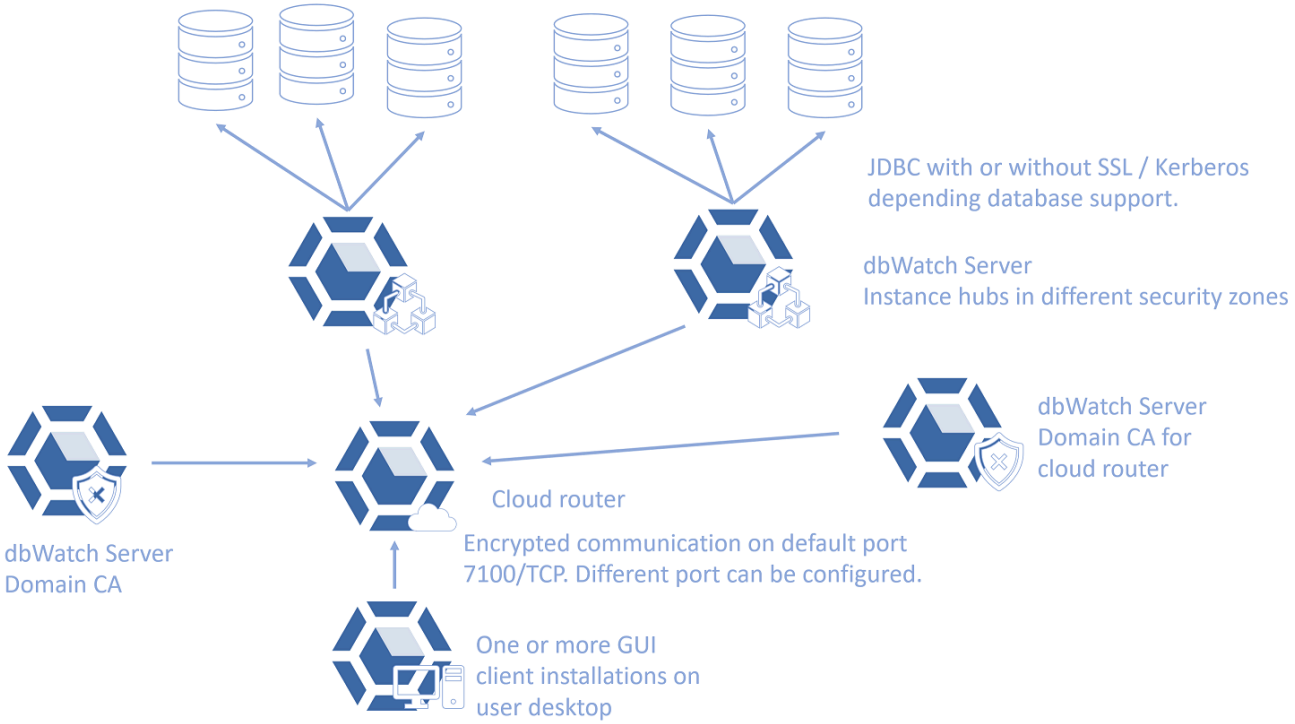
Example drawing:



Multiple security zones without direct connection or behind firewalls

In a scenario where no direct connection is allowed between the dbWatch domain CA node and any of the Instance Hubs, a dbWatch Cloud Router can provide a secure connection point. It could be in a case where a Managed Service Provider wants to monitor several customer locations in the same setup, without a VPN and without opening ports in the customer firewall due to the increased attack vector this creates. A dbWatch Cloud Router is a specialized hardened dbWatch Server, more info on this here

Example drawing:



Custom configurations

dbWatch Control Center is designed to work in all network and security setups we have seen in our customer environments, so the configuration possibilities are extensive. dbWatch Server node roles can be separated or combined. Connections can be one or two ways, in any direction on a port of choice. Security domains can be used to separate customers and security zones.

<< Architecture / Cloud router >>

Cloud router

Introduction

The dbWatch Control Center is structured around a network of nodes dedicated to managing server infrastructure. While these nodes effectively communicate within a single network, cross-network communication poses a unique challenge. To address this, we introduce the Cloud Router – a specialized node whose primary role is to securely route messages between different networks, ensuring the privacy and integrity of each domain is maintained.

Overview of Cloud Router

The Cloud Router is a pivotal element in the dbWatch architecture, designed to enable secure and efficient communication between different network segments. Uniquely positioned outside the individual networks, these routers eliminate the necessity for inbound firewall openings. This strategic placement not only enhances network security but also streamlines network management by simplifying the challenges associated with routing messages between networks.

A key aspect of the Cloud Router's functionality is its heavy reliance on encryption protocols. These protocols are integral in controlling and securing the communication channels, ensuring that only authorized networks can interact with each other. This level of security is paramount in maintaining the integrity and privacy of each network domain while facilitating necessary inter-network connections.

Key Features

- **Untrusted by Design**: The Cloud Router is engineered with the assumption that it could potentially be compromised. This "untrusted by design" approach ensures robust security measures are in place. By utilizing ephemeral keys and implementing comprehensive encryption for all communications, the system minimizes risks associated with compromised nodes. This design philosophy ensures that even in less secure environments, the integrity of message routing remains uncompromised.
- **Reduced Need for Inbound Rules**: By positioning Cloud Routers outside individual networks, the need for configuring inbound firewall rules is significantly reduced, enhancing overall network security.
- **Streamlined Inter-Network Communication**: Facilitates the management of message routing across various network segments, making the process more efficient and less complex.
- **Network Resilience**: The Cloud Router network is structured to enhance reliability. By deploying a series of Cloud Routers, the system ensures message delivery even in instances where some routers encounter issues. This redundancy and resilience are key to maintaining continuous and reliable communication across the network, ensuring that critical messages are always routed efficiently, regardless of individual node disruptions.

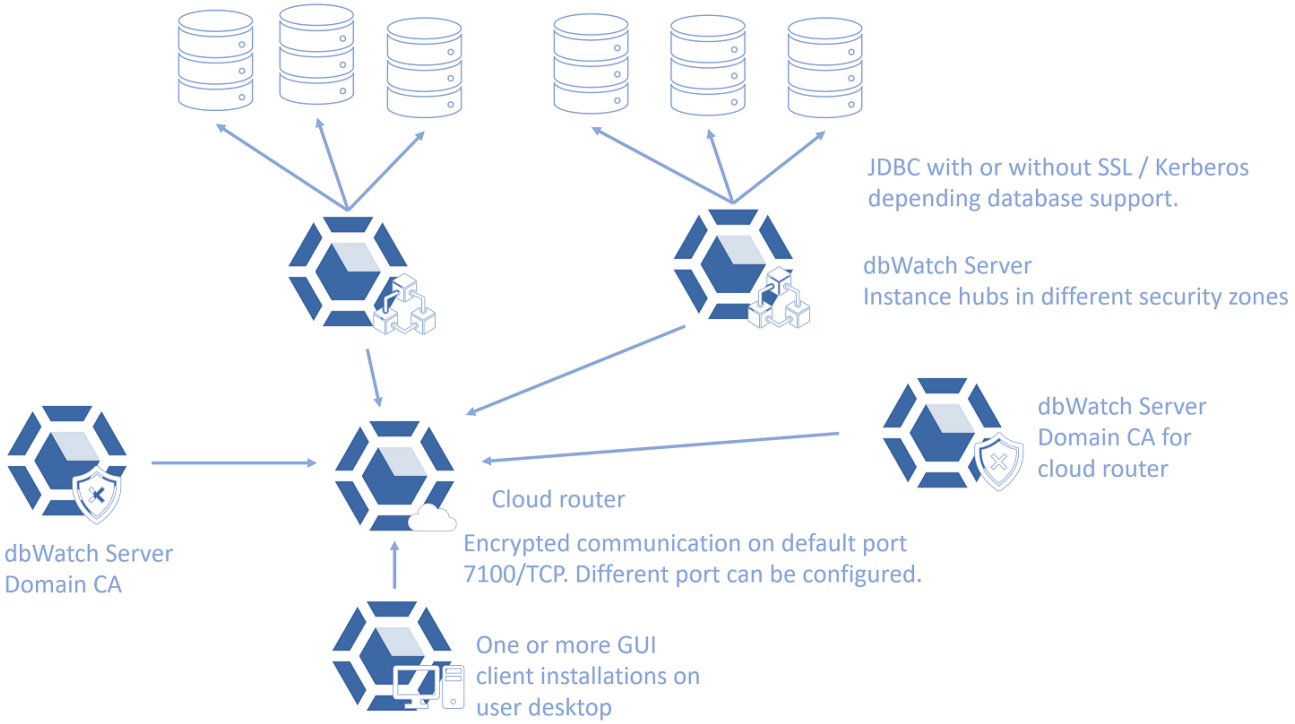
Use Cases

- **Managing Multiple Networks as One**: Businesses often operate servers across various network zones, including high-security internal zones, DMZ zones, and clouds. The Cloud Router enables the seamless management of these servers, making it as straightforward as if they were all located within a single zone. This capability simplifies the administration of complex network architectures, ensuring efficient and centralized control.
- **Managing Customer Networks**: For service providers managing networks on behalf of customers, maintaining strict separation between different customers' networks is crucial. The Cloud Router facilitates this by providing a secure way to manage each customer's network independently, ensuring no crossover or unintended connection between them.
- **Temporary Access for Consultants**: Hiring consultants often requires granting them access to specific network resources. The Cloud Router technology allows for the provision of limited, controlled access to consultants, enabling them to perform necessary tasks without full network access, thus ensuring security and privacy.
- **Instant Support from dbWatch Technicians**: For immediate technical support, the Cloud Router offers a swift solution. By connecting to the Cloud Router network and authorizing access, dbWatch technicians can provide assistance quickly. This feature ensures prompt and efficient support, minimizing downtime and enhancing problem resolution.

Architecture

In a scenario where no direct connection is allowed between the dbWatch domain CA node and any of the Instance Hubs, a dbWatch Cloud Router can provide a secure connection point. It could be in a case where a Managed Service Provider wants to monitor several customer locations in the same setup, without a VPN and without opening ports in the customer firewall due to the increased attack vector this creates. A dbWatch Cloud Router is a specialized hardened dbWatch Server, more info on this here

Example drawing:



See Also

Cloud router security.

<< Example architecture designs / Cloud router security >>

Cloud router security

Introduction

Robust Security Despite Exposure to Threats: Designed for internet accessibility, the cloud router operates under the assumption that it could be compromised. This perspective has shaped its security architecture to ensure that users remain protected, even if control over the router is lost. By employing layered security measures and encryption, the cloud router is built to safeguard communication and data integrity, maintaining a secure environment despite its inherent vulnerabilities.

TLS/SSL Security

TLS/SSL in Cloud Routers: Cloud routers use TLS (Transport Layer Security) and SSL (Secure Sockets Layer) for secure data transmission. These protocols encrypt and ensure the integrity of data exchanges between applications.

Certificate-Based Authentication and Access Control: Nodes authenticate with the cloud router using TLS, with certificates issued by the domain controller of each domain. These certificates grant access to domain-specific data and routing information. Without the appropriate certificate, a node is unable to access or route any information related to that domain, allowing the domain controller to control the security and integrity of its network.

Message Encryption

Diffie-Hellman Key Exchange for Ephemeral Key Generation: Nodes establish ephemeral keys using a Diffie-Hellman key exchange to secure their communications, creating temporary encryption keys without a pre-shared secret.

Role of Ephemeral Keys in Traffic Encryption and Security: These keys encrypt traffic between nodes. The cloud router, handling routing, accesses only necessary metadata like target address and TTL. The content, encrypted by these keys, remains confidential, ensuring security even if a router is compromised.

Message Filtering and Inspection

Service Information

Service Lists and Their Composition: The cloud router maintains lists of services provided by each domain, including encrypted and unencrypted parts. Access to the encrypted sections requires keys from the respective domain controller.

Access Control Based on TLS Authentication: Upon receiving a service information request, the cloud router checks the TLS connection and provides service lists corresponding to the certificates used.

Route Information

Access Control for Routing Information: The cloud router shares routing information based on the TLS certificates in the connection, allowing domain controllers to regulate access to their routing data.

Routing

Certificate-Based Routing Verification: When routing a message, the cloud router compares the incoming and outgoing sockets' certificates. Messages are routed only if these sockets have a matching domain certificate, ensuring that communication to and from the domain network is authorized and secure.

Authentication and Authorization

Certificate-Based Authentication: All authentication in the cloud router environment is based on the validation of certificates. The cloud router uses these certificates to authenticate nodes, ensuring that all connections and data transmissions are with entities that are recognized and authorized based on their certificate validity.

Authorization Based on Certificates: Authorization within the cloud router is directly tied to the certificates issued for specific domains. The behavior and access a node is authorized for in a given domain are determined by the certificates it possesses.

Domain Controllers and Certificate Signing: Domain controllers receive certificates signed by dbWatch, validating their authority over a given domain. This central validation by dbWatch is crucial, as it establishes the trust needed for the cloud router and other nodes in the network to honor the certificates issued by the domain controllers.

<< Cloud router / File Structure >>

File Structure

Files stored in the users home directory

The dbWatch Monitor stores files in the ".dbwatch/.monitor/" catalog in the users home directory. This directory contains the following directories:

- **config** Config files
- **crypto** Cryptographic keys
- **data** Currently not used
- **log** Log files
- **work** Workarea

Relevant files are:

- **config/uuid.txt** This contains the unique id for this monitor in the dbWatch network.

- **config/node/node.connections** Contains the addresses this monitor should connect to (i.e dbWatch Server addresses) and an optional port it should listen to.
- **config/node/domain_connections.json** Contains the dbWatch domains the Monitor should connect to and the authentication to use.
- **config/node/node.resources** The libraries the Monitor should load at startup.

- **config/work/monitor.xml** Stores various settings and preferences for the monitor.
- **config/work/layout_cache.json** Cache of the domain layout, for faster Monitor startup.
- **config/work/node_name_cache.txt** Cache of Server names. Lets the Monitor know the names of Servers it is unable to connect to.
- **config/work/profiles/custom/** Files defining custom profile settings for the Monitor (custom views etc.)

Monitor files in the installation directory

- **monitor.conf** Startup configuration for the Monitor.
- **monitor.vmoptions** Optional startup arguments for the JVM.

Server files in the installation directory

- **server.conf** Startup configuration for the Server.
- **server.vmoptions** Optional startup arguments for the JVM.

Comon files in the installation directory

- **jre/** The bundled Java Runtime
- **lib/** Some 3 party libraries
- **classpath** Contains the dbWatch base library (dependent)

Files stored by the dbWatch Server

The dbWatch Server places some files in a workarea (default “C:/ProgramData/dbWatchControlCenter/”)

- **config** Config files
- **crypto** Cryptographic keys and certificates
- **data** Misc data store
- **log** Log files
- **work** Workarea

Relevant files are:

- **config/uuid.txt** This contains the unique id for this Server in the dbWatch network.

- **config/node/node.connections** Contains the addresses this Server should connect to (i.e other dbWatch Server addresses) and the port it should listen to.
- **config/node/node.resources** The libraries the Server should load at startup.
- **config/node/trustStore/** Cryptographic certificates the Server trusts.

- **config/domain/[domain name]/** Contains configuration info for the domain. (Present only on the domain controller server)

- **config/server/group_defaults.xml** Contains the defined groups and their default values.
- **config/server/server_configuration.xml** The Server configuration
- **config/server/server_list_configuration.xml** Contains the definition for the lists (object sets)

- **config/data/** Contains static metadata on the entities defined in dbWatch

- **config/extensions** Configuration files for the extensions (currently e-mail extension)

- **data/data** Dynamic property store

Click Here for Start Up Configuration

<< Cloud router security / Network and communications >>

Network and communications

Network and communications

Default ports and protocol

dbWatch Control Center uses by default port 7100/tcp for client to node and node to node connections. This can be configured to a different port. On each node there is a node.connections file that both lists what ports the dbWatch Server node listens to, and what nodes it has a preconfigured connection to.

Example node.connections file on a dbWatch Server node:

```
{
        "listenTo":[
                 "dbw://localhost:7100"
        ]
}
```

Example node.connections file on a dbWatch client node:

```
{
        "versionCutoff":"",
        "connectTo":[
                "dbw://localhost:7100"
        ]
}
```

Any node can have a combination of listenTo and connectTo statements, and also a comma seperated list of hosts and ports. This can be used to form static connection from a server to another. In one scenario from an instance hub node to a cloud router node outside a firewall. This provides access without an inbound firewall opening, and only allows for the encrypted dbWatch traffic. As audit can be enabled in the system, this setup can provide better security than open port or direct jdbc connections.

Mesh communications

dbWatch Server nodes form a mesh network when connected and authenticated by a dbWatch domain CA node and accepted into the domain. The mesh network uses end to end encryption and provides routing to allow dbWatch traffic multiple paths to reach the target node.

Domain security and the CA system

The domain security model used in dbWatch is a method of creating a security bubble that encompasses the dbWatch Client and the dbWatch Server nodes that constitutes the dbWatch environment in a customer location. The dbWatch Server with the domain ca node responsibilities will create new

certificates for the domain members, on average every 10 minutes. The certificates are short lived, and is only used internally in dbWatch to provide packet signing and encryption.

Database operations philosophy

The reasons for database operations

The value of database operations comes from the value of the systems the databases support. . In a typical operation, the value of the managed systems is much greater than the resources used on the operations side.
Running a successful It organization is a question of managing risks, resources, and manpower in order to maximize the value without taking an unacceptable risk.

We recognize that imitate dangers must be handled when they appear, since they represent immediate risk. There are also a set of maintenance routines that must be performed on a regular basis in order to avoid risk in a short to medium time frame.
When these risks are controlled it is possible to move the focus of the organization toward the more strategic parts of operations.

We have defined a system of maturity levels to describe the state of an IT organization’s processes to give advice on which areas to focus on in order to optimize its value.

Maturity levels

1. Control of immediate dangers.

The organization is working to achieve control over the immediate dangers to the operation.

Mastering this level involves establishing a process of continually looking for potential dangers and handling them.

Achieving control over immediate dangers frees up resources and provides a measure of stability by drastically reducing the chance of disruptive events. This will allow
the organization to focus on medium-term issues.

2. Automaton of maintenance

The organization is working to achieve control over the day-to-day maintenance of the databases.

The process of maintaining a database involves running a set of maintenance routines on a regular schedule and verifying the outcome
of these routines. Automation is essential since it allows for including all databases in the management regime.

Achieving control over the day-to-day maintenance will make sure that your databases are running smoothly over time. It will

3. Optimization of performance

The organization is working to achieve a stable optimized performance for the system as a whole.

The organization establishes processes for analyzing, prioritizing, and tuning the parts of the system with the biggest potential gain. Note that
the perfect optimized state will never be achieved.

Achieving control over the system's performance will greatly stabilize the performance characteristics of the system as a whole. This is
essential to be able to manage resource utilization in a reasonable way.

4. Resource management

The organization is working to achieve optimal resource utilization.

Having control over the performance of the systems provides a good and stable foundation for managing the resource utilization of these systems.

Difference between maturity levels

The first level of database operations maturity is monitoring. This involves having complete control of all your databases and knowing if there is a problem in your systems. To achieve this you need all your database instances enrolled into monitoring, where you know that all the database instances have the correct monitoring enabled. When all is green in this level of maturity, you can focus on the second level of maturity.

The second level of database operations maturity is maintenance. This involves automating the daily routines, where you detect and fix any deviations to how the routines should work. When all is green in this level of maturity, you can focus on the third level of maturity.

The third level of database operations maturity is performance.

The fourth level of database operations maturity is capacity.

The fifth level of database operations maturity is consistency. This involves monitoring the other levels of maturity to make sure it is within a set standard.

[<< Network and communications] / [Monitor dashboards and overview screens >>]

Monitor dashboards and overview screens

Being effective in database operations are one of the key focus areas for the dbWatch database operations team.

This chapter will focus on the dashboards or overview screens used by our internal database operations team, what information they show, how they are used to detect and find issues early.
The chapter differs from the chapter of web dashboards, where the web dashboards helps to identify problems, the monitor dashboards are used by the individual DBA in the client software, dbWatch monitor, and present a more detailed view, enabling drill down and helps struturing the day to day work performed on database operations.

<< Database operations philosophy / Installing and using premium packages >>

Installing and using premium packages

Additional packages

dbWatch Control Center also provides additional premium packages to cover specific areas.
They may be specific only to one or a few of the platforms supported by dbWatch Control Center.
More details are found on the pages specific to the package.

The current modules provided are:

SQL Performance package

Security and Compliance package

<< Monitor dashboards and overview screens / SQL Performance package >>

SQL Performance package

SQL Performance package

SQL Performance module is an optional extra cost module in dbWatch Control Center that allows for historical monitoring and analysis of SQL Statements in the database instance.

Currently supported platforms:

- MS SQL Server 2014 and newer
- Oracle 11g and newer
- PostgreSQL 9.4 and newer
- MySQL 5.7 and newer

Installation steps are identical for all platforms, but the data displayed and some functionality can be platform dependent.

[<< Installing and using premium packages] / [Installation of SQL Performance package >>]

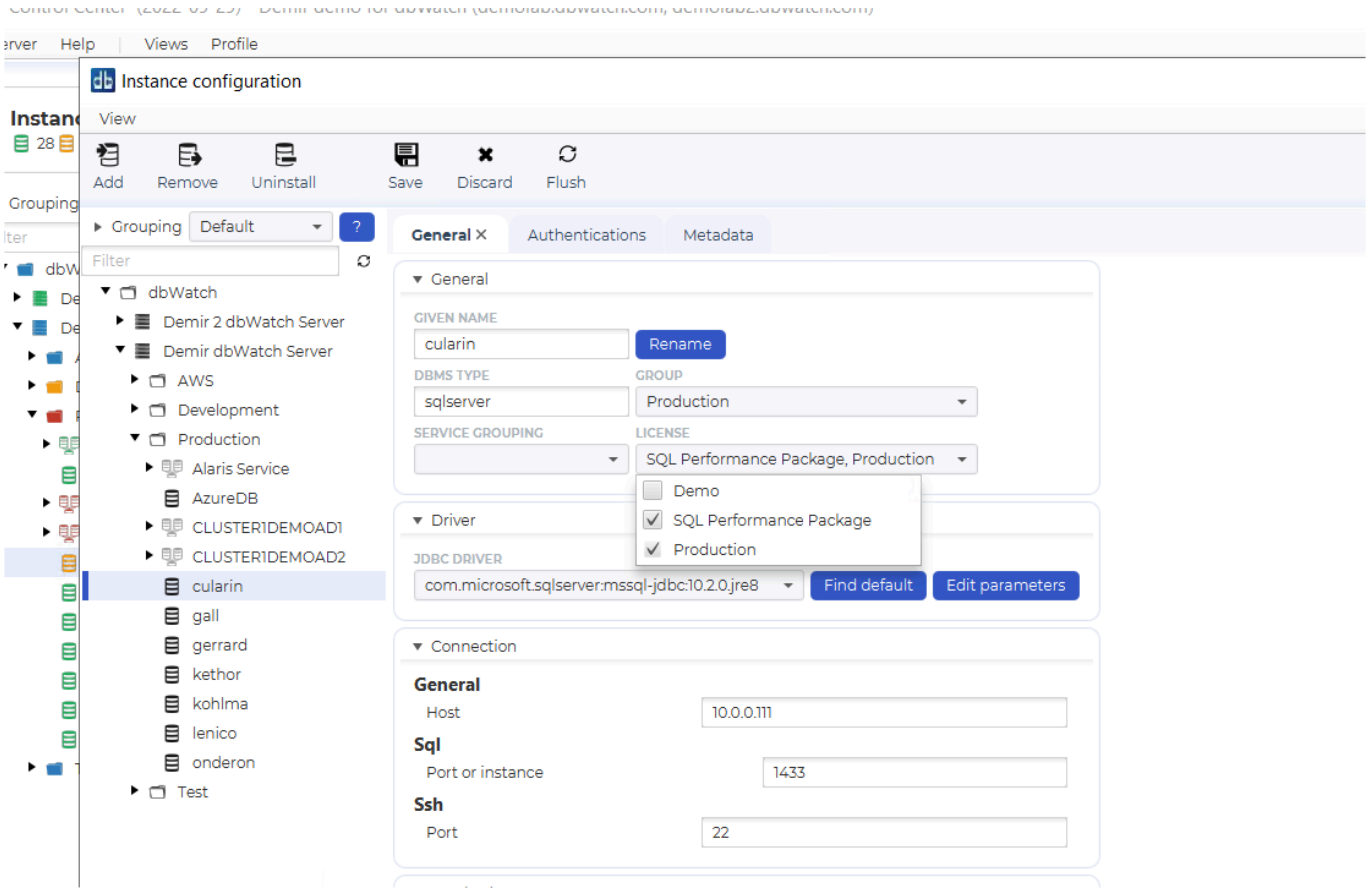
Installation of SQL Performance package

Requirements for installation

- dbWatch Control Center release 2022-09-29 / 501 or newer
- License that includes SQL Performance module. It can be as the separate “SQL Performance Package” license or as part of a larger licence package. It is included in the DEMO license type.
- Database platform and version that is supported
- Space on the database instance, around 5 GB depending on the amount of SQL queries, configured thresholds and desired history requirements.

Adding the licence

If the database instance is using the DEMO licence type, you can skip this step, as it is already enabled.



Right click on an instance, choose Configure instance. If you pull down the Licence list, you must make sure a licence that includes the SQL Performance package is chosen. In this case, both Demo and SQL Performance Package is valid choices. After you have made your choices, click “Save” to save changes.

Additional steps for PostgreSQL

The SQL Performance package on PostgreSQL needs some additional steps.

First, make sure you have the following settings in the postgresql.conf file.

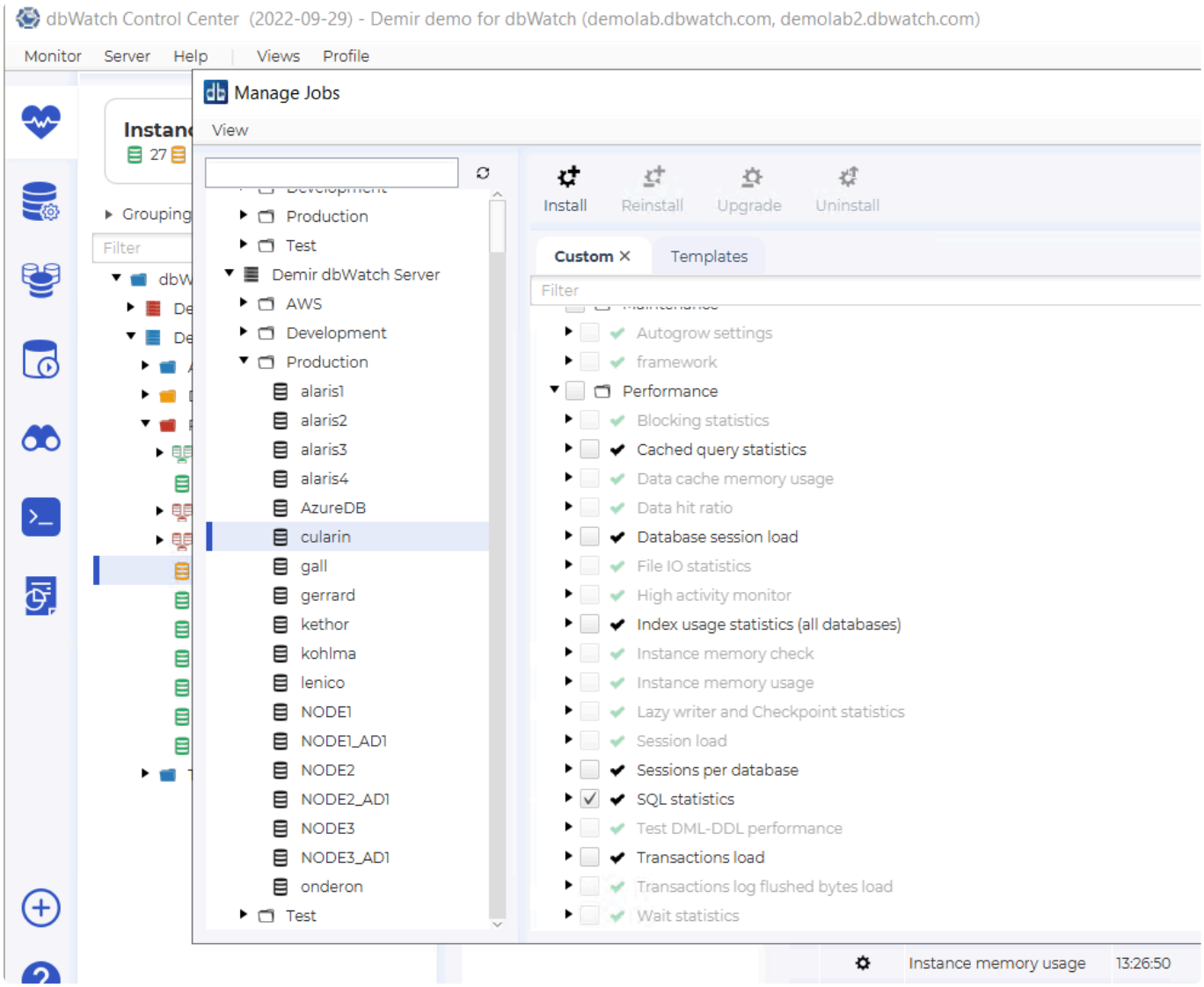
track_activity_query_size = 2048
shared_preload_libraries = 'pg_stat_statements'
pg_stat_statements.track = all
pg_stat_statements.max = 10000

Also in the database instance, add the extension:
CREATE EXTENSION pg_stat_statements with schema public

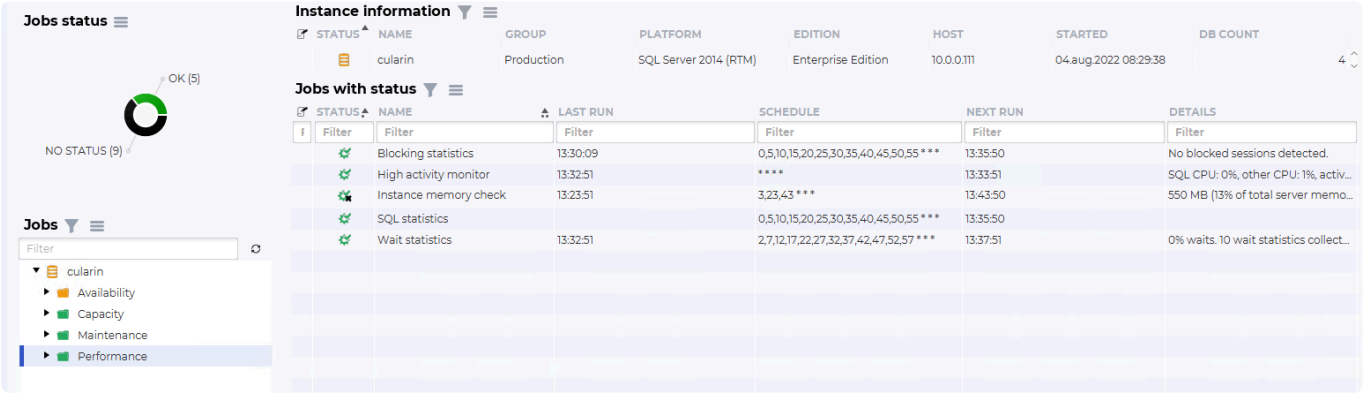
Installing the SQL Performance package repository job

Once the correct licence is enabled, you can install the job that collects SQL performance data for the repository.

Right click on the instance name, and chose "Configure jobs"



Scroll down to the Performance group, and select the job "SQL statistics". If it is not available, you have not enabled the correct licence, or the platform/version is not supported. Once selected, click "Install" to install it on that instance. You should get an OK popup. Close the popup and the "Manage jobs" view.



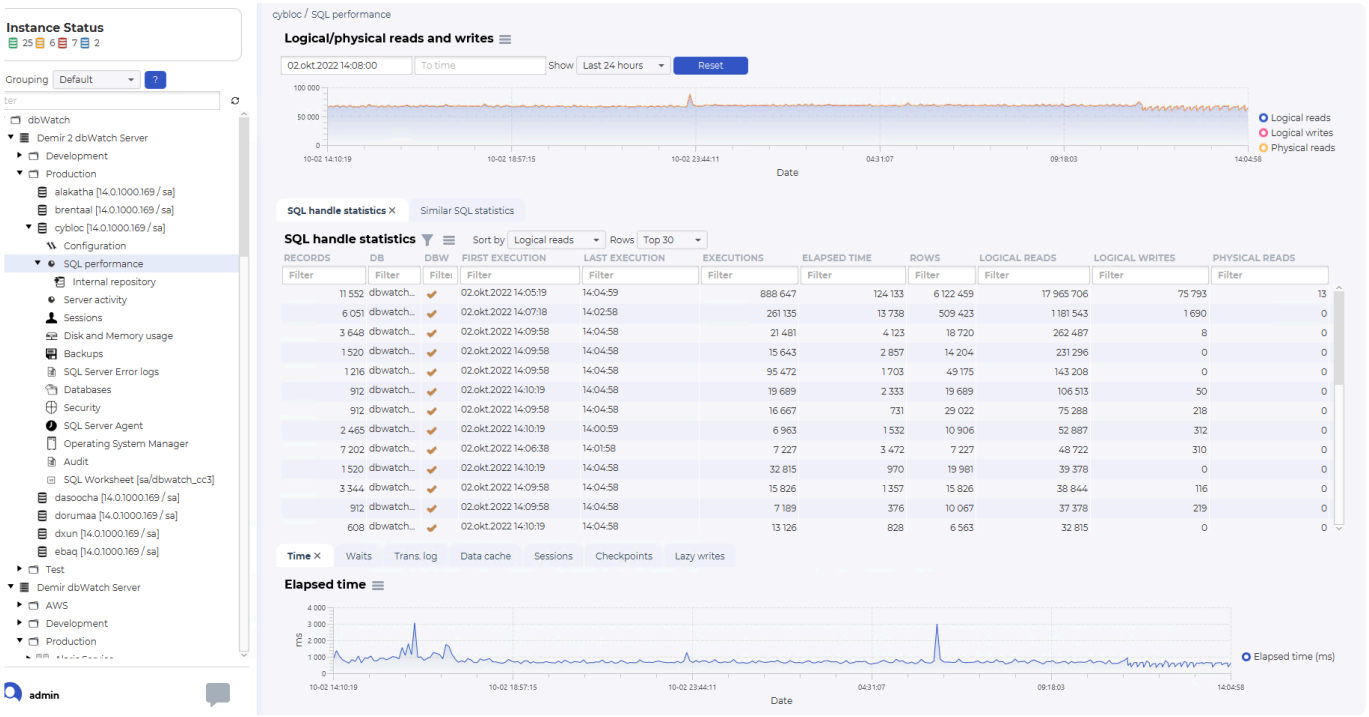
It should now be available in the Performance group on that instance. It will run automatically every 5 minutes by default, and can be triggered manually if you right click and choose “Run”. Allow it to gather data for some time, such as one day.

<< SQL Performance package / Using SQL Performance package on SQL Server >>

Using SQL Performance package on SQL Server

Navigating to the SQL Performance package dashboard

Once the SQL statistics job is installed and have collected data, you can navigate to the dashboard. It is located in the management interface for that database instance. The pictures and descriptions are from SQL Server.

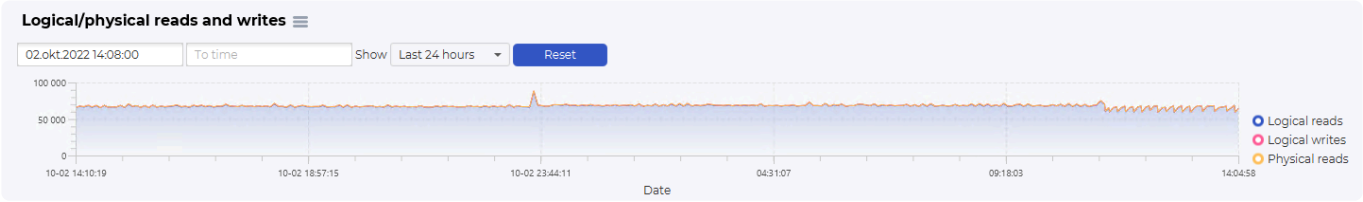


There should be a “SQL performance” node in the three structure. Click on it to open the SQL Performance dashboard.

Using the SQL Performance dashboard

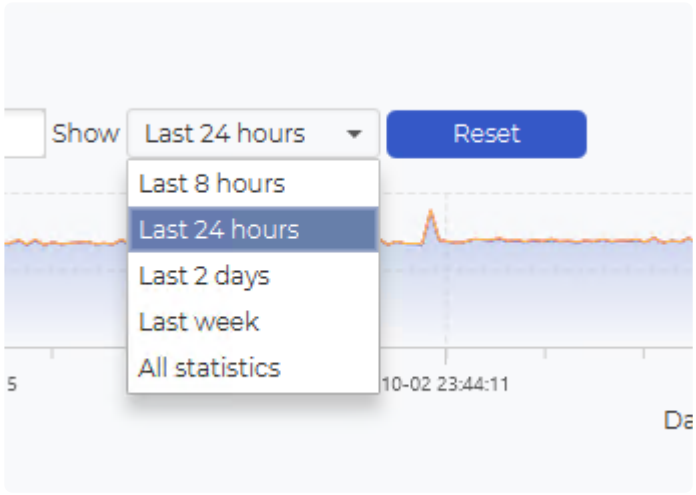
Navigating time

The structure of the SQL Performance dashboard is a top graph showing historical data, such as Logical/physical reads and writes.

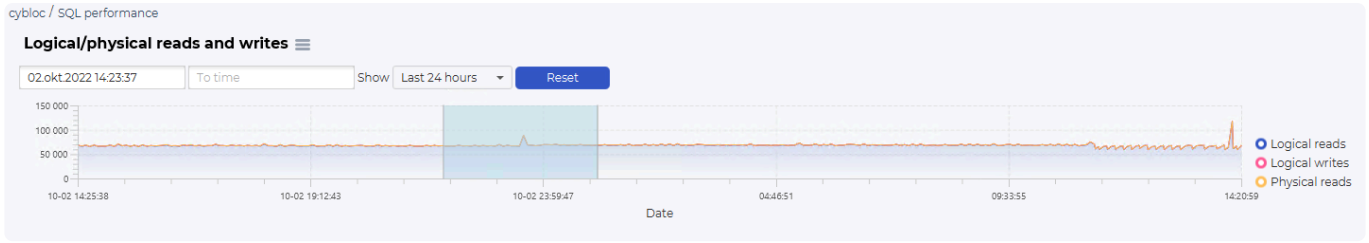


The logical/physical reads and writes graph is used to select the timeframe you want to look at.

You can select predefined time selections, such as Last 8 houres, Last 24 houres, Last 2 days, Last week and all statistics:



You can also use the mouse to select a timeframe from the graph. Click on your start time, hold the mouse button, drag to the desired end time, and release the mouse button.



This will change the selection on the SQL Handle statistics table and the additional information graphs.

SQL handle statistics (SQL Server) or SQL statistics (Oracle)

The SQL handle statistics/SQL statistics will list SQL handles or SQL statements that has been active in the selected time period.

The fields

RECORDS	DB	DBW	FIRST EXECUTION	LAST EXECUTION	EXECUTIONS	ELAPSED TIME	ROWS	LOGICAL READS	LOGICAL WRITES	PHYSICAL READS
190	dbwatch...	✓	02.okt.2022 22:50:39	02.okt.2022 23:15:39	12 898	1 896	100 018	294 027	541	0
105	dbwatch...	✓	02.okt.2022 22:52:18	02.okt.2022 23:12:18	4 520	252	8 815	20 469	15	0
60	dbwatch...	✓	02.okt.2022 22:54:58	02.okt.2022 23:14:58	374	65	325	4 550	0	0
25	dbwatch...	✓	02.okt.2022 22:54:58	02.okt.2022 23:14:58	275	52	250	4 100	0	0
20	dbwatch...	✓	02.okt.2022 22:54:58	02.okt.2022 23:14:58	1 600	22	825	2 400	0	0
15	dbwatch...	✓	02.okt.2022 22:55:18	02.okt.2022 23:15:18	336	41	336	1 858	0	0
15	dbwatch...	✓	02.okt.2022 22:54:58	02.okt.2022 23:14:58	275	15	475	1 250	0	0
125	dbwatch...	✓	02.okt.2022 22:51:38	02.okt.2022 23:11:38	125	54	125	849	4	0
34	dbwatch...	✓	02.okt.2022 23:00:19	02.okt.2022 23:10:19	96	20	140	696	0	0
55	dbwatch...	✓	02.okt.2022 22:54:58	02.okt.2022 23:14:58	275	25	275	675	0	0
25	dbwatch...	✓	02.okt.2022 22:55:18	02.okt.2022 23:15:39	560	17	340	672	0	0
15	dbwatch...	✓	02.okt.2022 22:54:58	02.okt.2022 23:14:58	125	2	175	650	0	0
27	dbwatch...	✓	02.okt.2022 22:52:18	02.okt.2022 23:12:18	27	30	90	613	2	0

(Depending on the release of dbWatch Control Center, some fields could be missing)

Records:
The number of times this SQL statements has been detected by dbWatch

DB/Schema:
The database where this procedure is located (SQL Server) or schema executing the SQL or procedure (Oracle)

DBW:
Is this a SQL or procedure assosiated with dbWatch

First execution:
The first time **in the selected time frame** where this SQL or procedure was executed

Last execution:
The last time **in the selected time frame** where this SQL or procedure was executed

Executions:
The number of times **in the selected time frame** where this SQL or procedure was executed

Elapsed time:
The total elapsed time (in milliseconds) **in the selected time frame** where this SQL or procedure was executed

Rows:
The total number of rows **in the selected time frame** where this SQL or procedure was executed

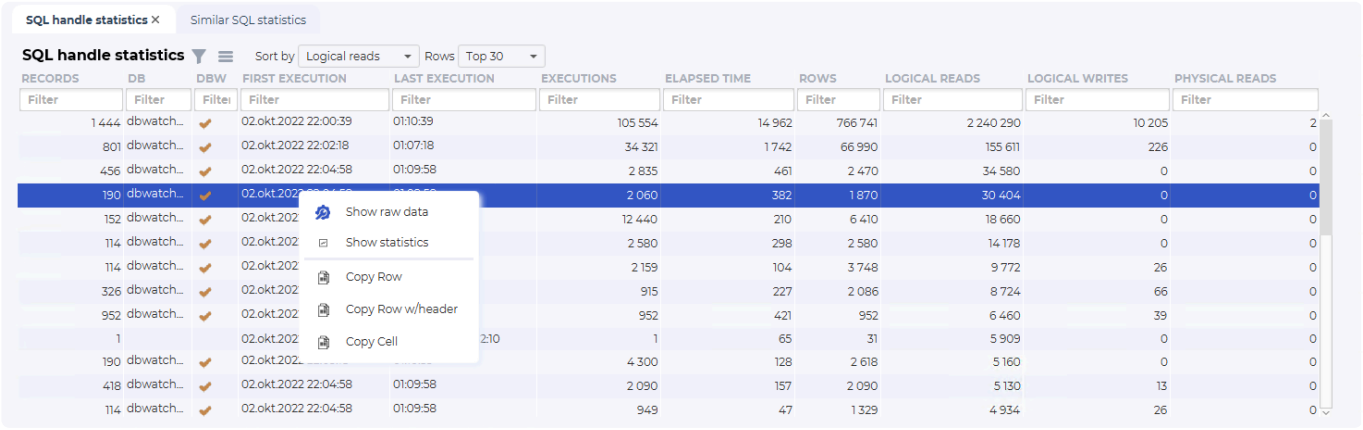
Logical reads/Buffer gets:
The total number of logical reads/buffer gets **in the selected time frame** where this SQL or procedure was executed

Logical writes/Direct writes:
The total number of logical writes/direct writes **in the selected time frame** where this SQL or procedure was executed

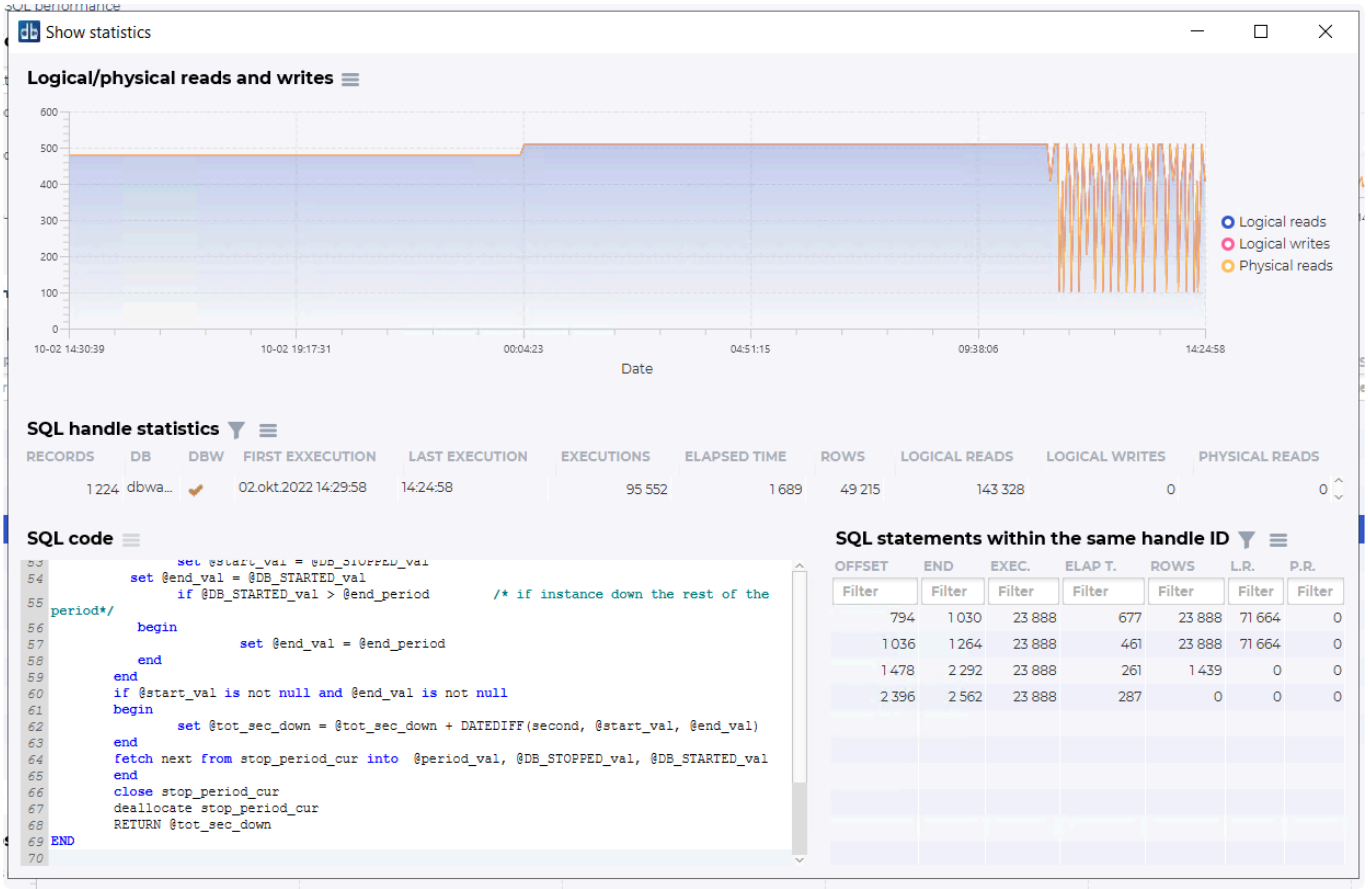
Physical reads/Disk reads:
The total number of physical reads/disk reads **in the selected time frame** where this SQL or procedure was executed

M.I.
Indicates that this SQL handle is missing an index

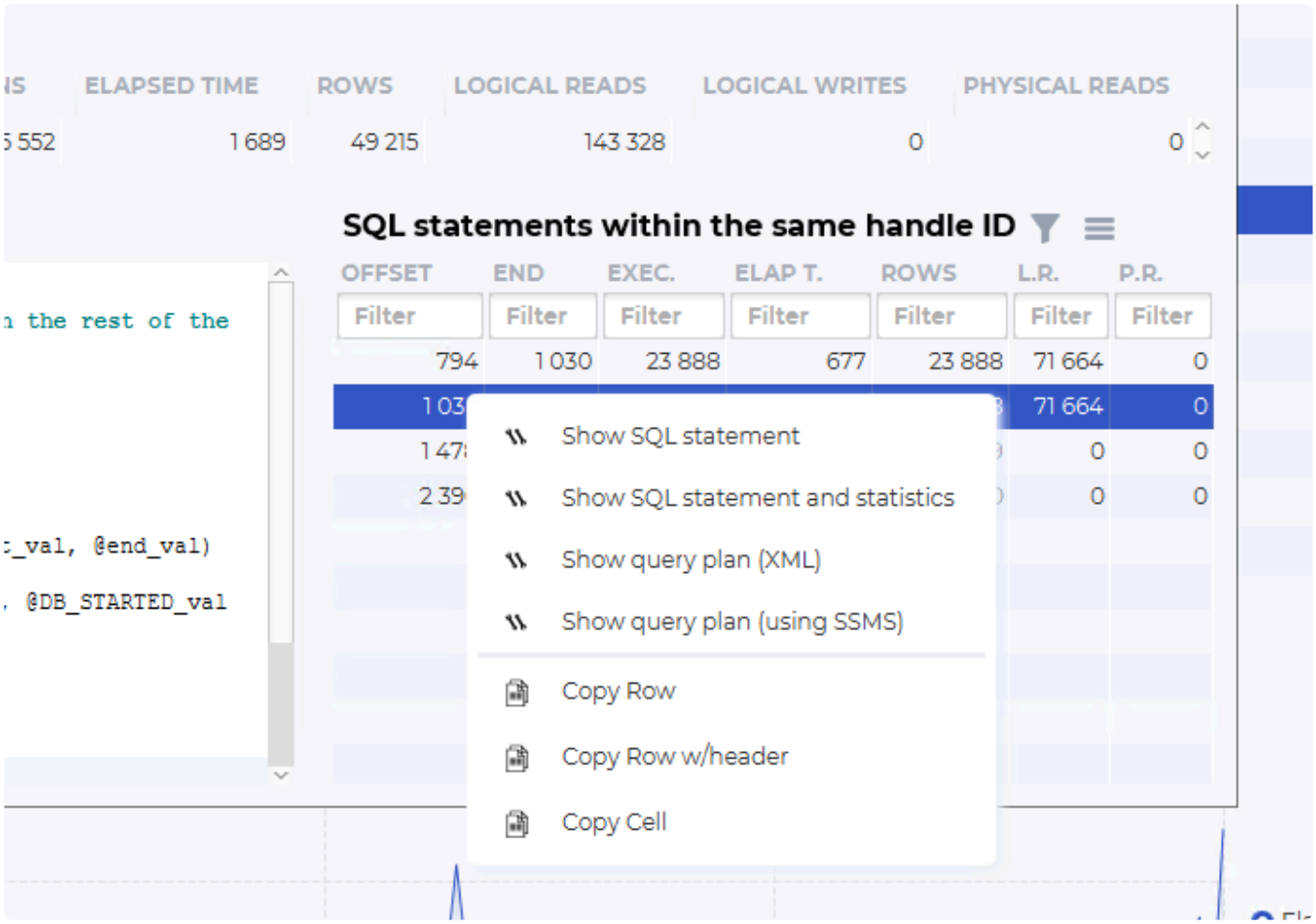
Navigating SQL handle statistics / SQL statistics



You can investigate the SQL handles further by right clicking on one of the lines. Show statistics will open a new dashboard focusing on one SQL handle or SQL statement.



This dashboard will show logical/physical reads and writes statistics for this statement, the statistics for the SQL handle, the SQL code that has been run, and also individual SQL statements within the SQL handle if it is a procedure with multiple SQL statements.



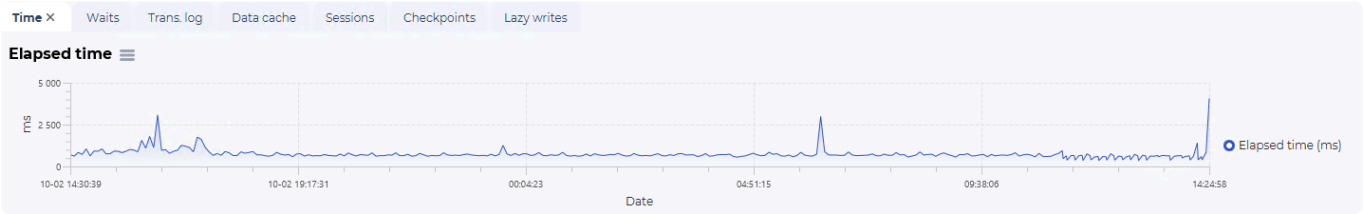
You can also open each SQL statement within the SQL handle to get statistics for that statement. “Show query plan(using SSMS)” requires setting up the [integration with SSMS.]

Additional graphs

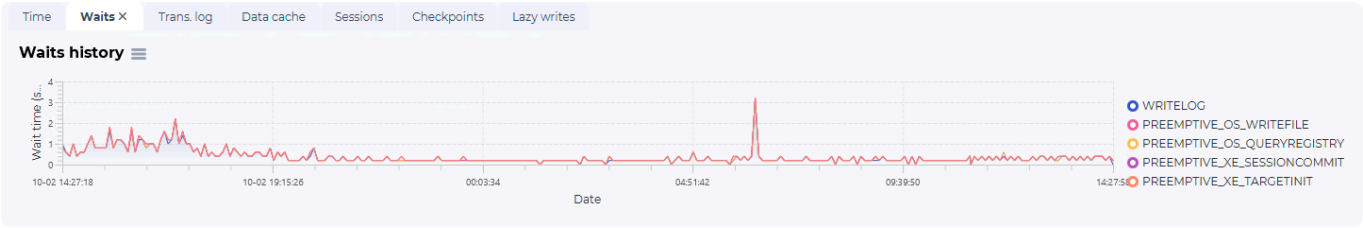
In addition to the logical/physical reads and writes and the SQL handle statistics overview, there are additional graphs to visualize other performance statistics from the same selected timeframe.

Examples:

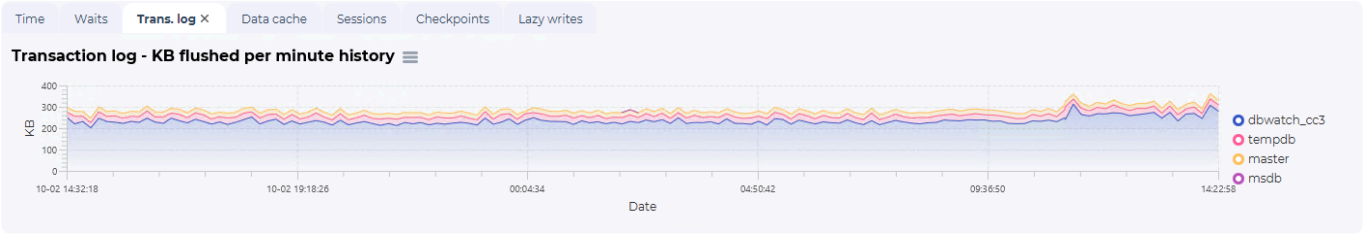
Elapsed time:



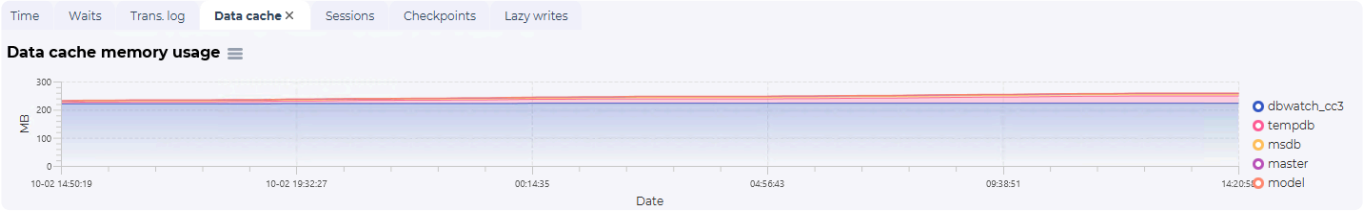
Waits history:



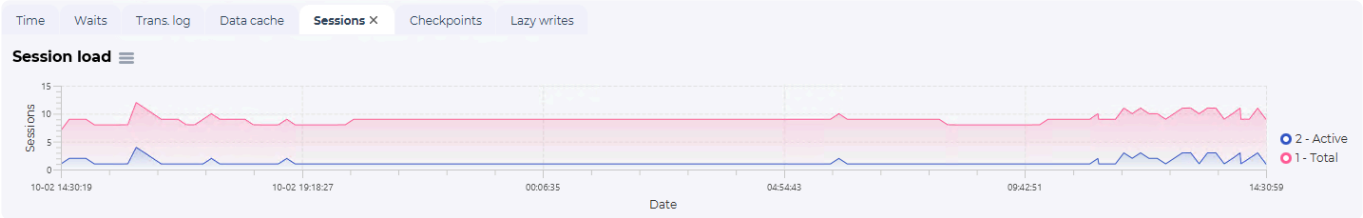
Transaction log – KB flushed per minute:



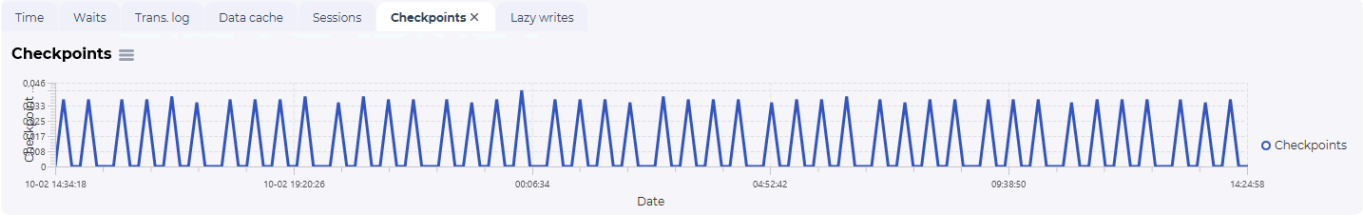
Data cache memory usage:



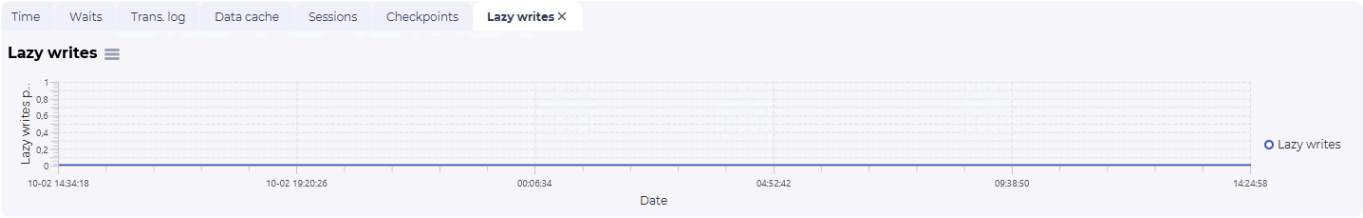
Session load:



Checkpoints:



Lazy writes:



<< Installation of SQL Performance package / Integration with SSMS >>

Integration with SSMS

When analyzing a SQL statement, the choice “Show query plan (using SSMS)” appears on SQL Server. This is for opening a query plan in SQL Server Management Studio.
Clicking on this will not open anything, if it is not setup correctly.

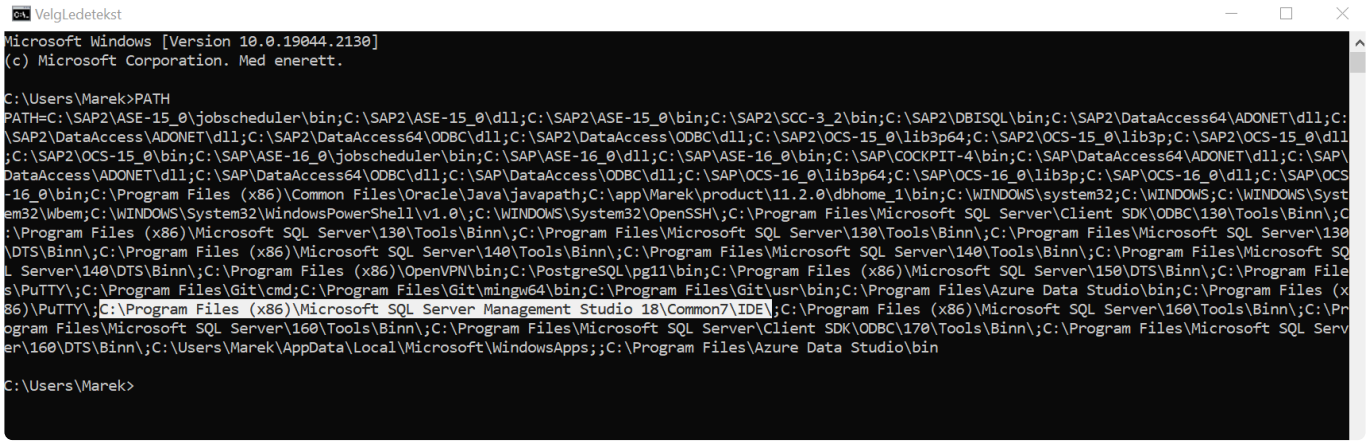
There is a monitor.xml file in the .dbwatch/.monitor/work/ directory for each user. Example: C:\Users\[USERNAME]\.dbwatch\.monitor\work\

There you must verify that there is a “queryplan_viewer_sqlserver” entry and that it is correct. If it is without a path, the path variable for your user cmd.

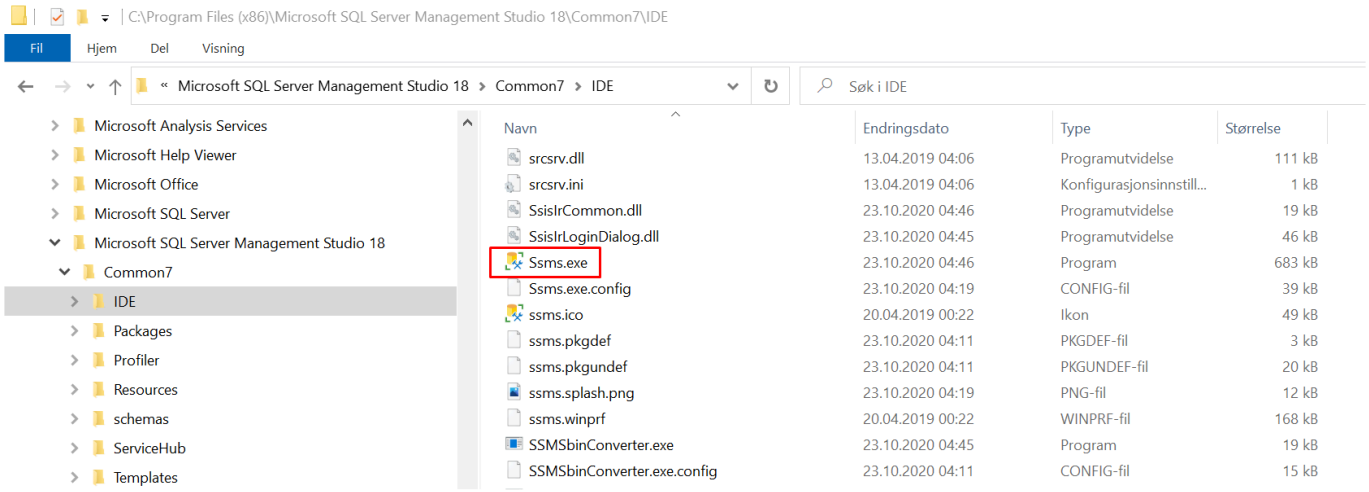
Example queryplan_viewer_sqlserver entry:

```
        <value>0.12431289640591967</value>
    </param>
    <param>
        <key>queryplan_viewer_sqlserver</key>
        <class>java.lang.String</class>
        <value>ssms.exe</value>
    </param>
    <param>
        <key>REMOTE_LAUNCHER:Ping_displayres</key>
        <class>java.lang.String</class>
        <value>YES</value>
```

Example path:



Example location:

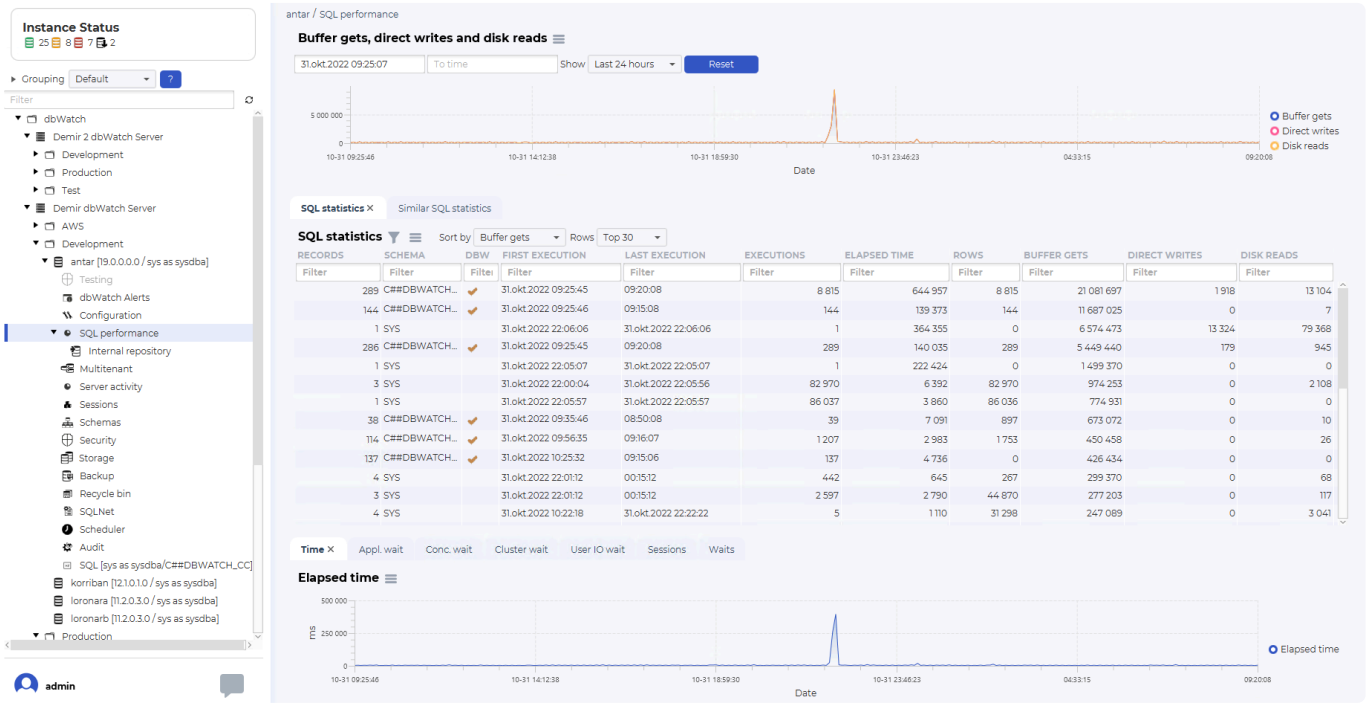


[<< Using SQL Performance package on SQL Server] / [Using SQL Performance package on Oracle >>]

Using SQL Performance package on Oracle

Navigating to the SQL Performance package dashboard

Once the SQL statistics job is installed and have collected data, you can navigate to the dashboard. It is located in the management interface for that database instance. The pictures and descriptions are from Oracle.

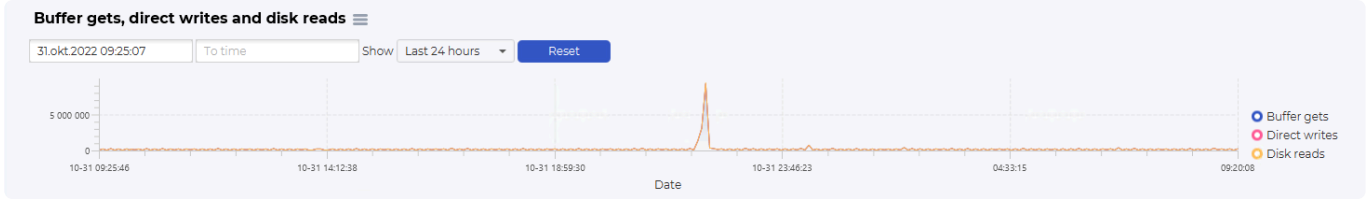


There should be a “SQL performance” node in the three structure. Click on it to open the SQL Performance dashboard.

Using the SQL Performance dashboard

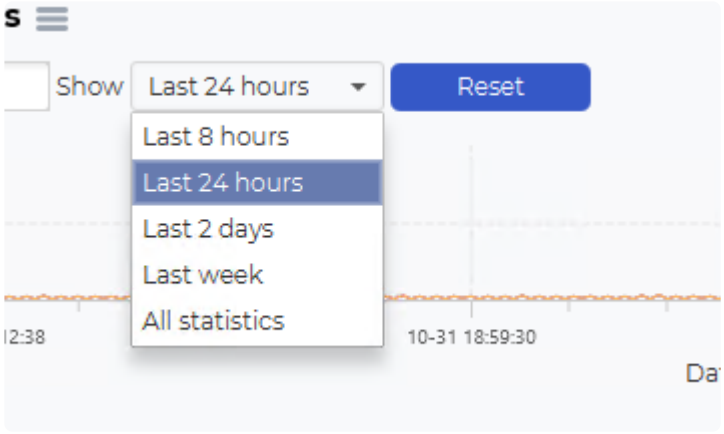
Navigating time

The structure of the SQL Performance dashboard is a top graph showing historical data, such as buffer gets, direct writes, and disk reads.

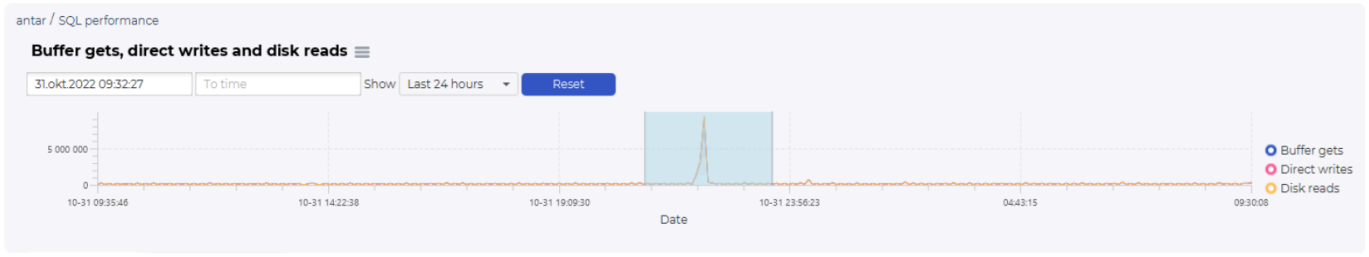


The buffer gets, direct writes, and disk reads graph is used to select the timeframe you want to look at.

You can select predefined time selections, such as Last 8 hours, Last 24 hours, Last 2 days, Last week, and all statistics:



You can also use the mouse to select a timeframe from the graph. Click on your start time, hold the mouse button, drag it to the desired end time, and release the mouse button.

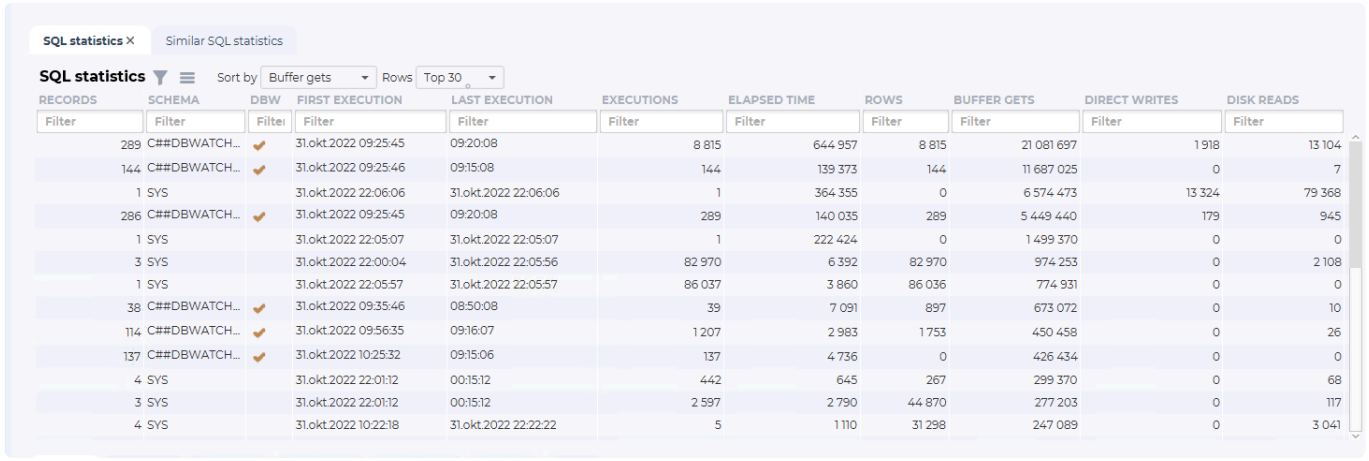


This will change the selection of the SQL statistics table and the additional information graphs.

SQL statistics

The SQL statistics will list SQL statements that have been active in the selected time period.

The fields



RECORDS	SCHEMA	DBW	FIRST EXECUTION	LAST EXECUTION	EXECUTIONS	ELAPSED TIME	ROWS	BUFFER GETS	DIRECT WRITES	DISK READS
289	C##DBWATCH...	✓	31.okt.2022 09:25:45	09:20:08	8 815	644 957	8 815	21 081 697	1 918	13 104
144	C##DBWATCH...	✓	31.okt.2022 09:25:46	09:15:08	144	139 373	144	11 687 025	0	7
1	SYS		31.okt.2022 22:06:06	31.okt.2022 22:06:06	1	364 355	0	6 574 473	13 324	79 368
286	C##DBWATCH...	✓	31.okt.2022 09:25:45	09:20:08	289	140 035	289	5 449 440	179	945
1	SYS		31.okt.2022 22:05:07	31.okt.2022 22:05:07	1	222 424	0	1 499 370	0	0
3	SYS		31.okt.2022 22:00:04	31.okt.2022 22:05:56	82 970	6 392	82 970	974 253	0	2 108
1	SYS		31.okt.2022 22:05:57	31.okt.2022 22:05:57	86 037	3 860	86 036	774 931	0	0
38	C##DBWATCH...	✓	31.okt.2022 09:35:46	08:50:08	39	7 091	897	673 072	0	10
114	C##DBWATCH...	✓	31.okt.2022 09:56:35	09:16:07	1 207	2 983	1 753	450 458	0	26
137	C##DBWATCH...	✓	31.okt.2022 10:25:32	09:15:06	137	4 736	0	426 434	0	0
4	SYS		31.okt.2022 22:01:12	00:15:12	442	645	267	299 370	0	68
3	SYS		31.okt.2022 22:01:12	00:15:12	2 597	2 790	44 870	277 203	0	117
4	SYS		31.okt.2022 10:22:18	31.okt.2022 22:22:22	5	1 110	31 298	247 089	0	3 041

Records:
The number of times this SQL statements has been detected by dbWatch

Schema:
The schema executing the SQL or procedure

DBW:
Is this a SQL or procedure associated with dbWatch

First execution:
The first time **in the selected time frame** where this SQL or procedure was executed

Last execution:
The last time **in the selected time frame** where this SQL or procedure was executed

Executions:
The number of times **in the selected time frame** where this SQL or procedure was executed

Elapsed time:
The total elapsed time (in milliseconds) **in the selected time frame** where this SQL or procedure was executed

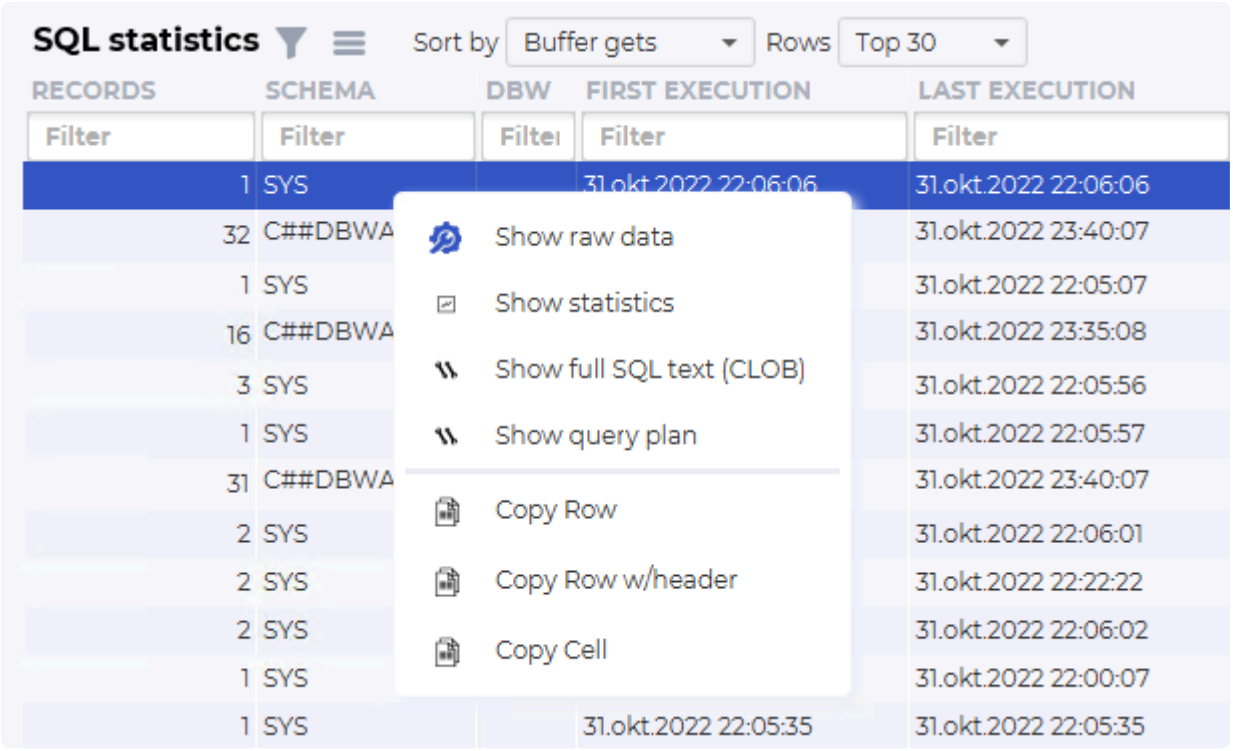
Rows:
The total number of rows **in the selected time frame** where this SQL or procedure was executed

Buffer gets:
The total number of buffer gets **in the selected time frame** where this SQL or procedure was executed

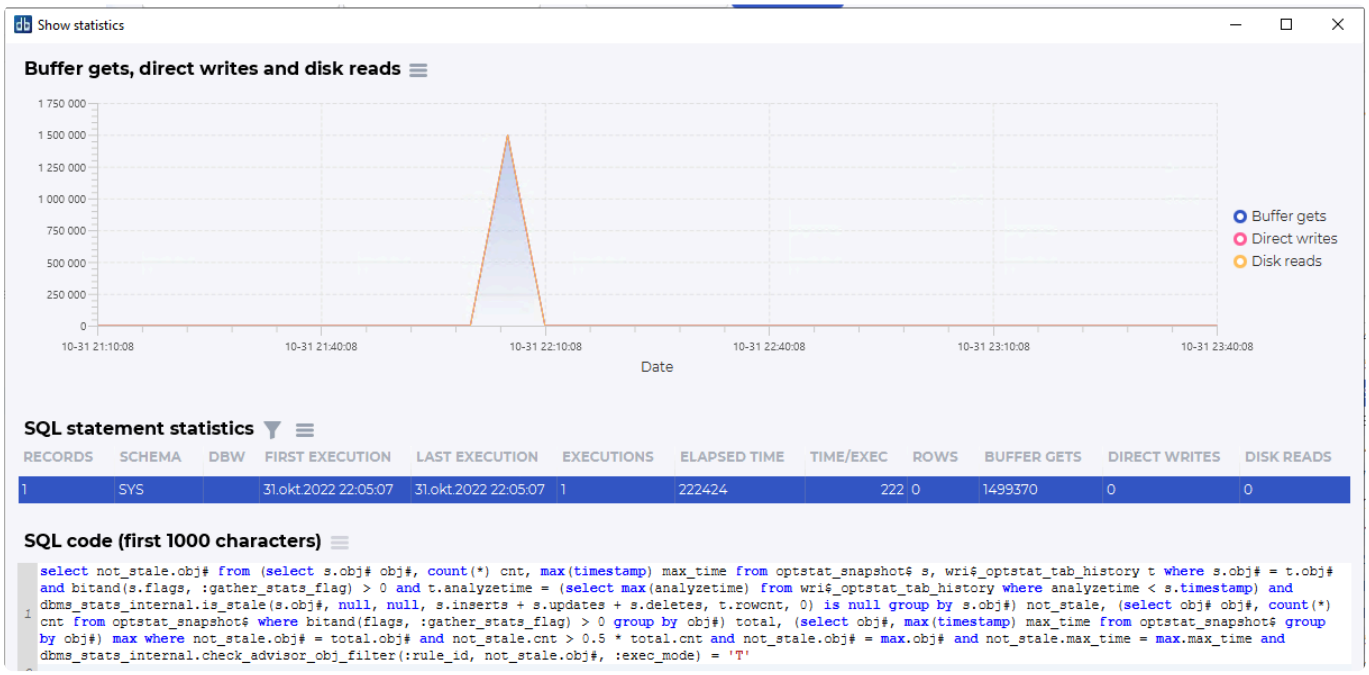
Direct writes:
The total number of direct writes **in the selected time frame** where this SQL or procedure was executed

Disk reads:
The total number of disk reads **in the selected time frame** where this SQL or procedure was executed

Navigating SQL statistics

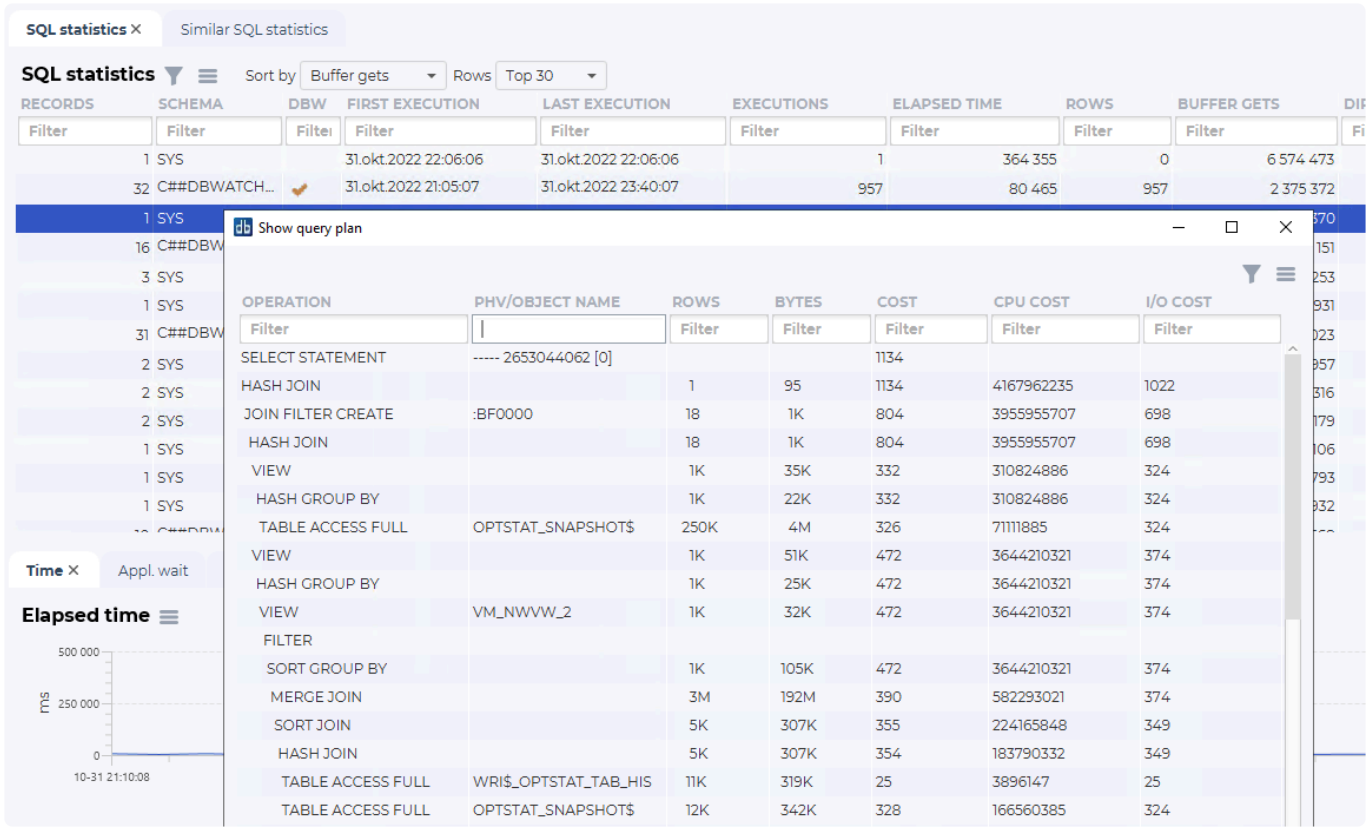


You can investigate the SQL further by right-clicking on one of the lines. Show statistics will open a new dashboard focusing on one SQL statement.



This dashboard will show buffer gets, direct writes and disk reads statistics for this statement, the statistics for the SQL statement, and the SQL code that has been run.

You can also open show query plan to open the query plan for the statement.

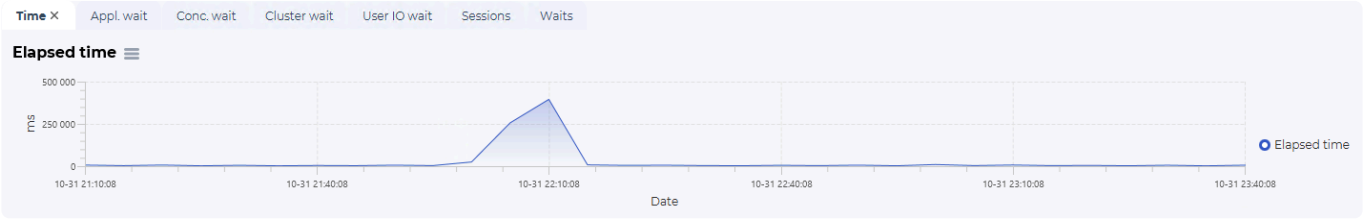


Additional graphs

In addition to the logical/physical reads and writes and the SQL statistics overview, there are additional graphs to visualize other performance statistics from the same selected timeframe.

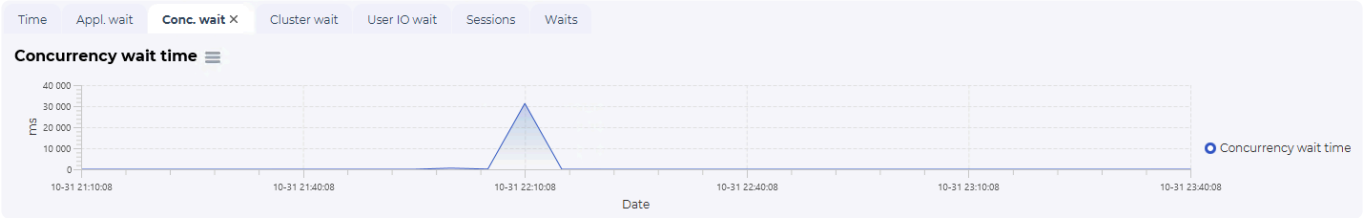
Examples:

Elapsed time:

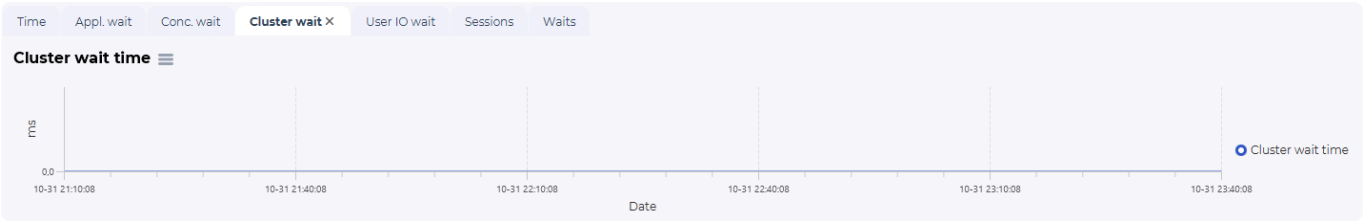


Application wait history:

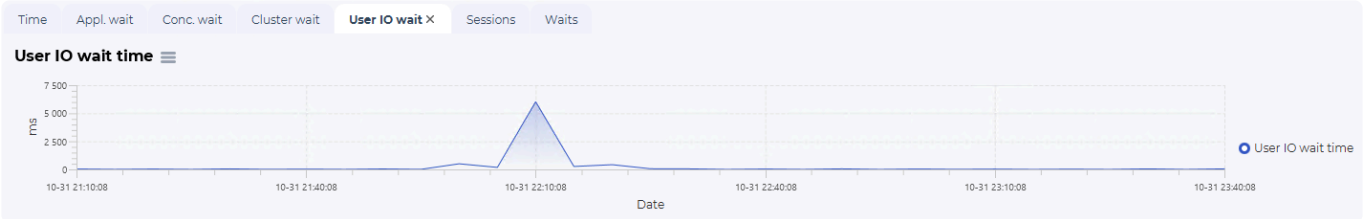
Concurrency wait history:



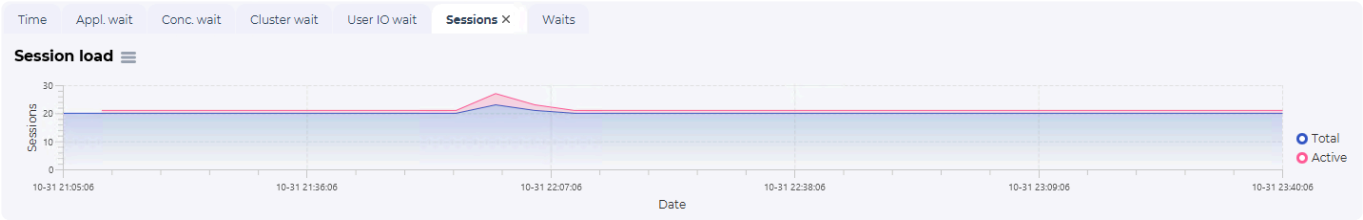
Cluster wait history:



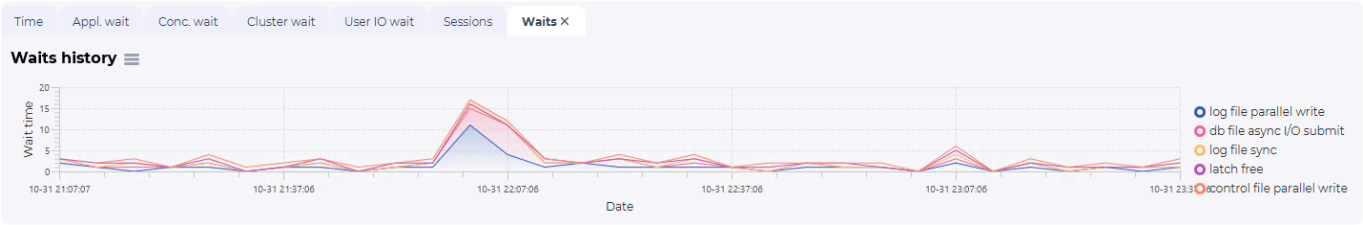
User IO wait history:



Session load:



Other waits history:



[<< Integration with SSMS](#) / [Using SQL Performance package on PostgreSQL >>](#)

Using SQL Performance package on PostgreSQL

Navigating to the SQL Performance package dashboard

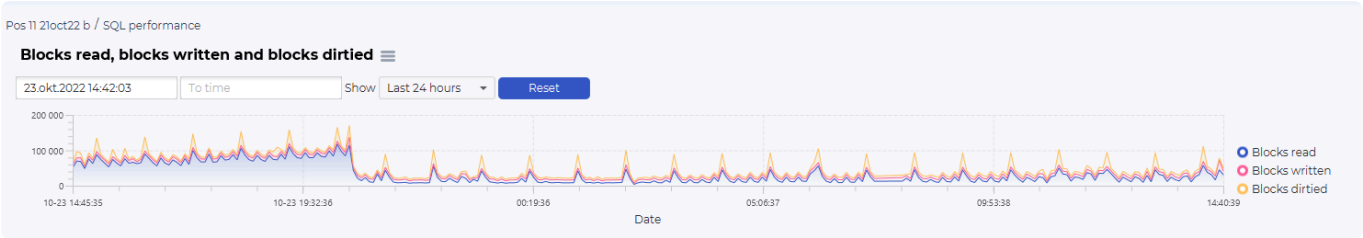
Once the SQL statistics job is installed and have collected data, you can navigate to the dashboard. It is located in the management interface for that database instance. The pictures and descriptions are from PostgreSQL.

There should be a “SQL performance” node in the three structure. Click on it to open the SQL Performance dashboard.

Using the SQL Performance dashboard

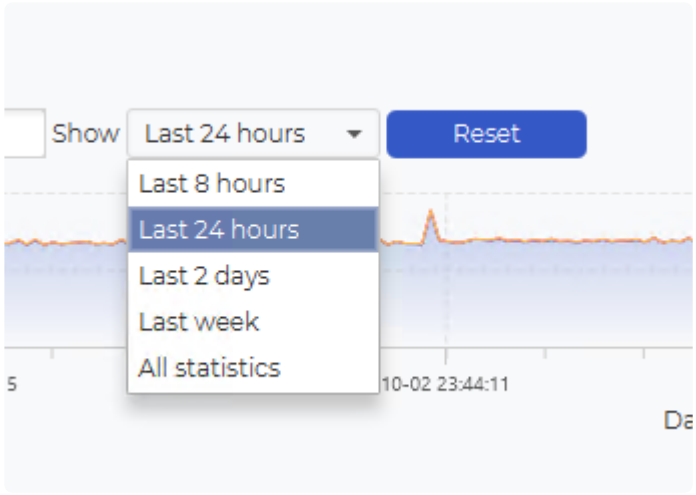
Navigating time

The structure of the SQL Performance dashboard is a top graph showing historical data, such as blocks read, blocks written and blocks dirtied.

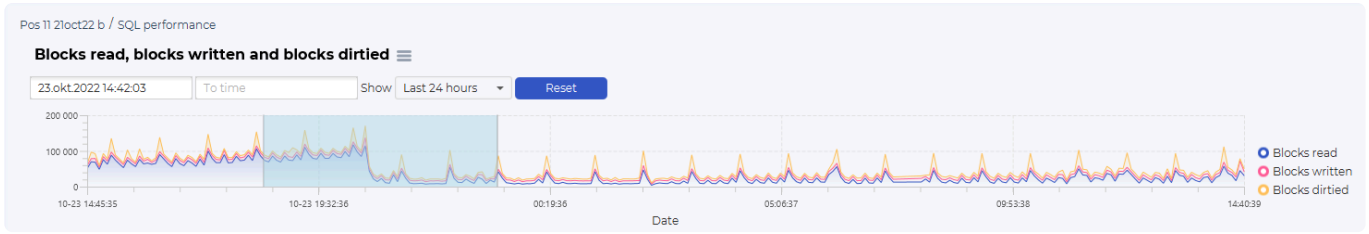


The blocks graph is used to select the timeframe you want to look at.

You can select predefined time selections, such as Last 8 houres, Last 24 houres, Last 2 days, Last week and all statistics:



You can also use the mouse to select a timeframe from the graph. Click on your start time, hold the mouse button, drag to the desired end time, and release the mouse button.



This will change the selection on the SQL statistics table and the additional information graphs.

SQL statistics

The SQL statistics will list SQL statements that has been active in the selected time period.

The fields

RECORDS	DATABASE	USER	DBW	FIRST EXECUTION	LAST EXECUTION	EXECUTIONS	ELAPSED TIME	ROWS	BLOCKS READ	BLOCKS WRITTEN	BLOCKS DIRTIED
282	postgres	dbwatch_1		23.okt.2022 14:45:35	14:40:39	2 297	426 580,38	2 297	713 868	126 681	177 907
281	postgres	dbwatch_1		23.okt.2022 14:45:35	14:40:39	281	368 063,88	281	636 583	126 475	171 938
282	postgres	dbwatch_2		23.okt.2022 14:45:35	14:40:39	755	4 009 084,99	755	548 985	106 387	226 908
281	postgres	dbwatch_1		23.okt.2022 14:45:35	14:40:39	281	59 216,32	281	480 692	20 786	3 325
282	dbwatch	dbw19oct...		23.okt.2022 14:45:35	14:40:39	2 562	377 627,39	2 562	368 327	108 019	74 797
282	testsql4	dbwatch_7		23.okt.2022 14:45:35	14:40:39	2 709	444 217,29	2 709	350 726	107 294	70 052
282	postgres	dbwatch_6		23.okt.2022 14:45:35	14:40:39	2 695	359 428,57	2 695	340 962	107 844	68 869
278	dbwatch	dbw19oct...		23.okt.2022 14:45:35	14:40:39	283	277 512,85	283	318 870	107 029	70 651
829	dbwatch	3 users		23.okt.2022 14:45:35	14:40:39	847	62 112,5	0	318 068	318 068	0
835	postgres	3 users		23.okt.2022 14:45:35	14:40:39	846	62 836,36	0	317 691	317 691	0
282	postgres	dbwatch_2		23.okt.2022 14:45:35	14:40:39	283	231 290,43	283	313 168	106 274	59 974
282	dbwatch	dbw_21oc...		23.okt.2022 14:45:35	14:40:39	2 561	363 249,16	2 561	282 164	105 905	59 208
282	dbwatch	dbw_21oc...	✓	23.okt.2022 14:45:35	14:40:39	2 569	374 466,92	2 569	280 259	106 325	72 698
276	testsql4	dbwatch_7		23.okt.2022 14:45:35	14:40:39	282	267 158,28	282	262 156	106 089	63 245
272	dbwatch	dbw_21oc...		23.okt.2022 14:45:35	14:40:39	282	244 052,42	282	256 460	105 905	55 408

Records:
The number of times this SQL statements has been detected by dbWatch

Database:
The database executing the SQL or procedure

User:
The user executing the SQL or procedure

DBW:
Is this a SQL or procedure associated with dbWatch

First execution:
The first time **in the selected time frame** where this SQL or procedure was executed

Last execution:
The last time **in the selected time frame** where this SQL or procedure was executed

Executions:
The number of times **in the selected time frame** where this SQL or procedure was executed

Elapsed time:
The total elapsed time (in milliseconds) **in the selected time frame** where this SQL or procedure was executed

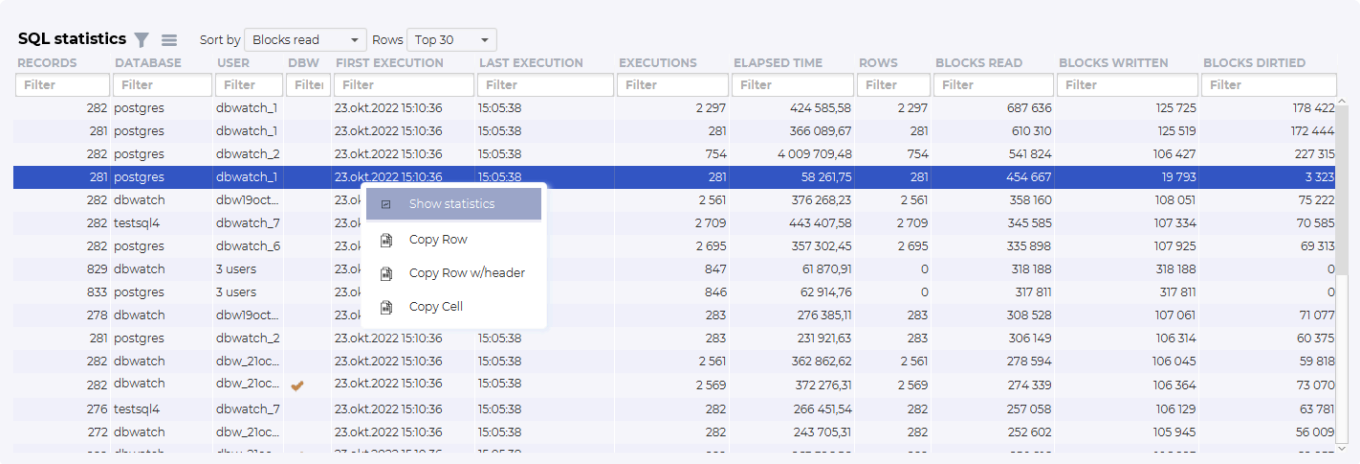
Rows:
The total number of rows **in the selected time frame** where this SQL or procedure was executed

Blocks read:
The total number of blocks read **in the selected time frame** where this SQL or procedure was executed

Blocks written:
The total number of blocks written **in the selected time frame** where this SQL or procedure was executed

Blocks dirtied:
The total number of blocks dirtied **in the selected time frame** where this SQL or procedure was executed

Navigating SQL statistics



You can investigate the SQL further by right-clicking on one of the lines. Show statistics will open a new dashboard focusing on one SQL statement.

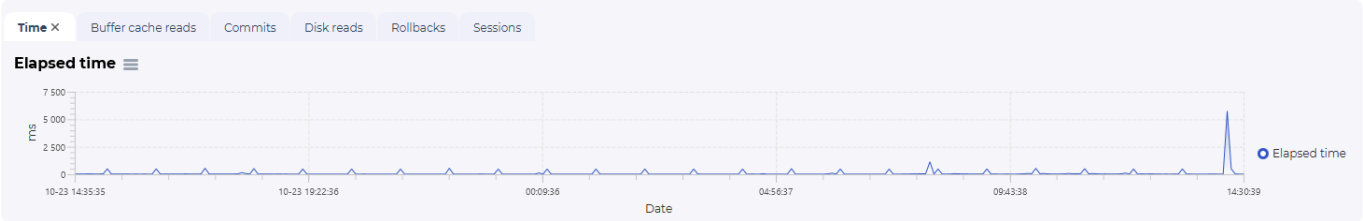
This dashboard will show blocks statistics for this statement, and the statistics for the SQL code that has been run.

Additional graphs

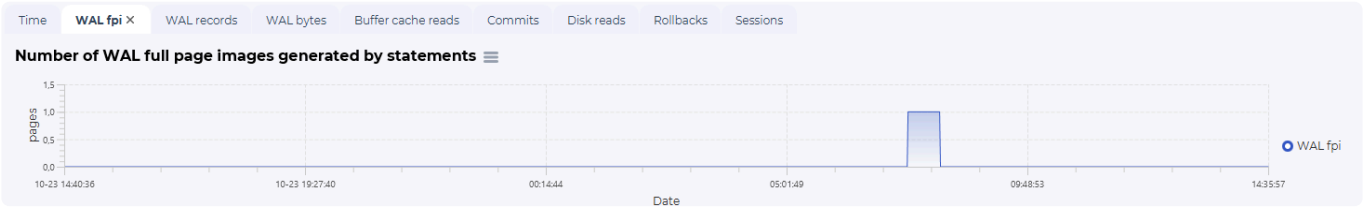
In addition to the logical/physical reads and writes and the SQL statistics overview, there are additional graphs to visualize other performance statistics from the same selected timeframe.

Examples:

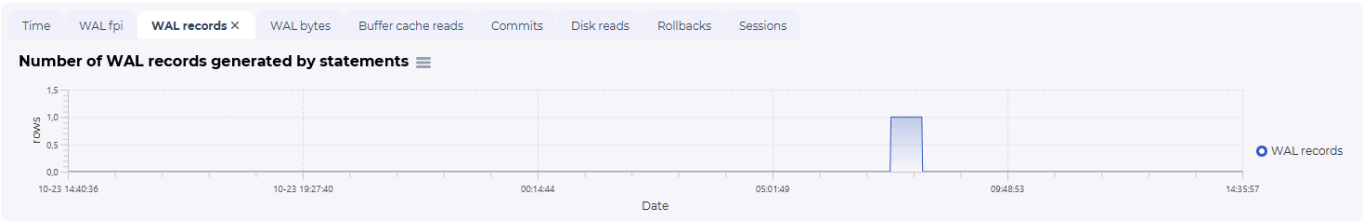
Elapsed time:



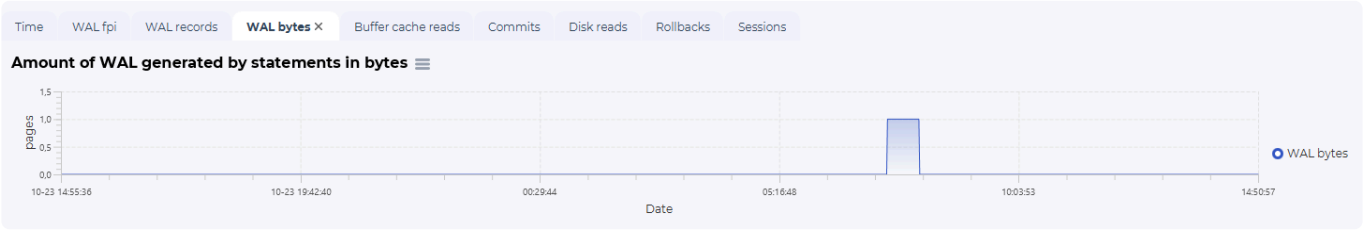
WAL fpi (only Postgres 14 and newer):



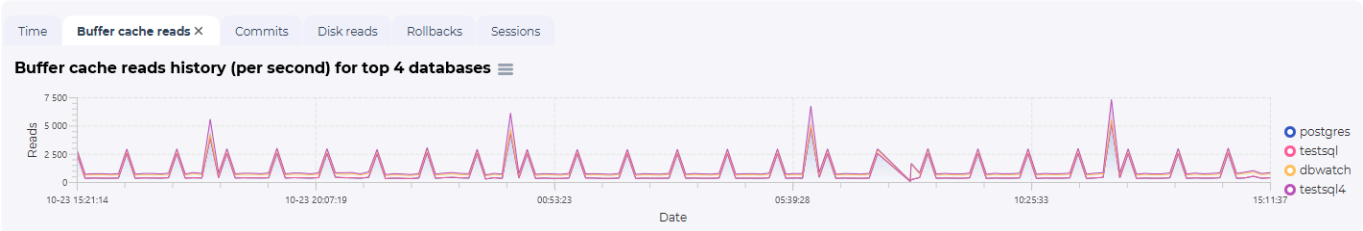
WAL records (only Postgres 14 and newer):



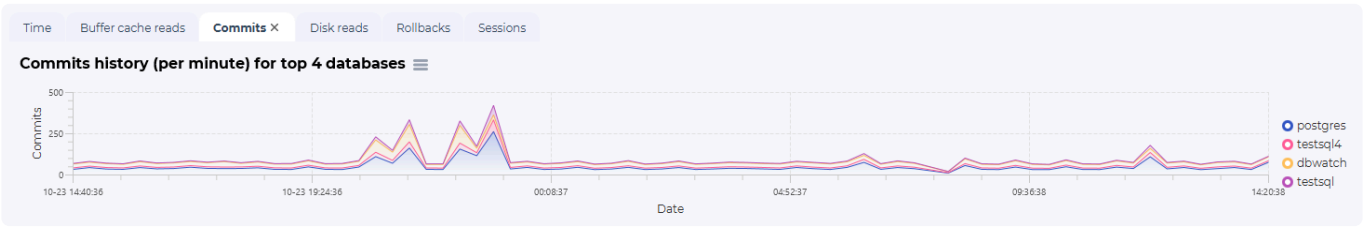
WAL bytes (only Postgres 14 and newer):



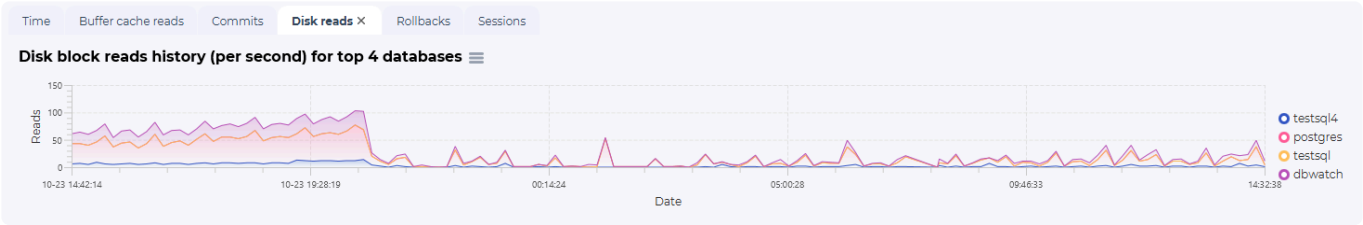
Buffer cache reads:



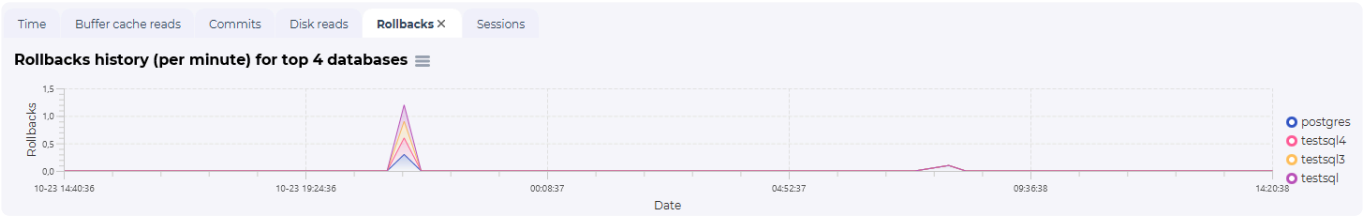
Commits:



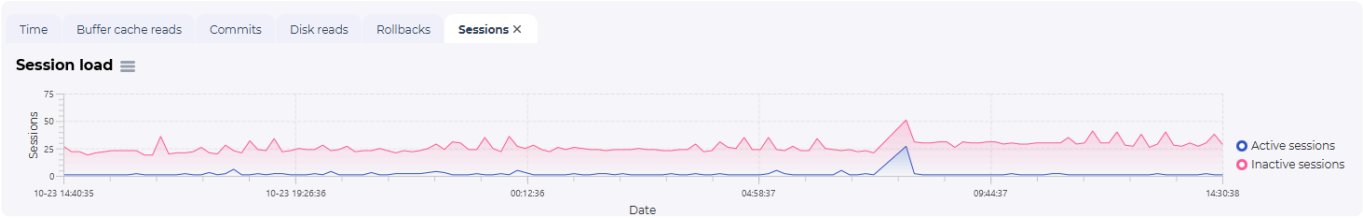
Disk reads:



Rollbacks:



Sessions:

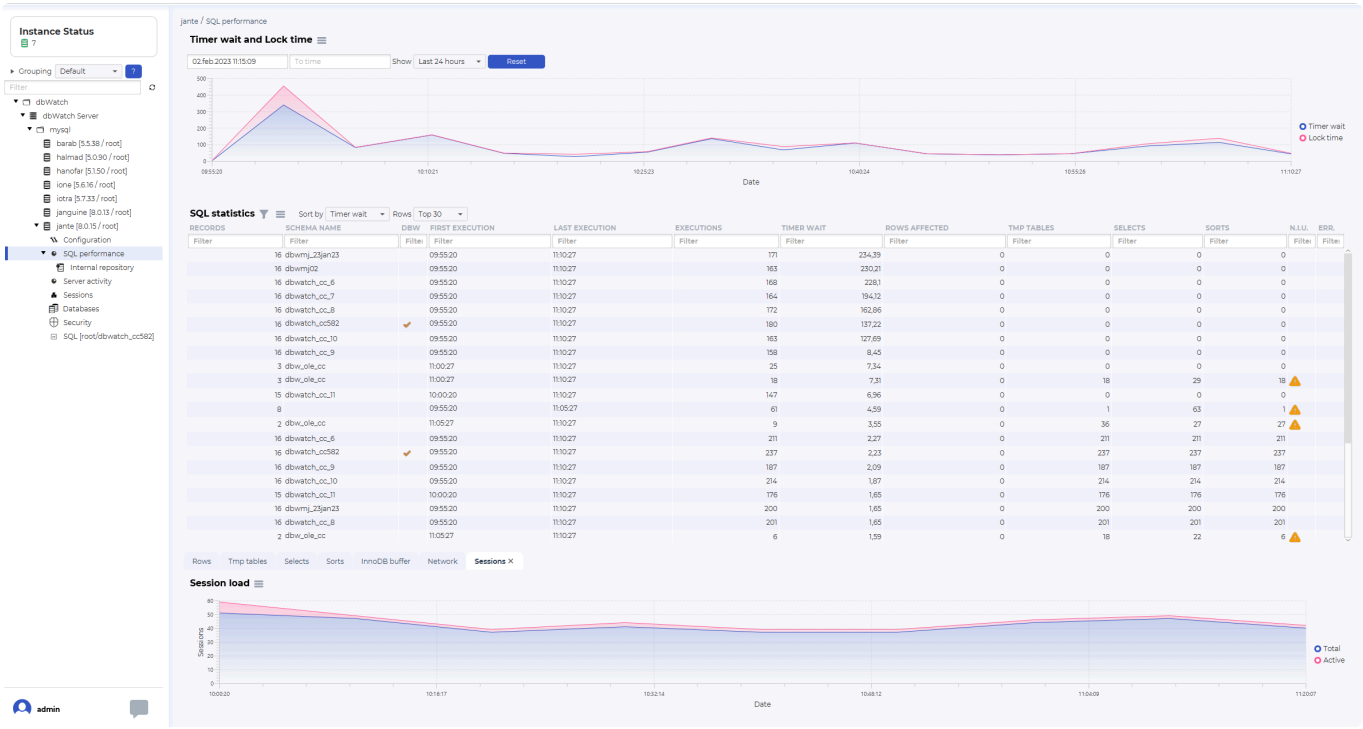


[<< Using SQL Performance package on Oracle] / [Using SQL Performance package on MySQL >>]

Using SQL Performance package on MySQL

Navigating to the SQL Performance package dashboard

Once the SQL statistics job is installed and have collected data, you can navigate to the dashboard. It is located in the management interface for that database instance. The pictures and descriptions are from MySQL.

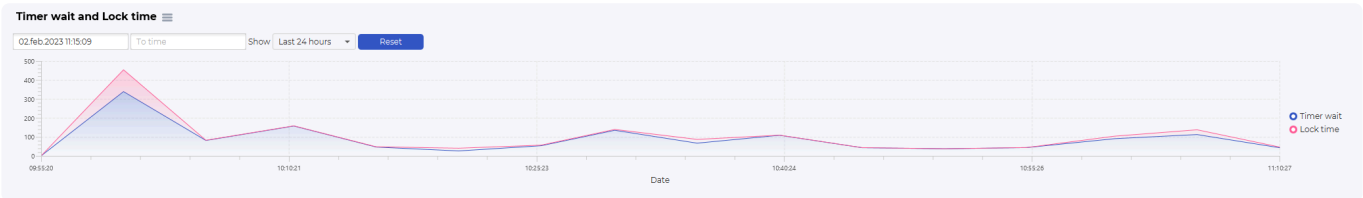


There should be a “SQL performance” node in the three structure. Click on it to open the SQL Performance dashboard.

Using the SQL Performance dashboard

Navigating time

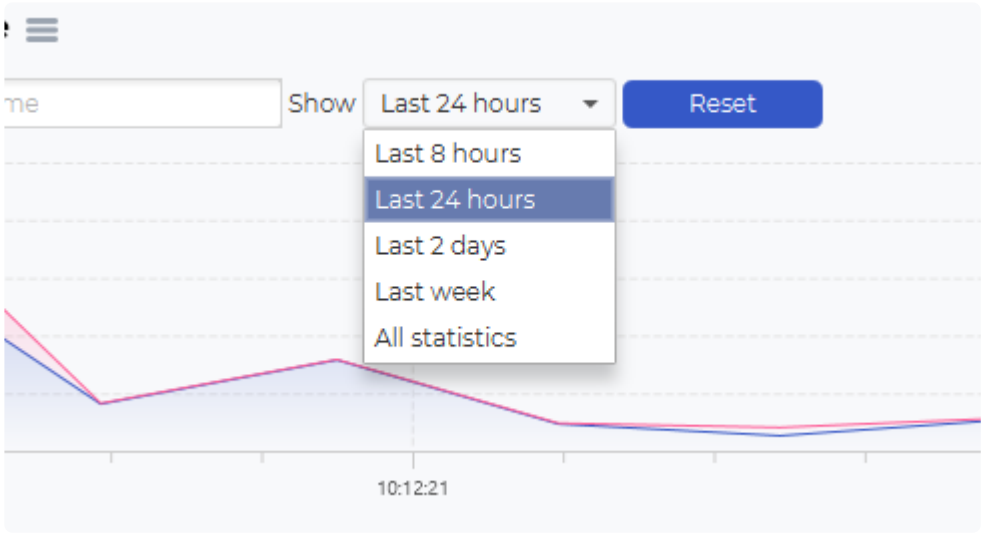
The structure of the SQL Performance dashboard is a top graph showing historical data, such as timer wait and lock time.



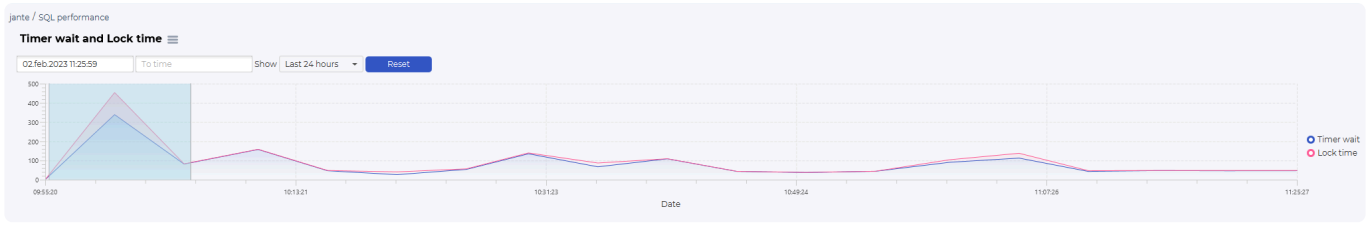
The blocks graph is used to select the timeframe you want to look at.

You can select predefined time selections, such as Last 8 houres, Last 24 houres, Last 2 days, Last

week and all statistics:



You can also use the mouse to select a timeframe from the graph. Click on your start time, hold the mouse button, drag to the desired end time, and release the mouse button.

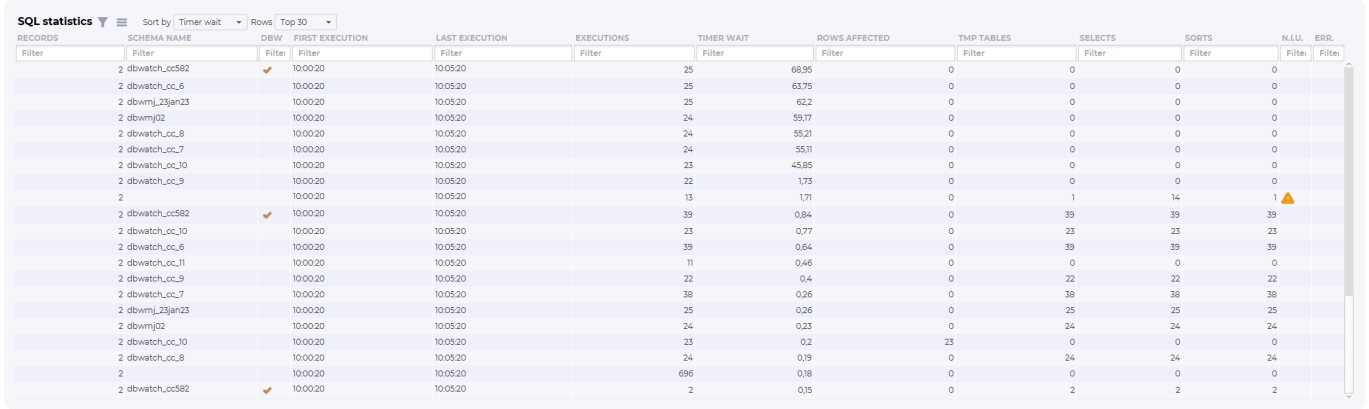


This will change the selection on the SQL statistics table and the additional information graphs.

SQL statistics

The SQL statistics will list SQL statements that has been active in the selected time period.

The fields



Records:
The number of times this SQL statements has been detected by dbWatch

Schema name:

The schema name executing the SQL or procedure

DBW:
Is this a SQL or procedure associated with dbWatch

First execution:
The first time **in the selected time frame** where this SQL or procedure was executed

Last execution:
The last time **in the selected time frame** where this SQL or procedure was executed

Executions:
The number of times **in the selected time frame** where this SQL or procedure was executed

Timer wait:
The total wait time (in milliseconds) **in the selected time frame** where this SQL or procedure was executed

Rows affected:
The total number of rows affected*in the selected time frame* where this SQL or procedure was executed

TMP tables:
The number of tmp tables created **in the selected time frame** where this SQL or procedure was executed

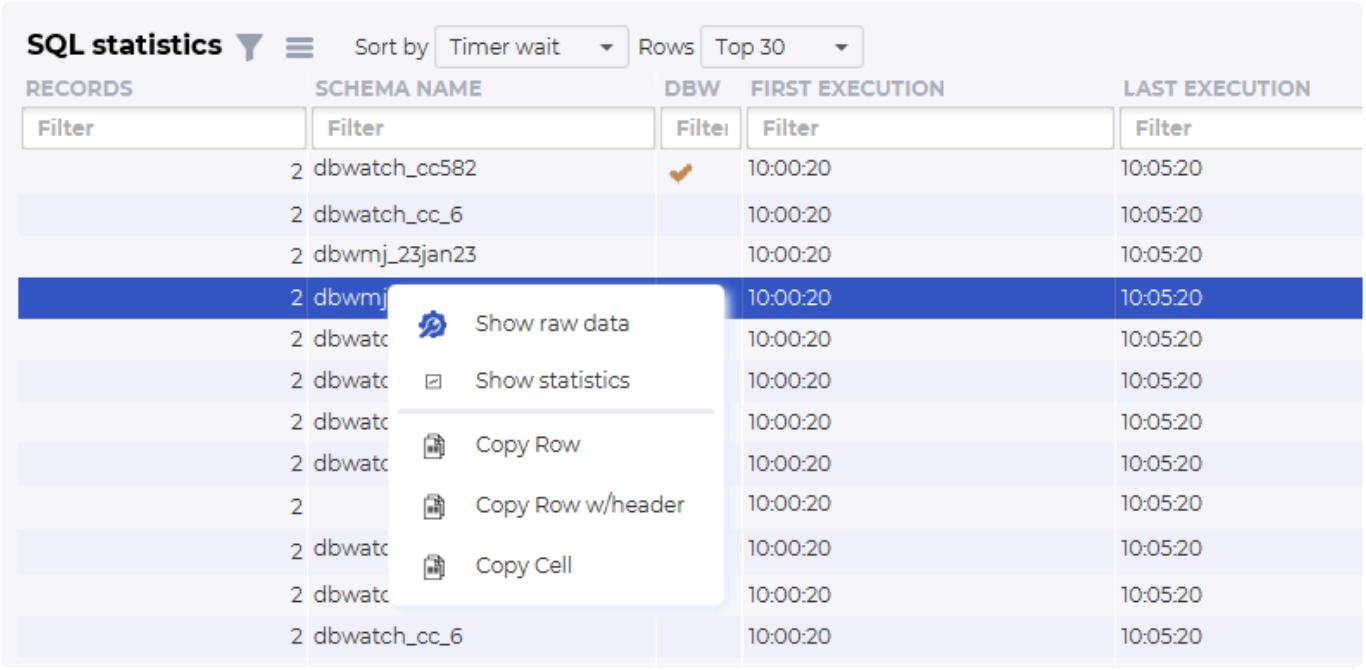
Selects:
The number of selects **in the selected time frame** where this SQL or procedure was executed

Sorts:
The number of sorts **in the selected time frame** where this SQL or procedure was executed

N.I.U.:
Indicating Not In Use indexes.

Err.:
Indicating errors with this query.

Navigating SQL statistics



You can investigate the SQL further by right-clicking on one of the lines. Show statistics will open a new dashboard focusing on one SQL statement.

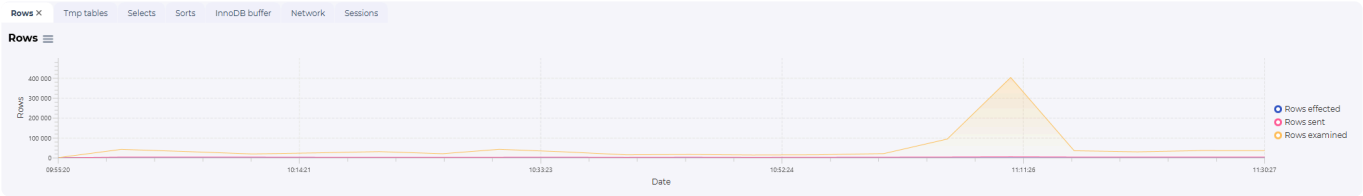
This dashboard will show timer wait and lock time statistics for this statement, and the statistics for the SQL code that has been run.

Additional graphs

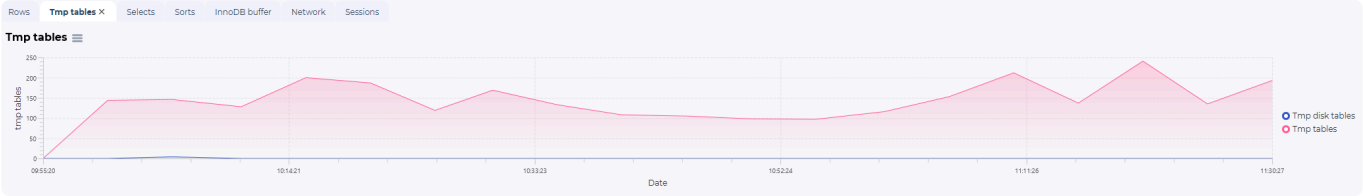
In addition to the logical/physical reads and writes and the SQL statistics overview, there are additional graphs to visualize other performance statistics from the same selected timeframe.

Examples:

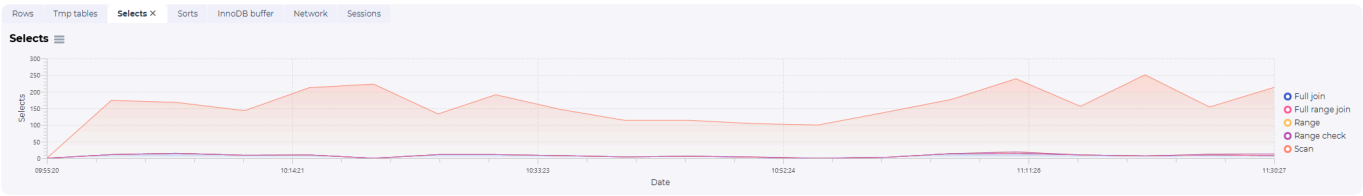
Rows statistics:



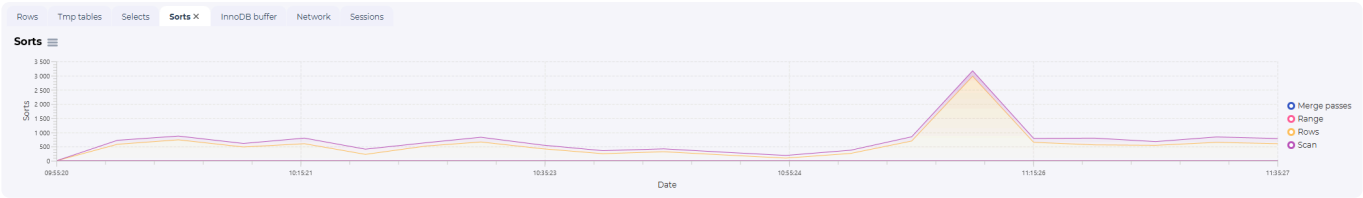
TMP tables statistics:



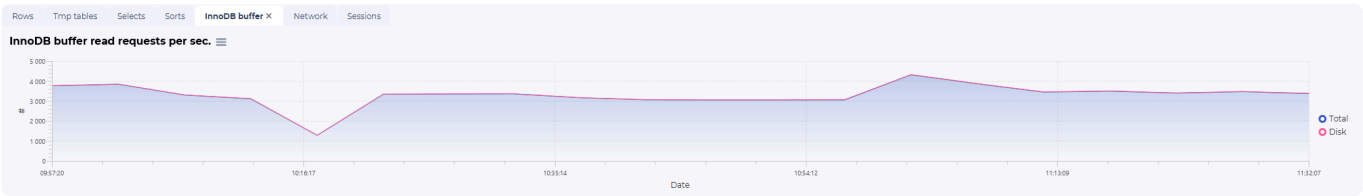
Selects statistics:



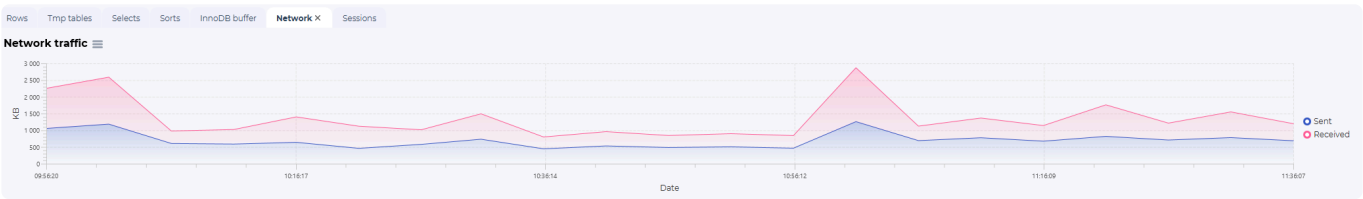
Sort statistics:



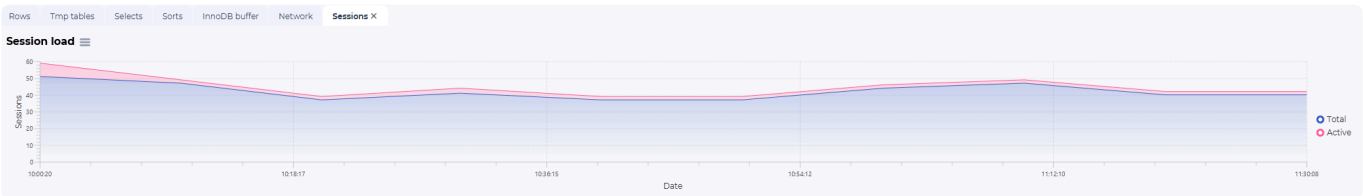
InnoDB buffer statistics:



Network usage statistics:



Sessions load:



[<< Using SQL Performance package on PostgreSQL] / [Repository management and monitoring >>]

Repository management and monitoring

Introduction

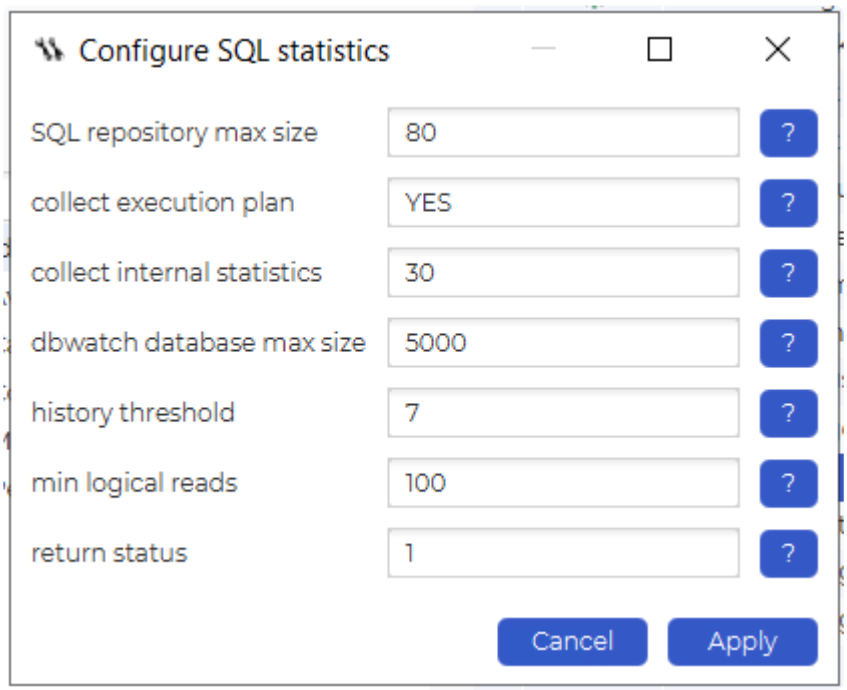
The SQL Performance package creates tables in the dbWatch schema/database on each database instance where the SQL Statistics job is installed and scheduled.

The process combines two parts of dbWatch Control Center. The monitoring is used to schedule the job responsible for gathering performance data and maintaining the repository, and the management is used to visualize and provide drill down for the DBA analyzing the SQL performance data gathered. As there can be much data gathered, there is both configuration possibilities for the SQL Statistics job to allow the job to gather more or less data, and also management interface to look into and fine-tune the inner workings of the SQL Statistics gathering process.

This process follows the same framework for all database platforms supported, and the process of management and monitoring of the SQL Performance job is very similar.

Most installations will fall well within the normal usage, and not require any adjustments.

Configuring the SQL Statistics job



There are several parameters that can be adjusted. When SQL Performance package is initially deployed, the DBA must keep track of space used by dbWatch to keep the performance statistics, so it does not reach the maximum configured level. This will usually stabilize when **history threshold** days of statistics have been gathered.

SQL repository max size, default (80), is the maximum space consumption in percentage of maximum size of dbWatch database that can be used for the SQL Performance package. This is not set to 100 (100%), to avoid SQL Performance data filling up all the space and stopping other monitoring. Once this

limit is reached, and **dbwatch database max size** does not allow us to extend the space usage any further, SQL Performance package will go into warning and stop collecting new statistics.

collect execution plan, default (YES), allows turning of collection of SQL plan data. This will save disk space.

collect internal statistics, default (30), is how often, in minutes, the size of internal objects is checked. This is used to track the space used by SQL Performance package and is important to make sure the database is not filled up.

dbwatch database max size. default (5000), is the size in MB that the dbWatch database is allowed to grow to. The SQL Performance job will adjust the database size if needed, but just up-to this max value. The space used will differ between databases based on activity.

history threshold, default (7), is the time period we keep data in, in days. until data is removed from the repository. When adjusting this to a higher value, you might also need to adjust **dbwatch database max size** to allow for more statistics to be gathered.

min logical reads, default (100), is a cut of value, in the number of logical reads per execution, where we don’t store statistics about this query. This is to avoid gathering a lot of statistics about queries that does not consume a lot of resources.

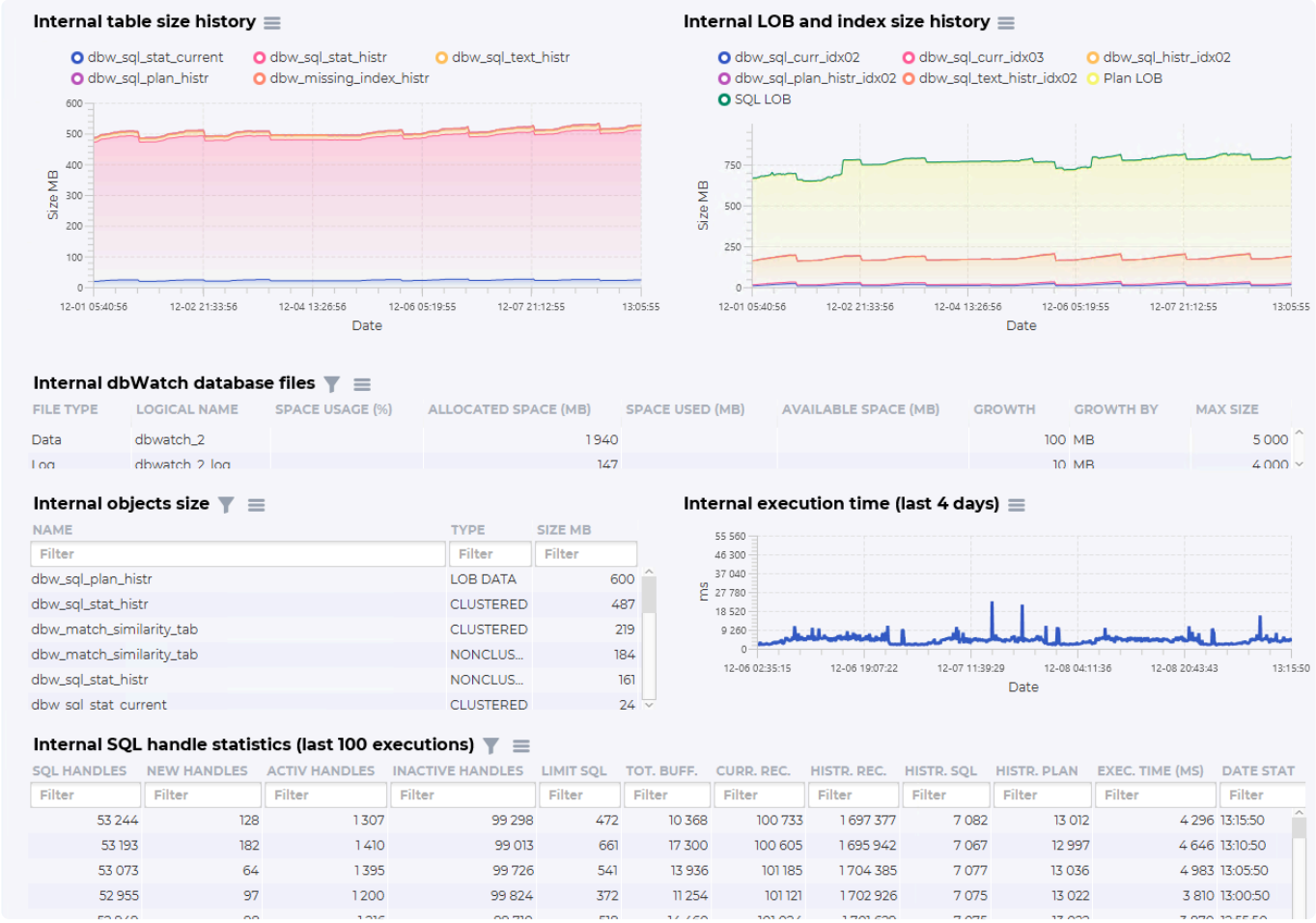
return status. default (1) which is Warning, is the status we will raise when the SQL statistics job run into problems with space consumption. (Alarm = 2, Warning = 1, OK = 0)

Clicking on the question mark next to the parameter will display information about the parameter and what it adjusts.

The inner-workings dashboard for the SQL Statistics job

In management, under the SQL Performance node, there is an “Internal repository” node that gives insight into the inner-workings of the SQL statistics job.

This is used to keep track of the process, time used, what statistics are gathered, and space usage.



The “Internal table size history” graph displays the tables and space usage. In a freshly installed system or one where **history threshold** is increased this could display a growing trend. It will usually stabilize once **history threshold** days of statistics are gathered.

The “Internal LOB and index size history” graph displays the LOB’s (such as plan data) and index space usage. In a freshly installed system or one where **history threshold** is increased this could display a growing trend. It will usually stabilize once **history threshold** days of statistics are gathered.

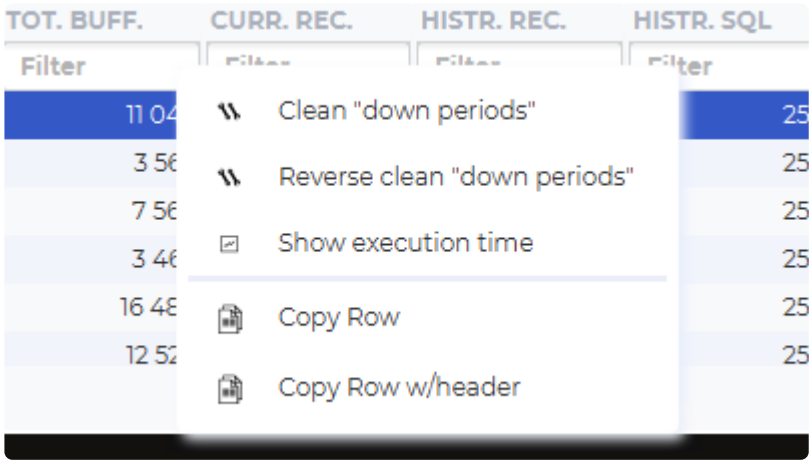
The “Internal dbWatch database files” is a table to see space usage within the allocated space for both the database file and the transaction log file.

The “Internal objects size” lists the size of the largest internal objects associated with the SQL Performance package.

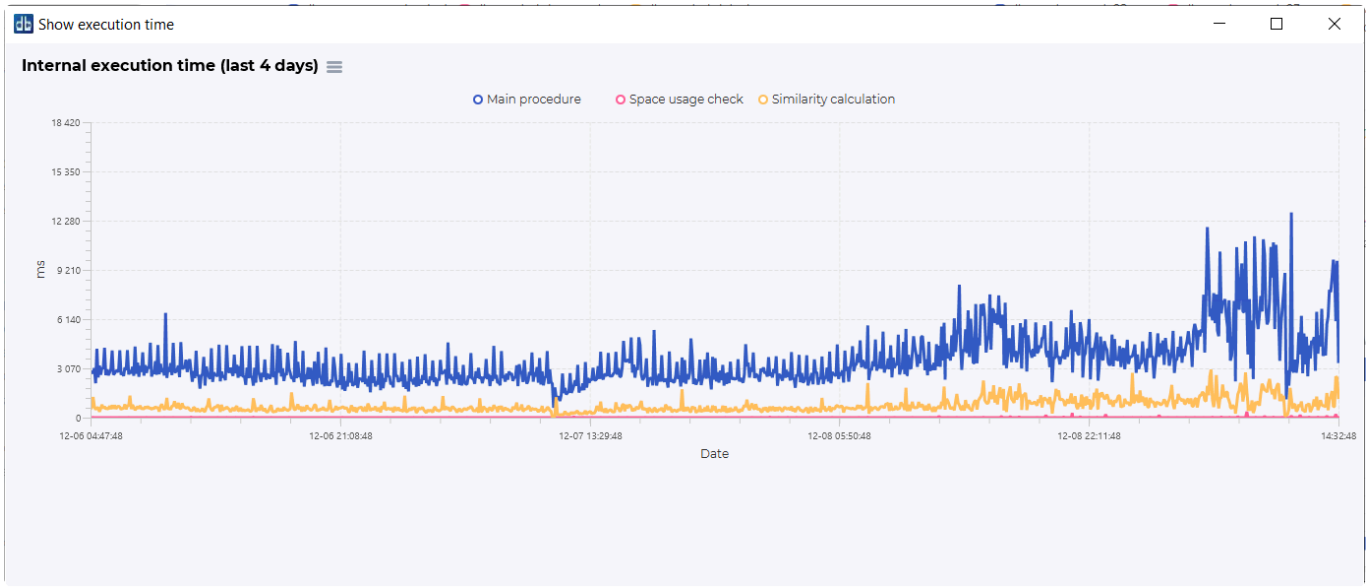
The “Internal execution time (last 4 days)” graph shows the execution time for the gathering of SQL performance statistics. Consistent (not spikes) values above 60000 are considered too high and might warrant contacting support, so the situation can be analyzed.

The “Internal SQL handle statistics (last 100 executions)” table displays statistics on the gathering of SQL Performance data.
The field “SQL handle” is the number of current SQL handles in the memory of the database instance.
“New handles” is new SQL handles since the last run of the SQL Performance job.
“Active handles” are SQL handles that are known, but have been active since the last run.
“Inactive handles” are the total amount of SQL handles we collect statistics on but were not active in the

last time period.
“Limit SQL” is the number of SQL handles that are cut due to the **min logical reads** parameter (default 100).
“Tot. Buff.” is the total amount of logical reads from all the SQL handles that are cut off.
“Curr. Rec” is the total number of SQL handles we have a record about.
“Histr. Rec” is the total number of records in our history table
“Histr. SQL” is the total number of unique SQL statements texts in the SQL Handles we are tracking.
“Histr. PLAN” is the total number of unique SQL plans we are storing.
“Exec. Time (MS)” is the execution time for our SQL Perfromance job.
“Date stat” is the time when these statistics were gathered.



Right-click on the table will show a menu. The options “Clean down periods” and “Reverse clean periods” is used in troubleshooting statistics issues and the option “Show execution time” will open the execution time graph.

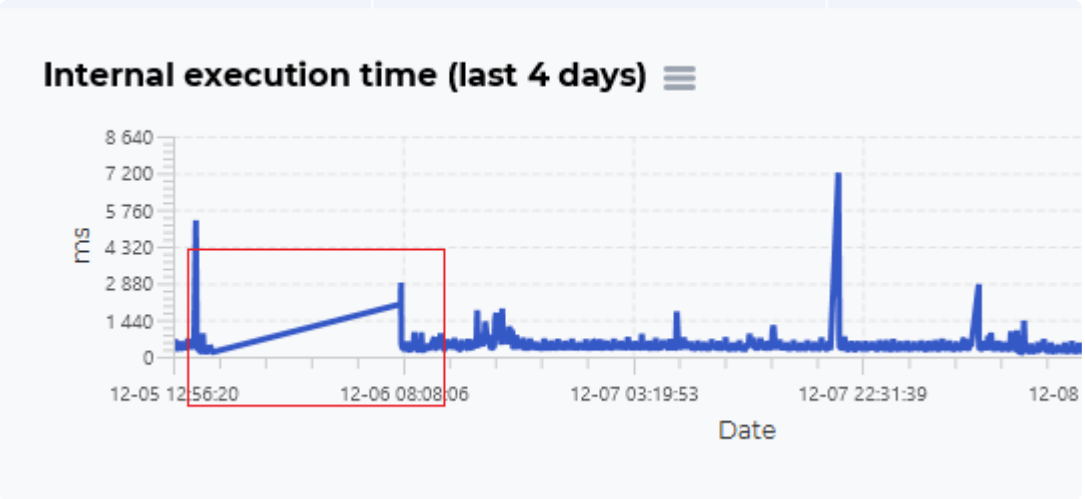


The “Show execution time” graph displays how much time is spent on the different procedures that make up the SQL Performance job, and is used to analyze long execution times.

Troubleshooting issues in SQL Performance package

Graph issues with monitoring has been switched off

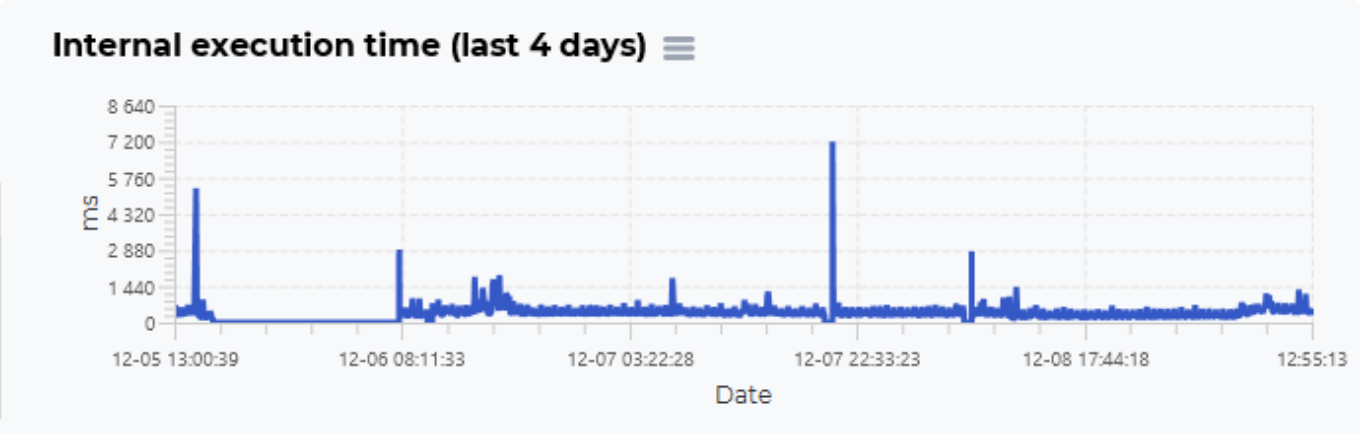
If for some reason dbWatch Control Center has not been connected to a database instance running SQL Performance package for an extended time, you can get odd statistics errors in the graphs. It will look like a linear gradual increase in number. This is due to statistics being based on a ever-increasing value, and we take snapshots of this value to determine the changes in a period.



Note that the internal execution time graph here displays this problem.

If you right-click on the “Internal SQL statistics” table, you can choose to “Clean down periods” to recalculate this correctly.

After this cleaning, the graph will look more like this:



Where the data is modified to express that we have no values in this time period.

Security and Compliance package

Security and Compliance package

The Security and Compliance package is an optional extra cost module in dbWatch Control Center that allows automated detection and reporting on security and compliance-related issues where database instances are not conforming to best practices.

Currently supported platforms:

- MS SQL Server 2014 and newer

Who is it for

Database Administrators (DBAs): Professionals responsible for managing and maintaining SQL Server databases, ensuring their security and compliance with industry standards.

IT Security Professionals: Individuals specializing in protecting IT environments from vulnerabilities, ensuring systems like SQL Server are secure against potential threats.

Compliance Officers: Those tasked with ensuring that SQL Server environments comply with relevant laws, regulations, and industry standards.

IT Managers and Decision Makers: Leaders who oversee IT infrastructure, need comprehensive insights into the security and compliance posture of their organization's SQL Server deployment.

Organizations with SQL Server Deployments: Enterprises, small businesses, and other entities that rely on Microsoft SQL Server for their database needs and prioritize maintaining a secure and compliant IT environment.

This package is ideal for those seeking to enhance their SQL Server's security, ensure compliance, and gain valuable insights for informed decision-making in IT management.

Installation and setup

Installation steps are identical for all platforms, but the data displayed and some functionality can be platform-dependent.

<< Troubleshooting issues in SQL Performance package / Installation of the Security and Compliance package >>

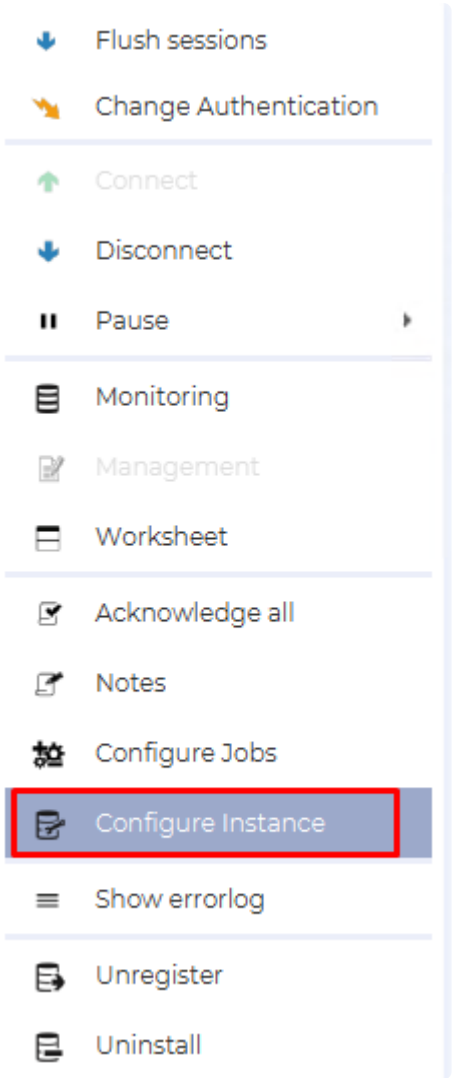
Installation of the Security and Compliance package

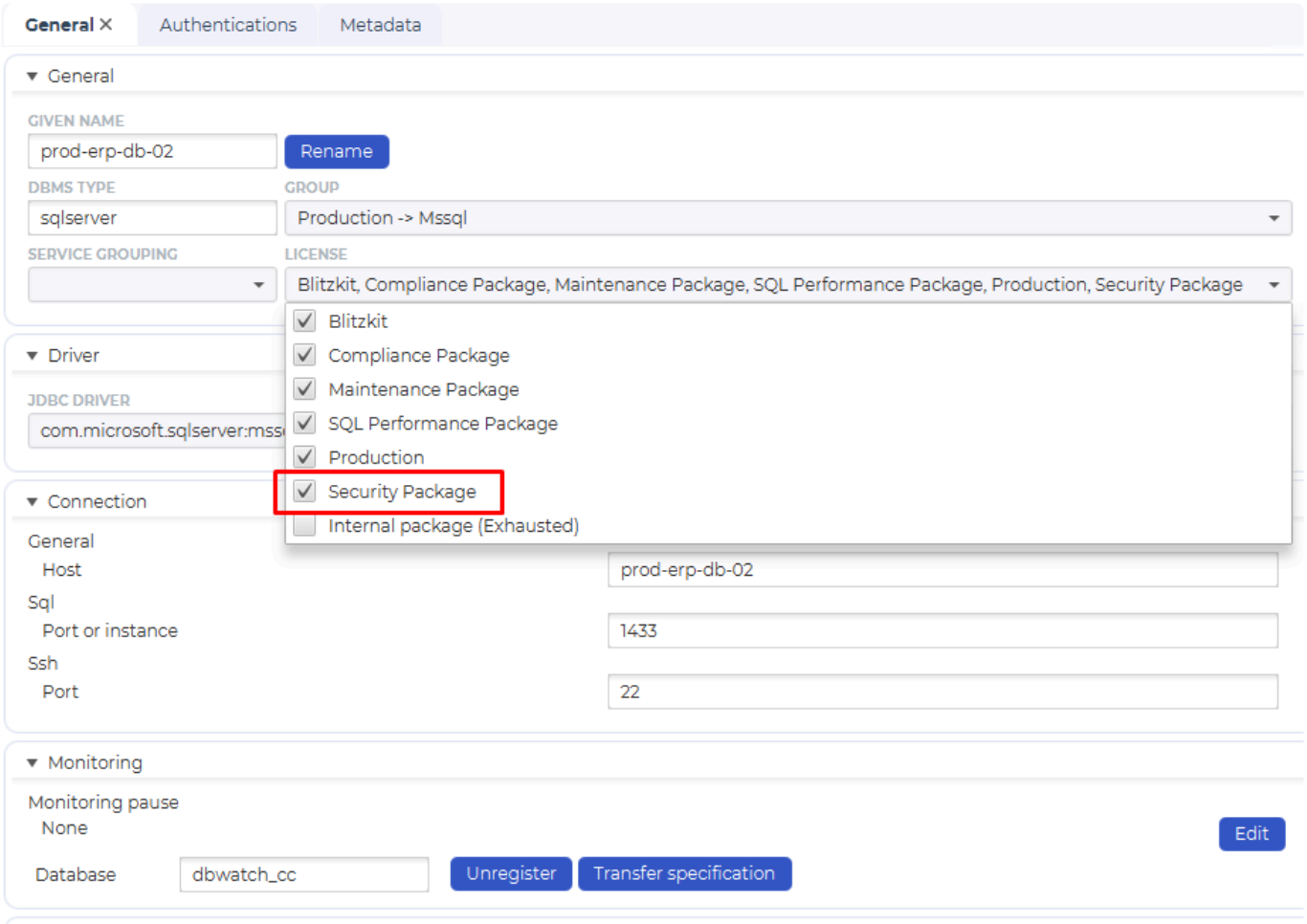
Installation of the Security and Compliance package

The Security and Compliance package is a premium package that comes with extra cost. If you want to get this package and its not part of your current license, please contact sales@dbwatch.com.

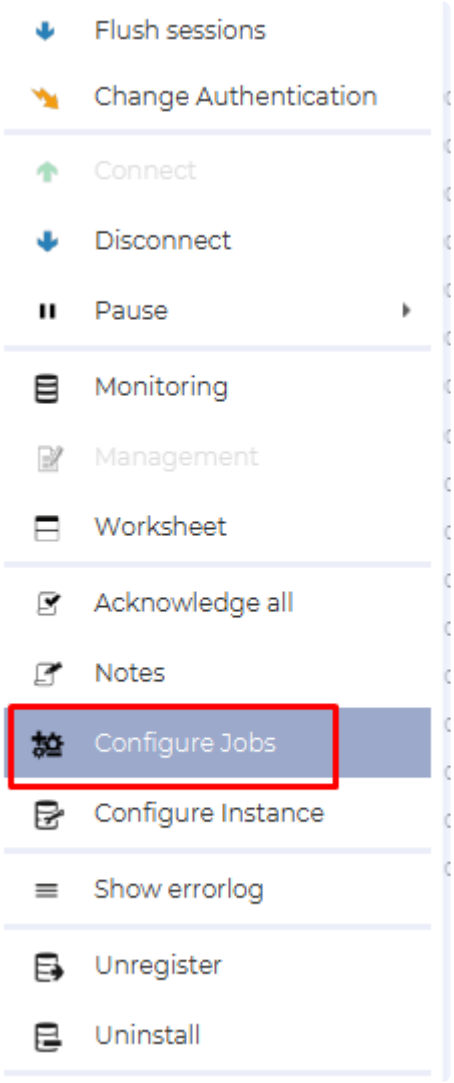
To install the “Security and Compliance” module, you must first add the license to the instance.

Select “Configure Instance” and then “LICENSE”.



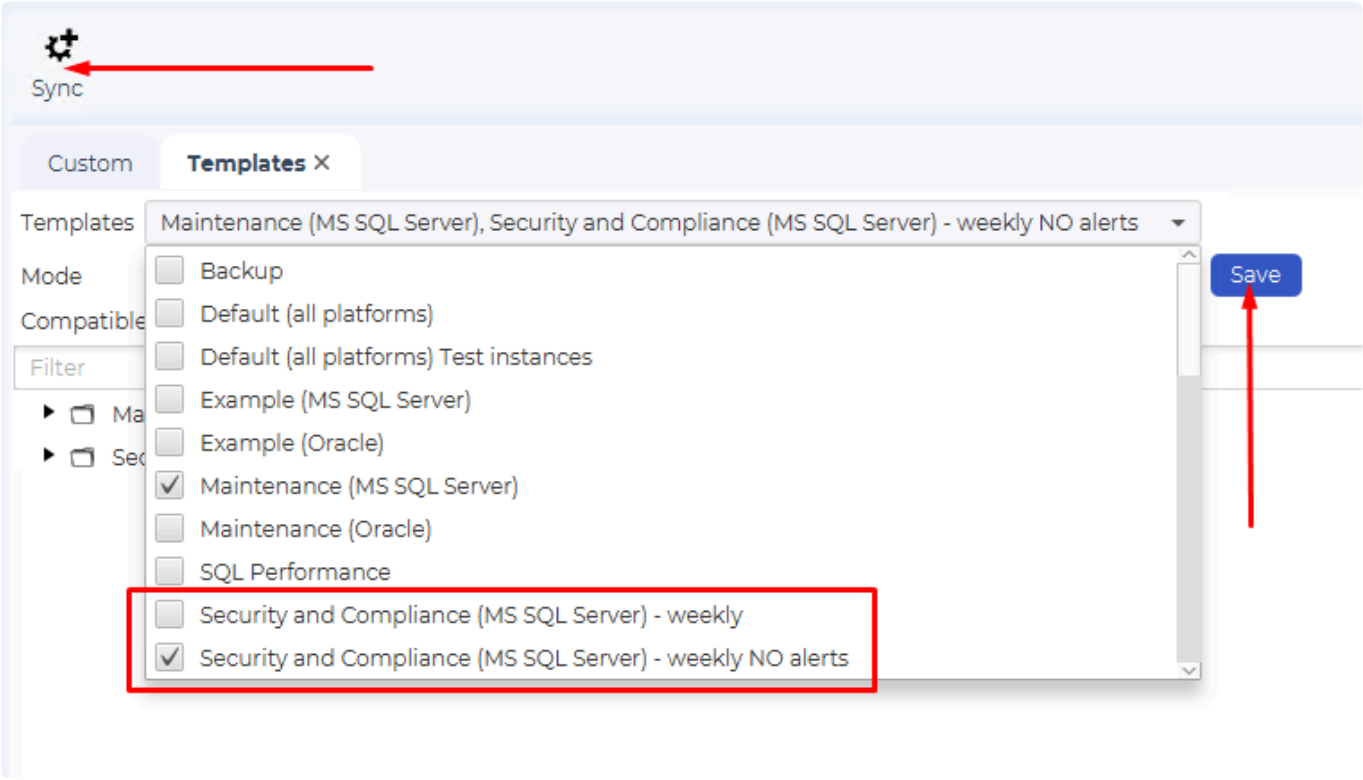


To deploy the Security and Compliance monitoring jobs, you need to deploy the template with the jobs to the instance you want to be monitored. Right-click on the instance name, and select “Configure jobs”



Select the “Templates” tab, choose one of the Security templates, then click “Save” and “Sync”. Pressing “Sync” initiates the installation of all the jobs defined in the Security template. There are a total of 40 jobs included.

By default, these are scheduled to run once a week, in the early hours of Monday, between 1:00 AM and 2:00 AM.



Once the deployment is complete the jobs will be visible in the monitoring interface and can be triggered manually if you don't want to wait for the normal schedule.

[<< Security and Compliance package] / [Security and Compliance package on SQL Server >>]

Security and Compliance package on SQL Server

The Security and Compliance package

The Security and Compliance package provides a detailed evaluation of your Microsoft SQL Server's security and compliance, managed via dbWatch Control Center. It offers in-depth insights into your server's security and compliance state, covering system configuration, adherence to industry norms, and potential vulnerabilities. The results include identified risk areas and practical recommendations to enhance security and ensure regulatory compliance. The aim is to equip you with the necessary information and tools for effective decision-making, risk mitigation, and proactive protection of your SQL Server environment.

Package contents

The Security and Compliance package includes the following:

Security and Compliance Assessment Report
Security and Compliance dashboards
38 new monitoring jobs that detect and alert on security and compliance-related issues
2 Monitoring templates

The Security and Compliance report

This report, comprised of five chapters, offers a comprehensive analysis of your Microsoft SQL Server's security and compliance status, as managed by the dbWatch Control Center. It aggregates data from all 40 "Security and Compliance" jobs (or those installed) and is tailored to be run against a specific instance.

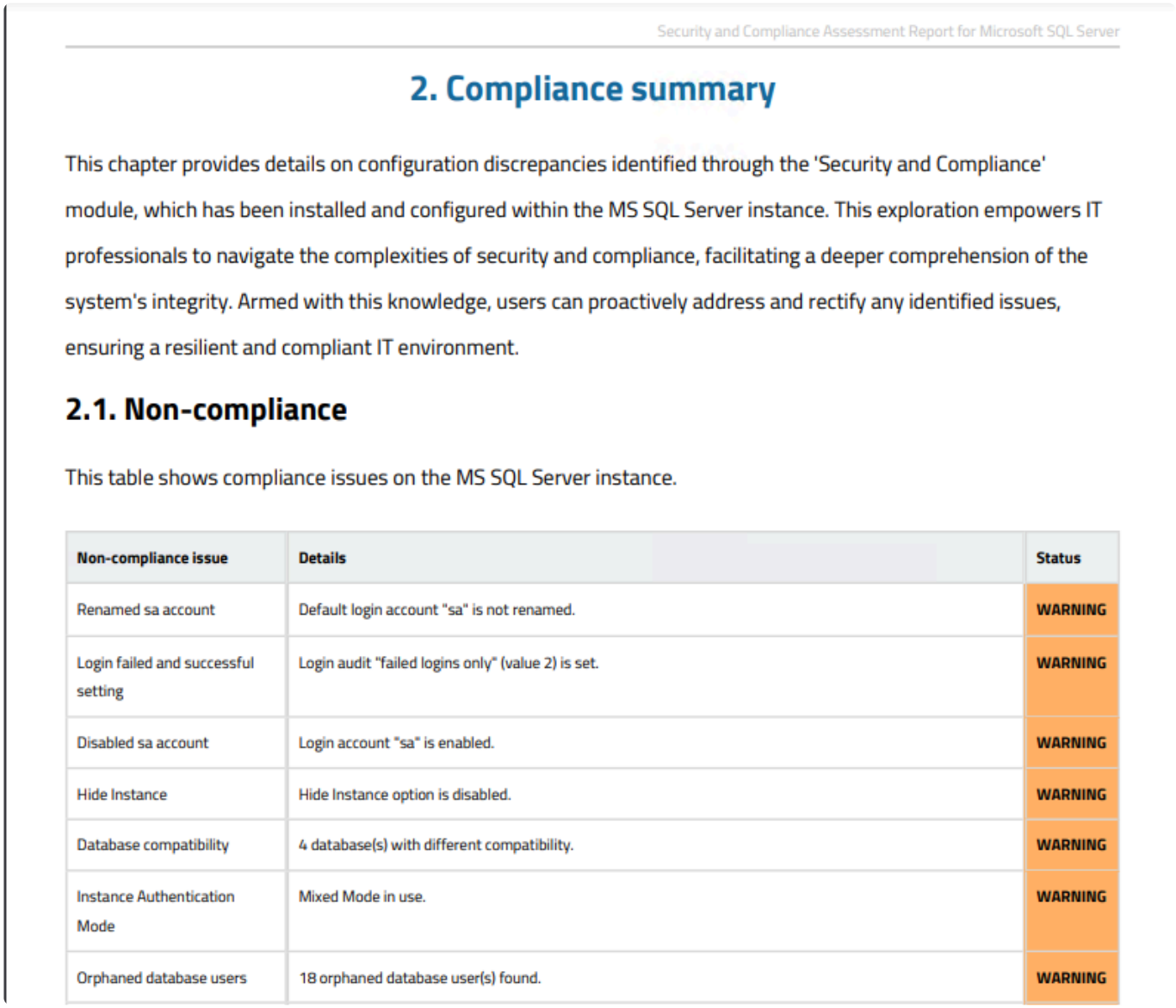
Overview

This document acts as a thorough evaluation of your SQL Server environment's security and compliance framework, facilitated through dbWatch Control Center. It provides a detailed examination of your deployment's security state and compliance level. Our review covers a full inspection of your system's setup, conformity with established industry protocols, and identification of any vulnerabilities that could threaten your data security. The findings of our analysis are clearly presented, pinpointing critical areas and offering practical advice for enhancing your security measures and meeting regulatory standards. Our objective is to equip you with essential insights and tools that enable well-informed decisions, reduce risk factors, and actively safeguard your SQL Server environment.

Compliance summary

This chapter provides details on configuration discrepancies identified through the 'Security and Compliance'
module, which has been installed and configured within the MS SQL Server instance. This exploration

empowers IT
professionals to navigate the complexities of security and compliance, facilitating a deeper comprehension of the
system’s integrity. Armed with this knowledge, users can proactively address and rectify any identified issues,
ensuring a resilient and compliant IT environment



Security and Compliance Assessment Report for Microsoft SQL Server

2. Compliance summary

This chapter provides details on configuration discrepancies identified through the 'Security and Compliance' module, which has been installed and configured within the MS SQL Server instance. This exploration empowers IT professionals to navigate the complexities of security and compliance, facilitating a deeper comprehension of the system's integrity. Armed with this knowledge, users can proactively address and rectify any identified issues, ensuring a resilient and compliant IT environment.

2.1. Non-compliance

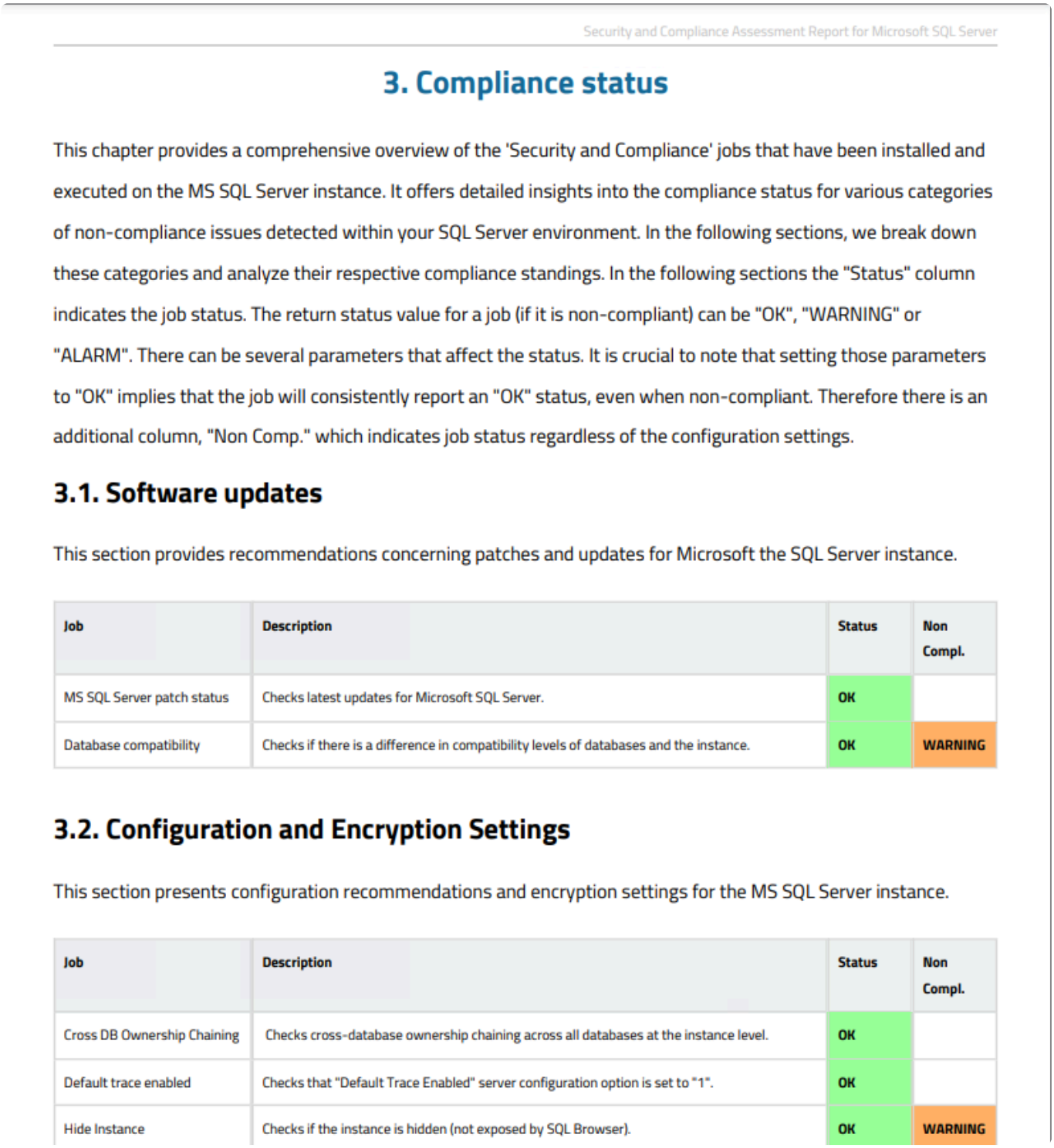
This table shows compliance issues on the MS SQL Server instance.

Non-compliance issue	Details	Status
Renamed sa account	Default login account "sa" is not renamed.	WARNING
Login failed and successful setting	Login audit "failed logins only" (value 2) is set.	WARNING
Disabled sa account	Login account "sa" is enabled.	WARNING
Hide Instance	Hide Instance option is disabled.	WARNING
Database compatibility	4 database(s) with different compatibility.	WARNING
Instance Authentication Mode	Mixed Mode in use.	WARNING
Orphaned database users	18 orphaned database user(s) found.	WARNING

Compliance status

This chapter provides a comprehensive overview of the ‘Security and Compliance’ jobs that have been installed and
executed on the MS SQL Server instance. It offers detailed insights into the compliance status for various categories
of non-compliance issues detected within your SQL Server environment. In the following sections, we break down
these categories and analyze their respective compliance standings. In the following sections the “Status” column
indicates the job status. The return status value for a job (if it is non-compliant) can be “OK”, “WARNING” or

“ALARM”. There can be several parameters that affect the status. It is crucial to note that setting those parameters
to “OK” implies that the job will consistently report an “OK” status, even when non-compliant. Therefore there is an
additional column, “Non Comp.” which indicates job status regardless of the configuration settings.



Security and Compliance Assessment Report for Microsoft SQL Server

3. Compliance status

This chapter provides a comprehensive overview of the 'Security and Compliance' jobs that have been installed and executed on the MS SQL Server instance. It offers detailed insights into the compliance status for various categories of non-compliance issues detected within your SQL Server environment. In the following sections, we break down these categories and analyze their respective compliance standings. In the following sections the "Status" column indicates the job status. The return status value for a job (if it is non-compliant) can be "OK", "WARNING" or "ALARM". There can be several parameters that affect the status. It is crucial to note that setting those parameters to "OK" implies that the job will consistently report an "OK" status, even when non-compliant. Therefore there is an additional column, "Non Comp." which indicates job status regardless of the configuration settings.

3.1. Software updates

This section provides recommendations concerning patches and updates for Microsoft the SQL Server instance.

Job	Description	Status	Non Compl.
MS SQL Server patch status	Checks latest updates for Microsoft SQL Server.	OK	
Database compatibility	Checks if there is a difference in compatibility levels of databases and the instance.	OK	WARNING

3.2. Configuration and Encryption Settings

This section presents configuration recommendations and encryption settings for the MS SQL Server instance.

Job	Description	Status	Non Compl.
Cross DB Ownership Chaining	Checks cross-database ownership chaining across all databases at the instance level.	OK	
Default trace enabled	Checks that "Default Trace Enabled" server configuration option is set to "1".	OK	
Hide Instance	Checks if the instance is hidden (not exposed by SQL Browser).	OK	WARNING

Job configuration

This chapter provides details on jobs responsible for gathering Security and Compliance statistics. Certain jobs have
the capability to modify (enable/disable/enforce) instance or database settings based on configured

parameters.

Auto change

This table presents the enable/disable/enforce functionality settings for some Security and Compliance jobs. When
configured, the job has the capability to automatically modify instance or database configurations to ensure
compliance. For jobs where automatic configuration changes are not feasible, both parameter names and values
remain empty.

4. Job configuration

This chapter provides details on jobs responsible for gathering Security and Compliance statistics. Certain jobs have the capability to modify (enable/disable/enforce) instance or database settings based on configured parameters.

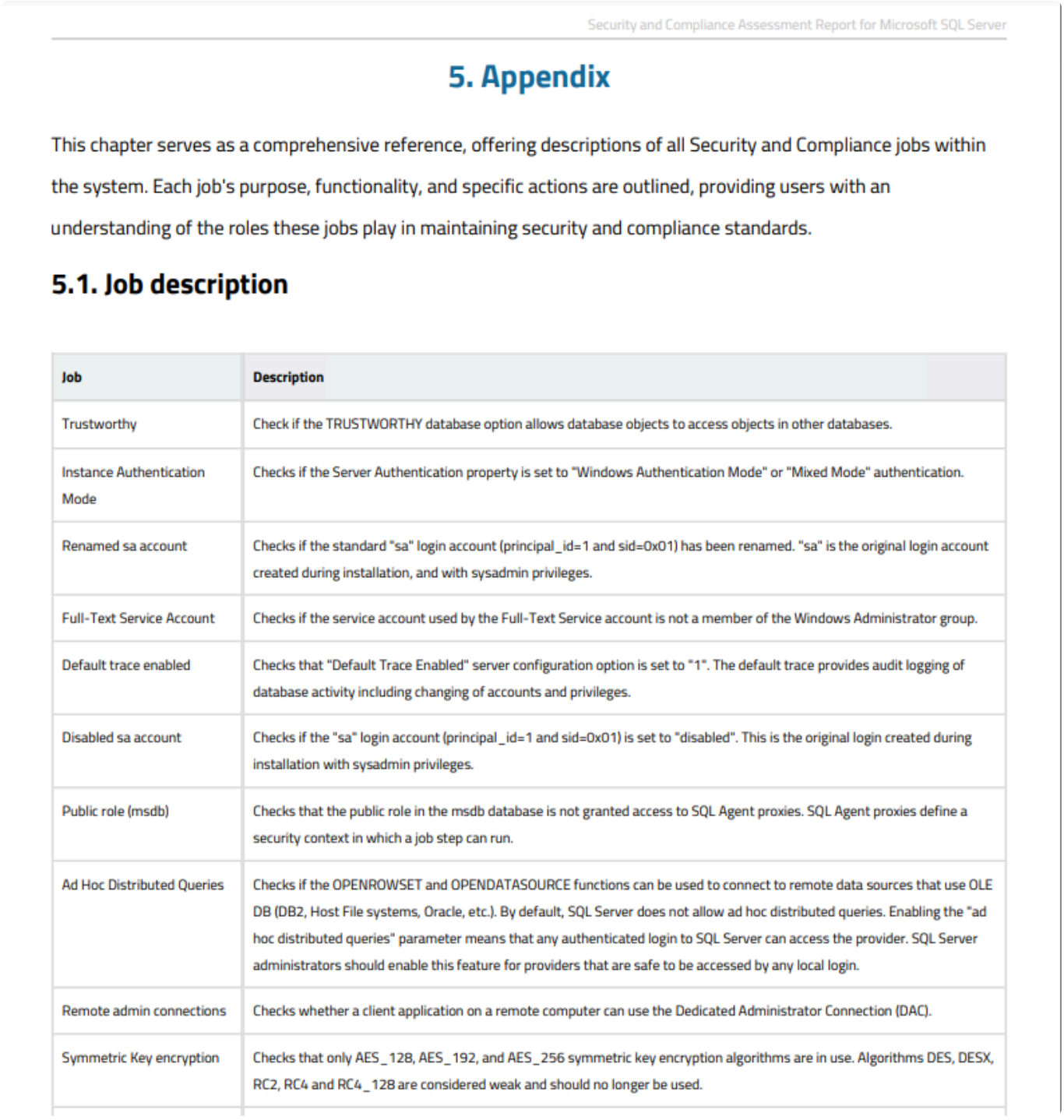
4.1. Auto change

This table presents the enable/disable/enforce functionality settings for some Security and Compliance jobs. When configured, the job has the capability to automatically modify instance or database configurations to ensure compliance. For jobs where automatic configuration changes are not feasible, both parameter names and values remain empty.

Job	Status	Last Run	Parameter	Auto change
Hide Instance	OK	15.jan.2024 01:13:45	enable HideInstance	NO
CLR Enabled	OK	15.jan.2024 01:04:45	disable CLR user code execution	NO
Instance Authentication Mode	OK	15.jan.2024 01:14:45		
Password policy	OK	15.jan.2024 01:21:45	enable check policy	NO
Login failed and successful setting	OK	15.jan.2024 01:36:45		
Command shell setting	OK	15.jan.2024 01:05:45	disable cmd shell setting	NO
Asymmetric Key size	OK	15.jan.2024 01:01:46		
Security and Compliance framework	OK	15.jan.2024 02:00:46		
MS SQL service account	OK	15.jan.2024 01:17:49		
Public server role	OK	15.jan.2024 01:22:45	revoke extra permission	NO
MS SQL Server patch status	OK	15.jan.2024 01:16:45		
Ole Automation Procedures	OK	15.jan.2024 01:18:45	disable Ole Automation Procedures	NO
Cross DB Ownership Chaining	OK	15.jan.2024 01:06:45	disable cross db ownership chaining	NO

Appendix

This chapter serves as a comprehensive reference, offering descriptions of all Security and Compliance jobs within
the system. Each job's purpose, functionality, and specific actions are outlined, providing users with an understanding of the roles these jobs play in maintaining security and compliance standards.



Security and Compliance Assessment Report for Microsoft SQL Server

5. Appendix

This chapter serves as a comprehensive reference, offering descriptions of all Security and Compliance jobs within the system. Each job's purpose, functionality, and specific actions are outlined, providing users with an understanding of the roles these jobs play in maintaining security and compliance standards.

5.1. Job description

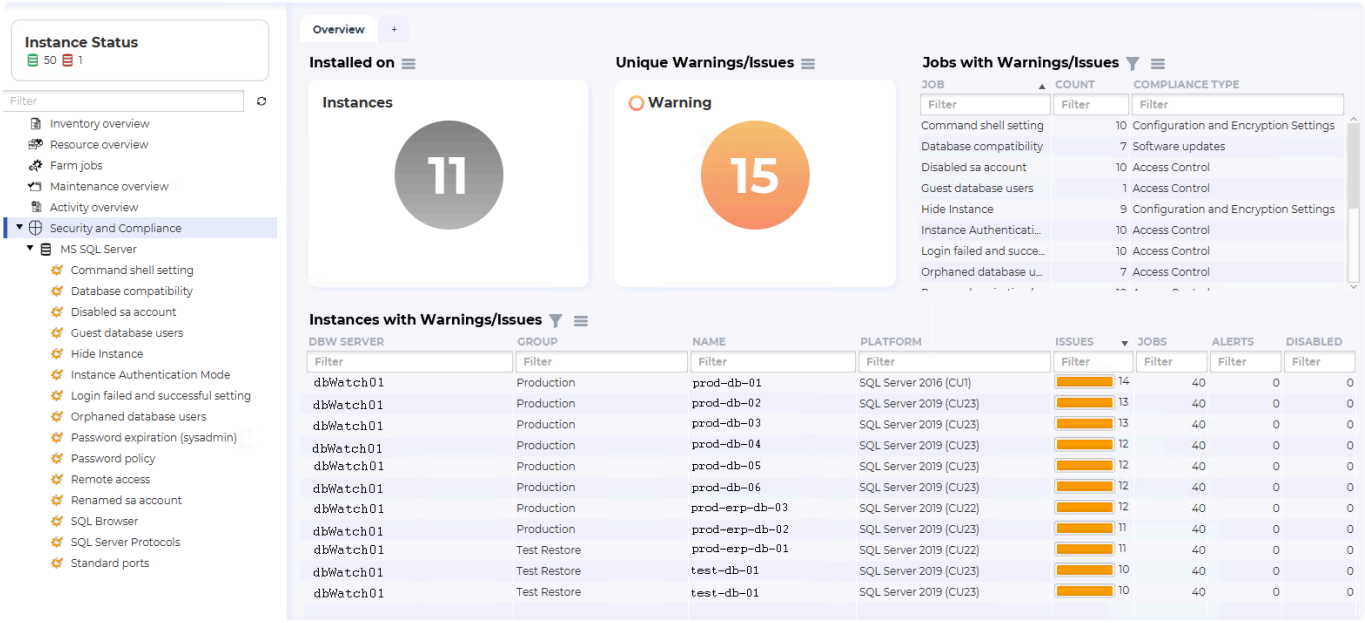
Job	Description
Trustworthy	Check if the TRUSTWORTHY database option allows database objects to access objects in other databases.
Instance Authentication Mode	Checks if the Server Authentication property is set to "Windows Authentication Mode" or "Mixed Mode" authentication.
Renamed sa account	Checks if the standard "sa" login account (principal_id=1 and sid=0x01) has been renamed. "sa" is the original login account created during installation, and with sysadmin privileges.
Full-Text Service Account	Checks if the service account used by the Full-Text Service account is not a member of the Windows Administrator group.
Default trace enabled	Checks that "Default Trace Enabled" server configuration option is set to "1". The default trace provides audit logging of database activity including changing of accounts and privileges.
Disabled sa account	Checks if the "sa" login account (principal_id=1 and sid=0x01) is set to "disabled". This is the original login created during installation with sysadmin privileges.
Public role (msdb)	Checks that the public role in the msdb database is not granted access to SQL Agent proxies. SQL Agent proxies define a security context in which a job step can run.
Ad Hoc Distributed Queries	Checks if the OPENROWSET and OPENDATASOURCE functions can be used to connect to remote data sources that use OLE DB (DB2, Host File systems, Oracle, etc.). By default, SQL Server does not allow ad hoc distributed queries. Enabling the "ad hoc distributed queries" parameter means that any authenticated login to SQL Server can access the provider. SQL Server administrators should enable this feature for providers that are safe to be accessed by any local login.
Remote admin connections	Checks whether a client application on a remote computer can use the Dedicated Administrator Connection (DAC).
Symmetric Key encryption	Checks that only AES_128, AES_192, and AES_256 symmetric key encryption algorithms are in use. Algorithms DES, DESX, RC2, RC4 and RC4_128 are considered weak and should no longer be used.

Here is a sample report from the Security and Compliance package.

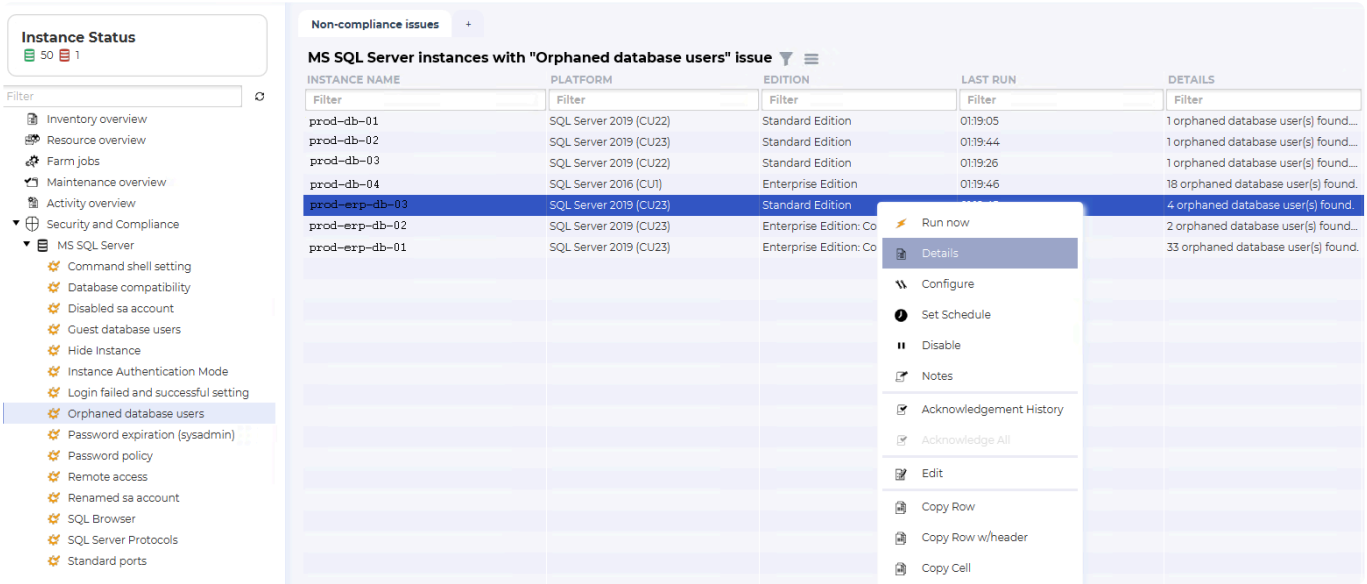
Dashboards / Farm views

The dashboard provides a comprehensive overview of where the 'Security and Compliance' module is

installed. It details the number and types of ‘errors’ relative to ‘Microsoft best practices’ and more. In summary, it features:
The number of instances where the module is installed.
The total count of unique ‘errors’.
Details of ‘errors’ and how many instances are affected by each. The number of ‘errors’ per instance.

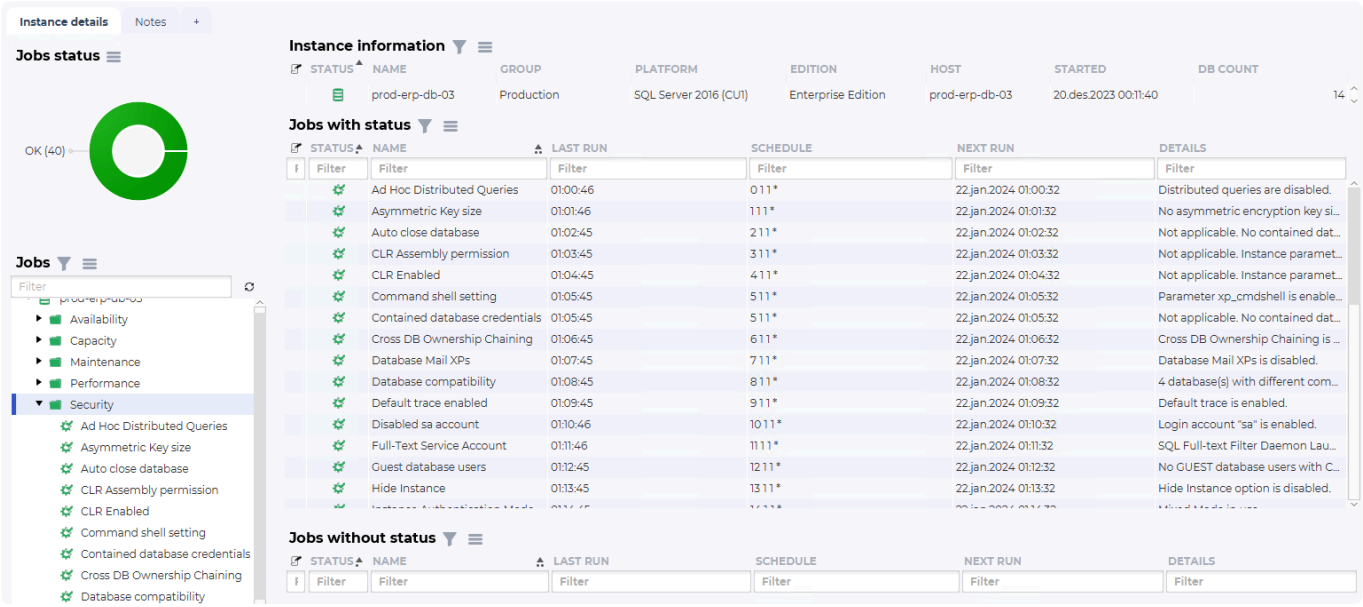


On each database instance, you can then drill down to a specific job to investigate what issues are present.

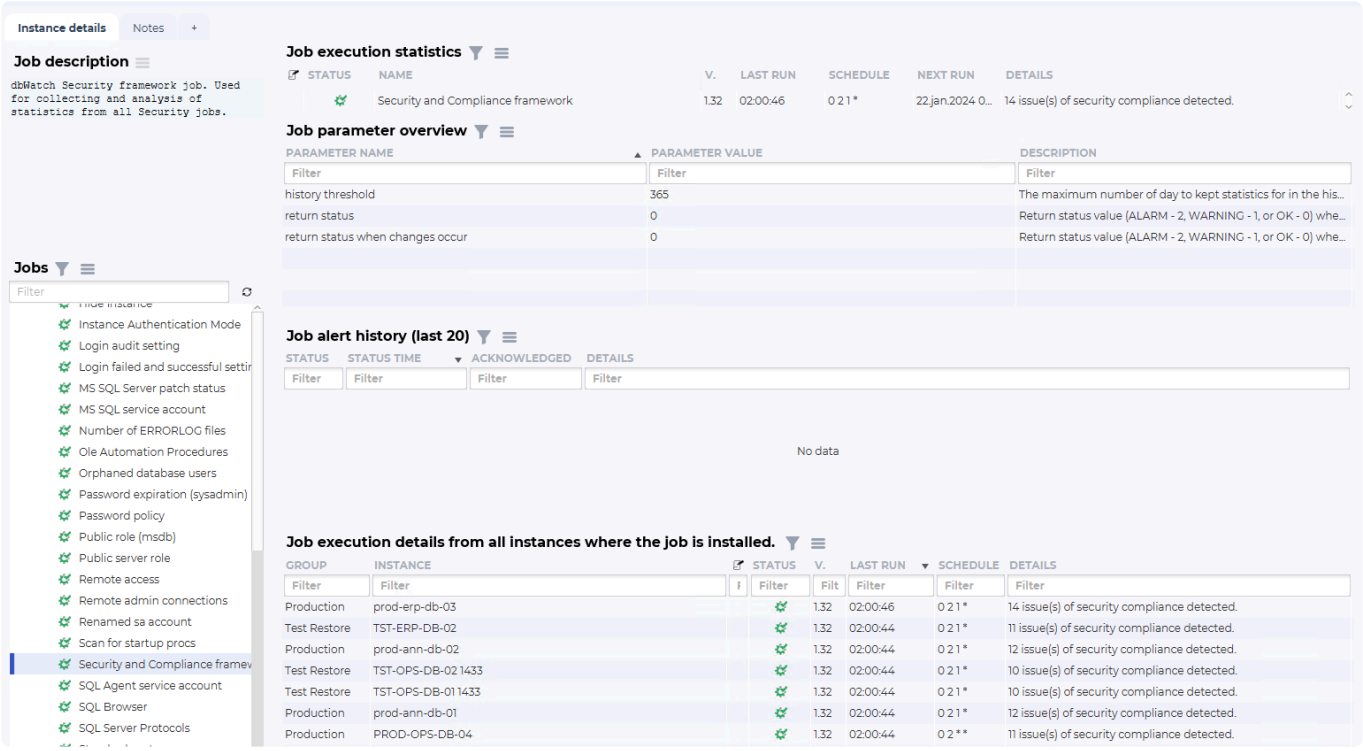


Monitoring

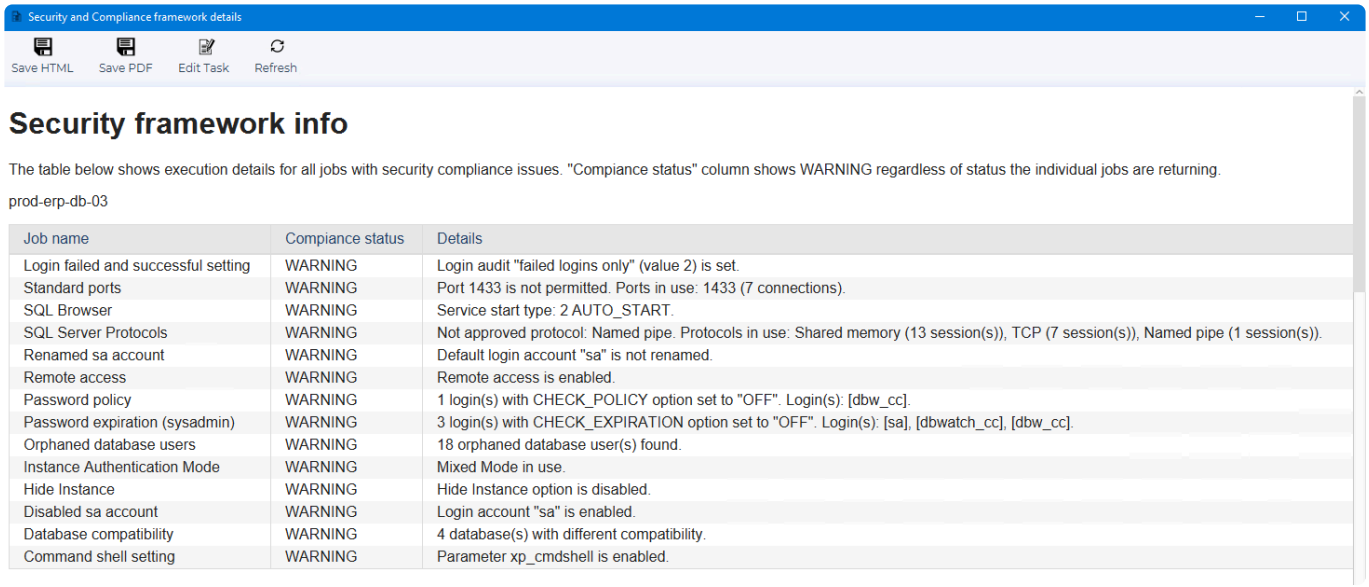
In the monitoring interface, you can see the overall status for the database instance, as well as all the monitoring jobs installed, their status, scheduled time, and short details. Each of these will have a built-in report for that particular metric.



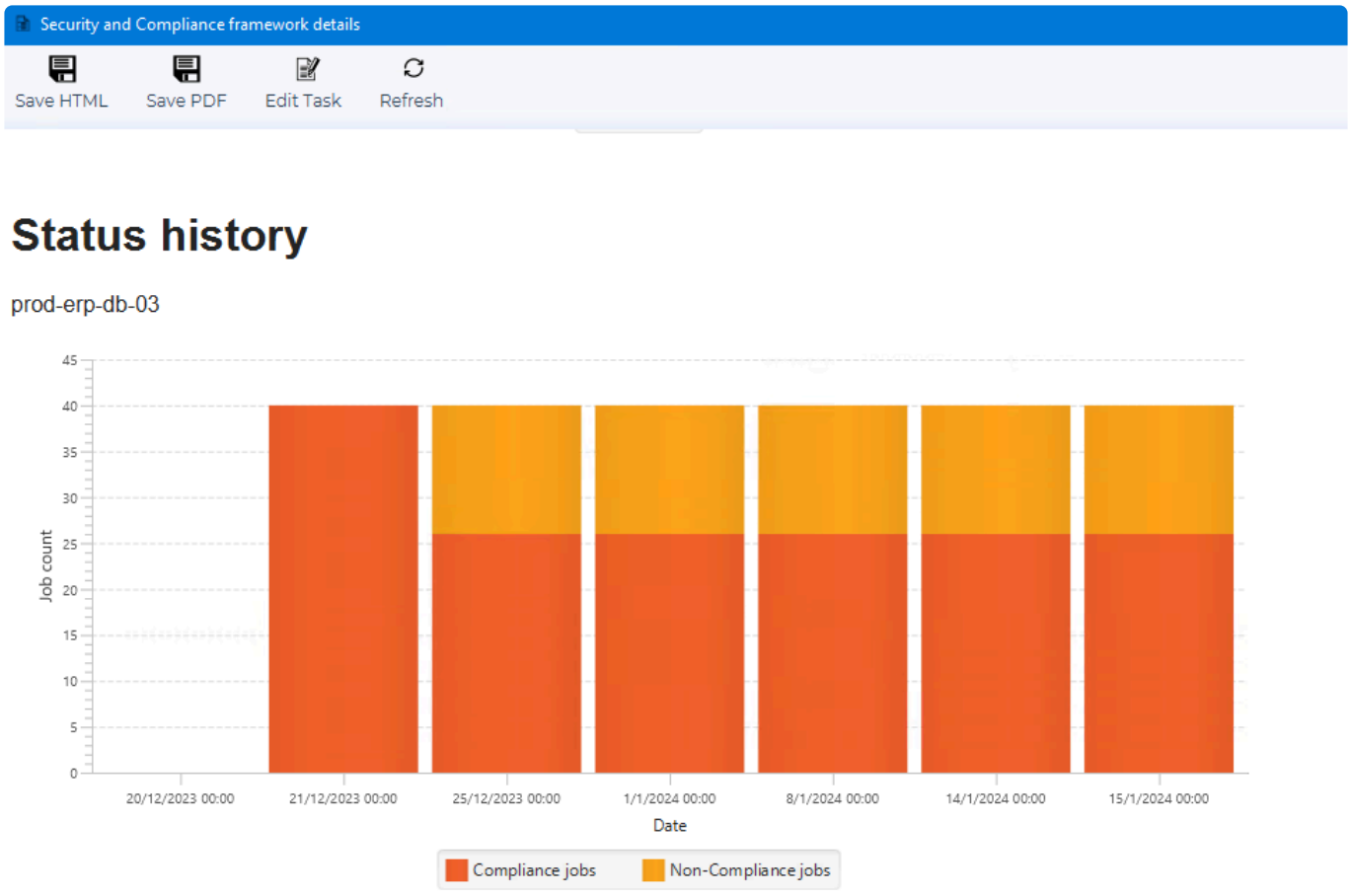
There is a job (Security and Compliance framework) that is exclusively used for aggregating and analyzing data from the remaining jobs. Among other functions, it can alert you to any changes in the 'errors' occurring within the instance.



A report (right-click on the job and select report) for this job has details on that status for all the Security and Compliance jobs for that instance and its status will typically look like this:

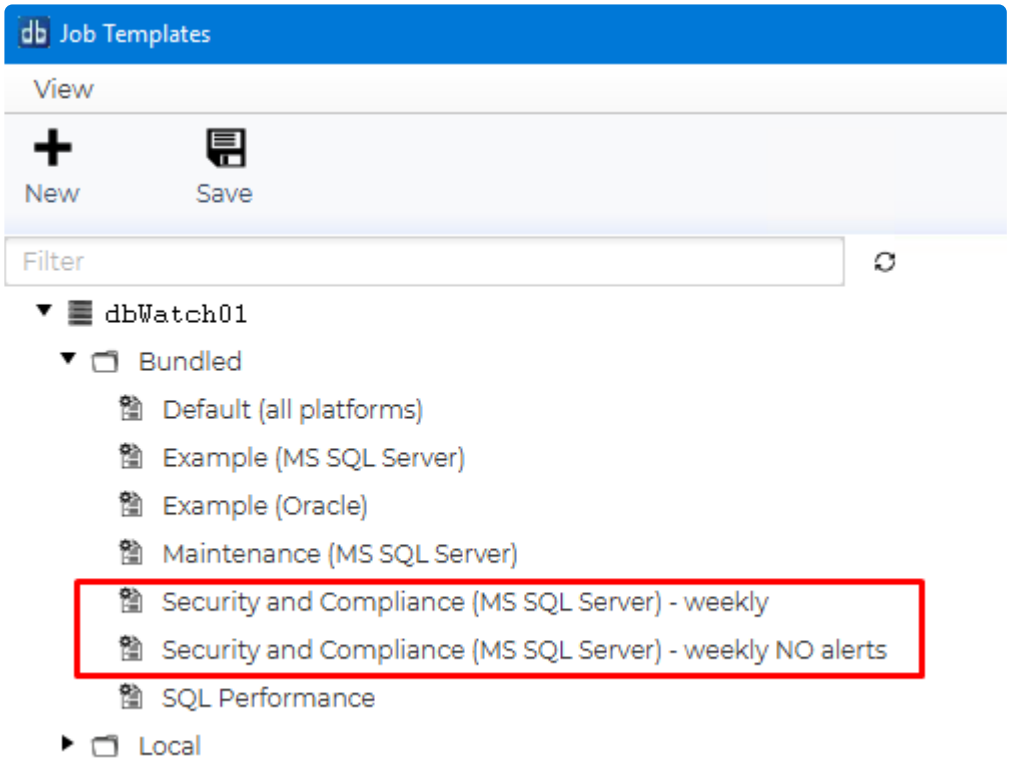


It also keeps track of changes in compliance warning over time, so you can see if your hard work improves the security of your database instance.



Templates

There are two templates that simplify the installation of the “Security and Compliance” module. One of them configures all jobs so that they don’t trigger alerts. Even without sending alerts, their actual status remains visible in Farm-views.

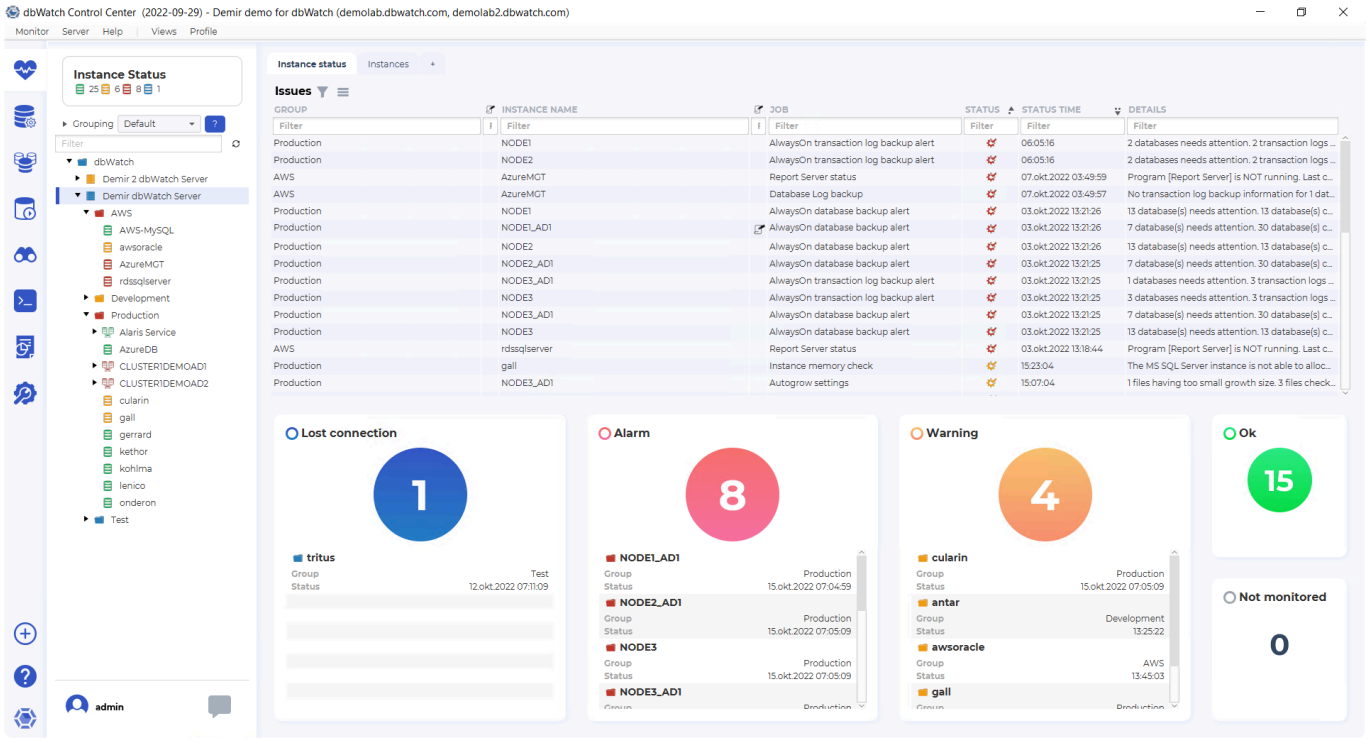


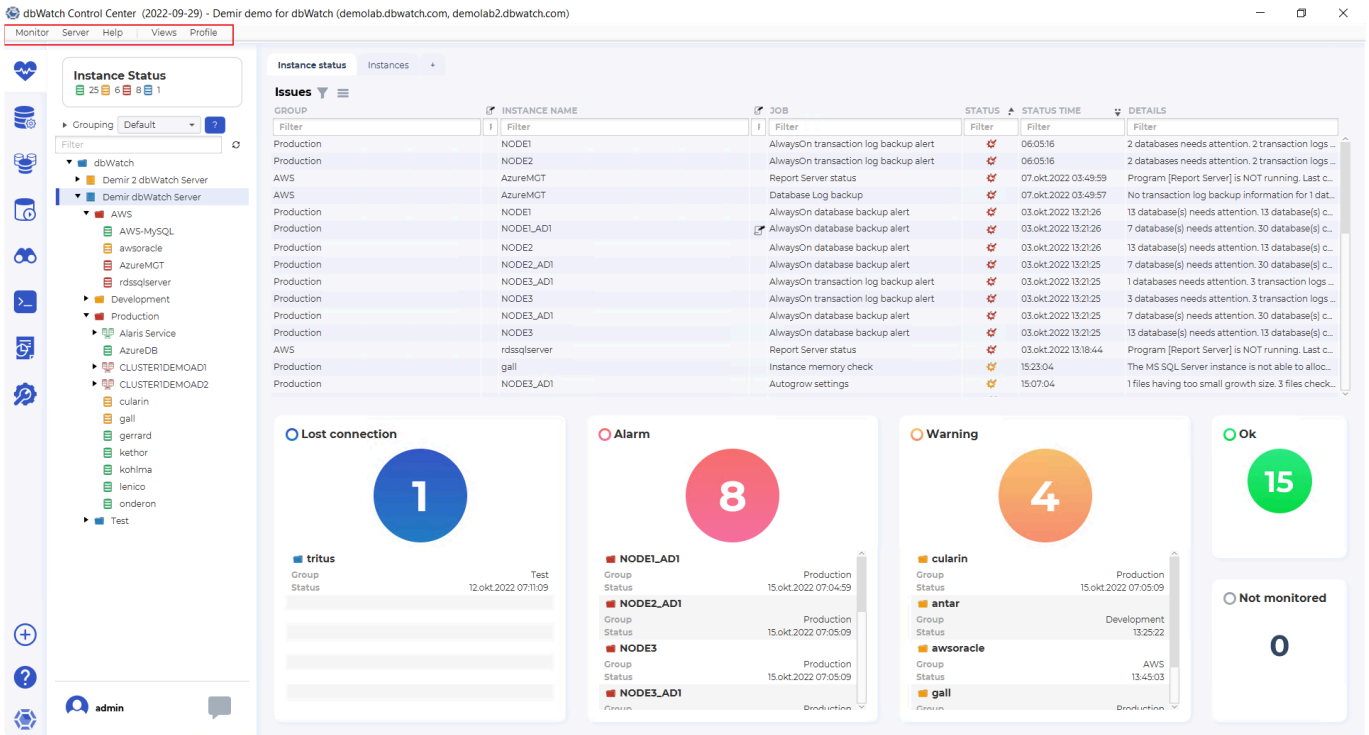
[<< Installation of the Security and Compliance package] / [Using Control Center >>]

Using Control Center

In this section, we will cover the user interface, the different modules, and how they are used.

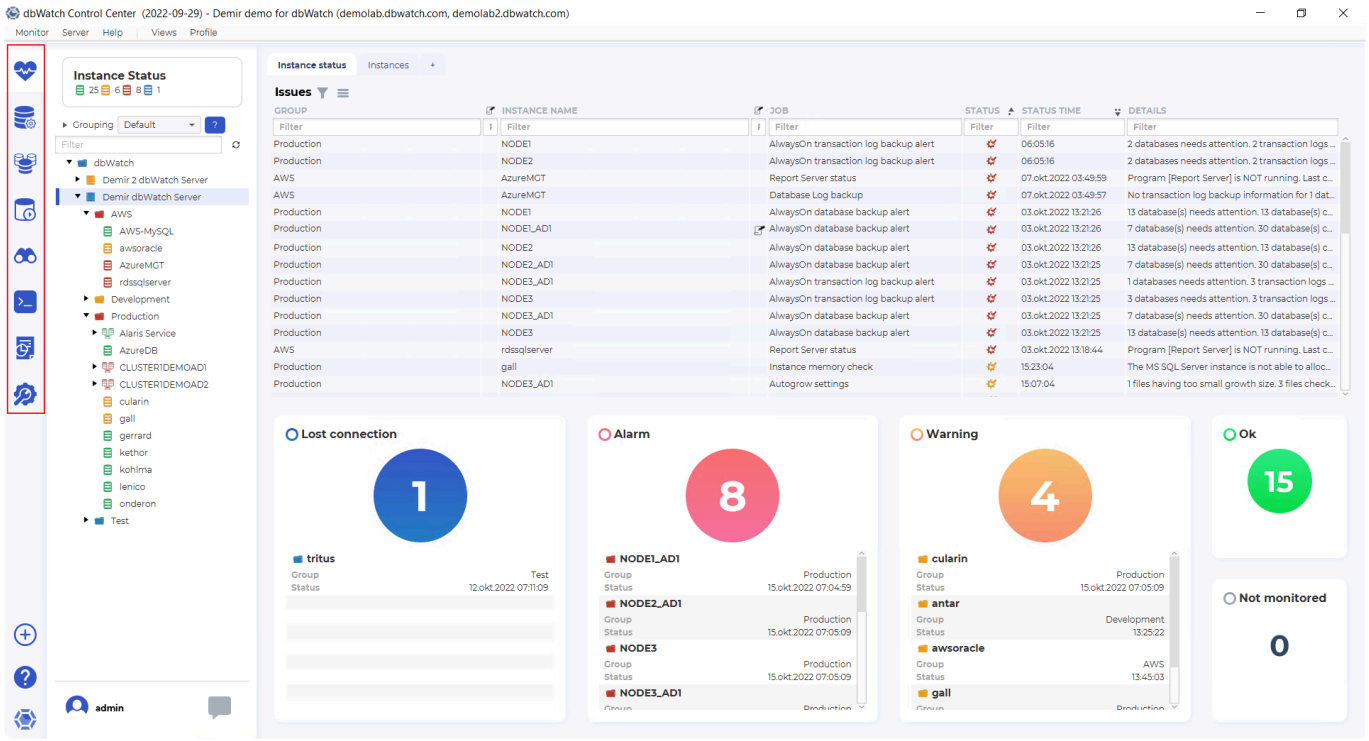
When we start dbWatch Control Center we will by default start-up in the monitoring view:



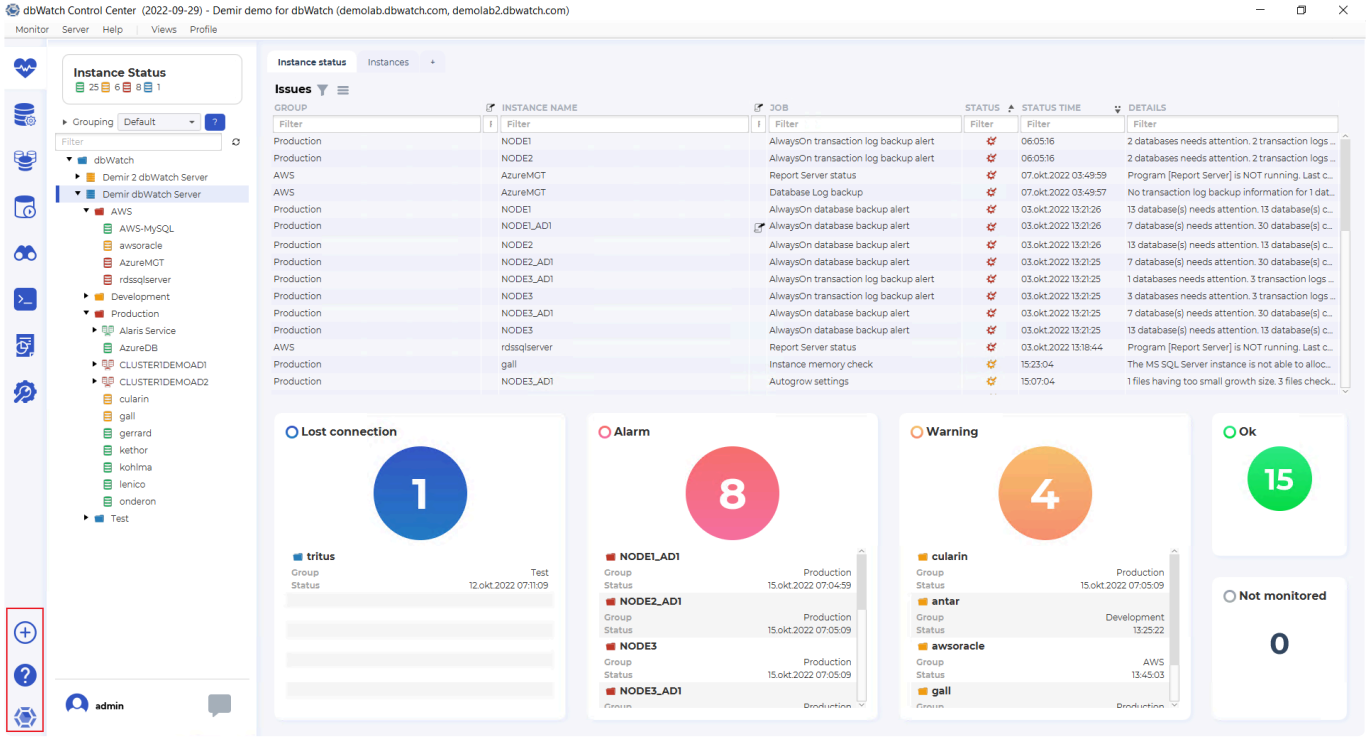


We have the top menu, where we find the configuration of client connections and server settings. Details in this section

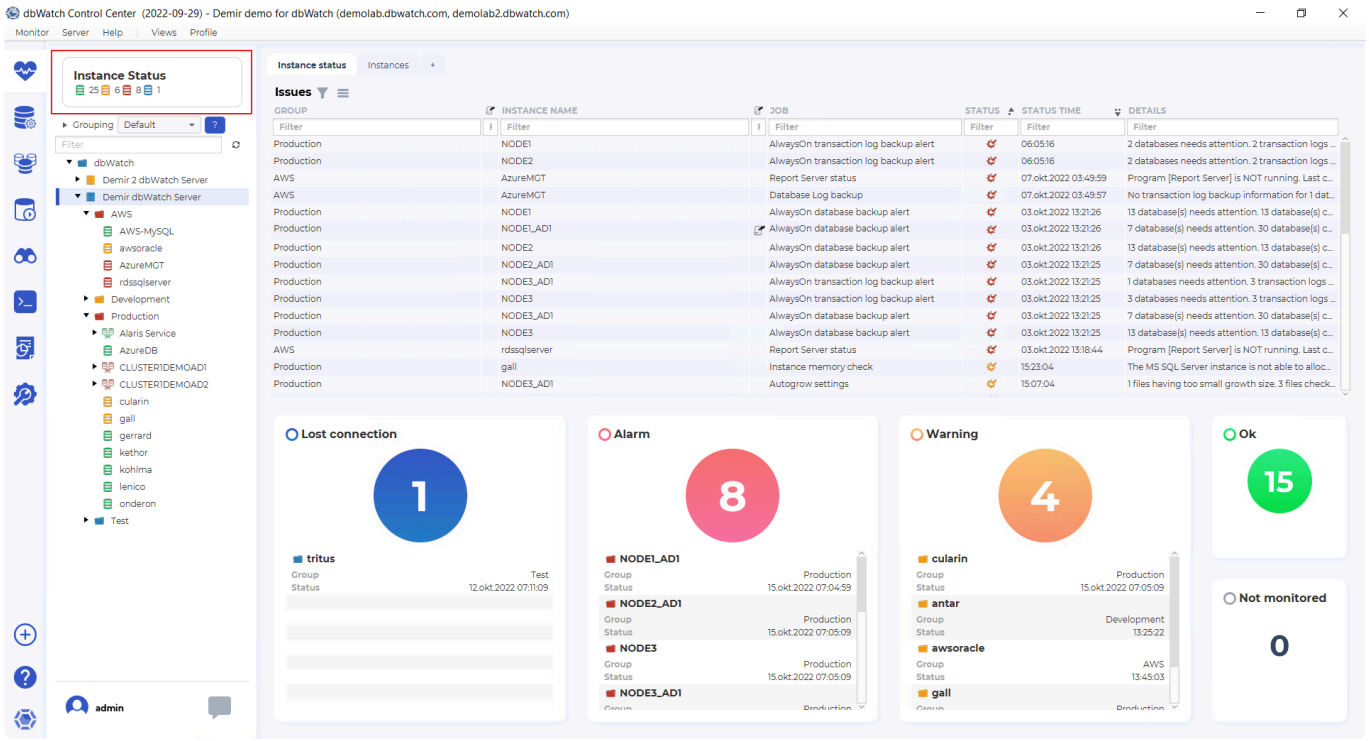
It is possible to turn off and on modules, check out Disabling and Enabling Views



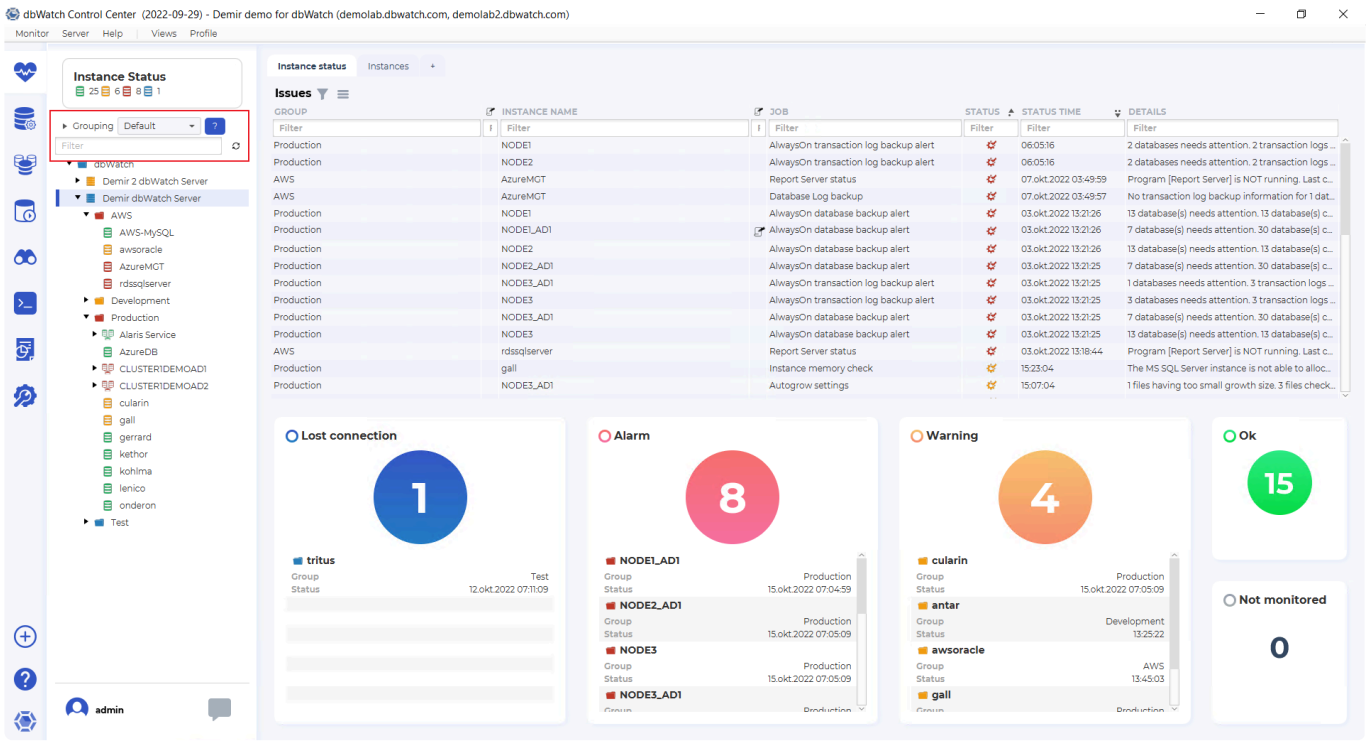
On the left side, we have the different [modules of Control Center](#).



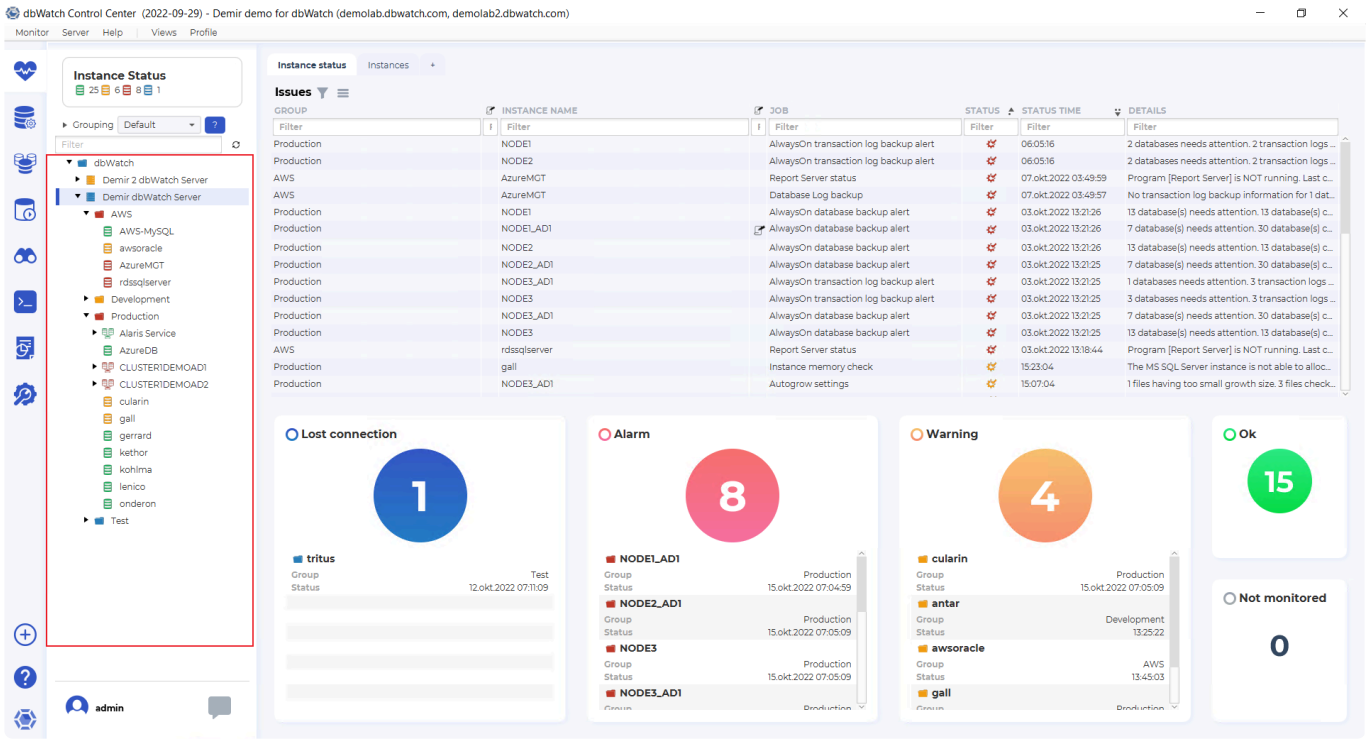
Underneath the modules, we have the menu for [adding new instances](#), the question mark button that opens this wiki, and the dbWatch button opening the “About dbWatch” view with version information.



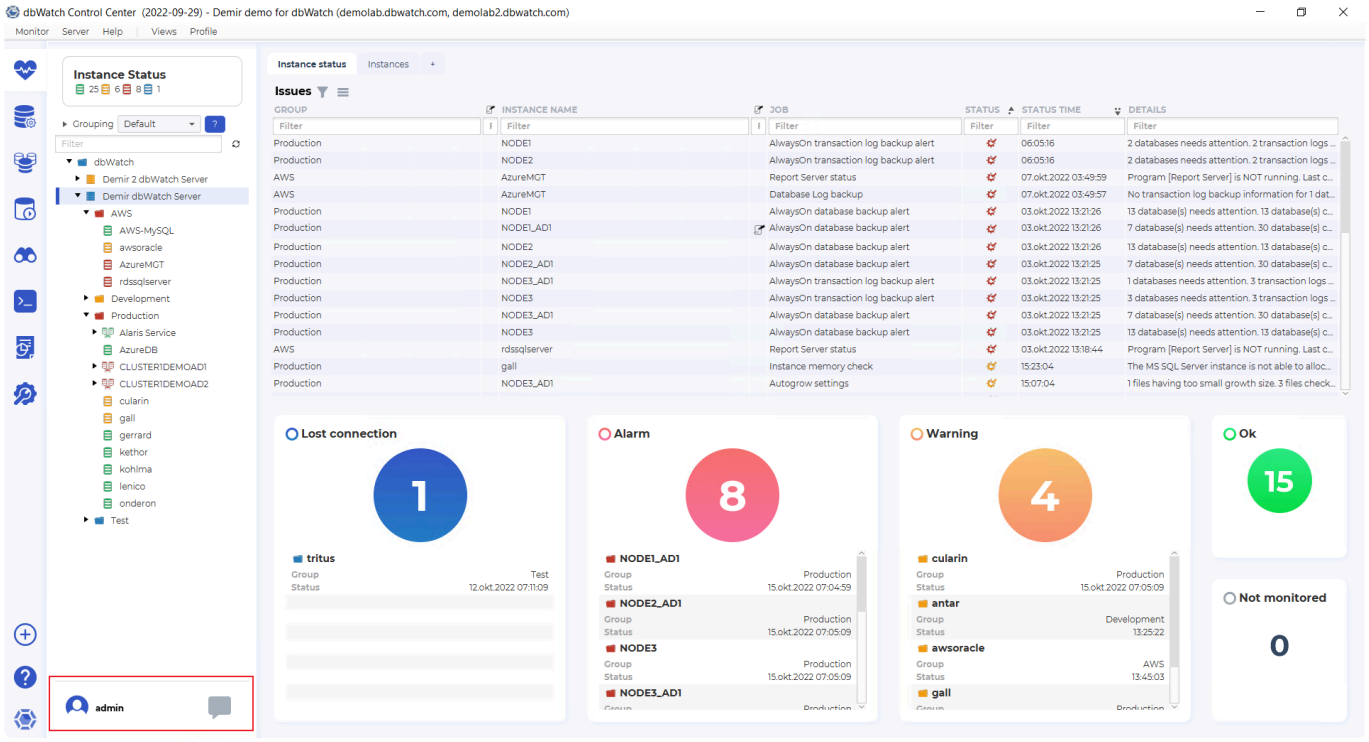
On the left top, next to the monitoring module menu, we have the instance status. Status colors indicate the number of instances in each status, green for ok, yellow for warning, red for alarm, and blue for instances where we have lost connection.



Underneath the instance status, we have the grouping and filter section that allows the user to group and filter the instance tree underneath. We go into more details on this here.



Underneath the grouping and filter section, we find the tree structure that allows selecting instances, to open the jobs overview for this instance.



Lastly, underneath the tree structure, we have the icon that shows the username logged in, and next to that a chat icon, that can show new messages, and be used to chat with other DBA's connected to Control Center.

For more information on the monitoring module go here.

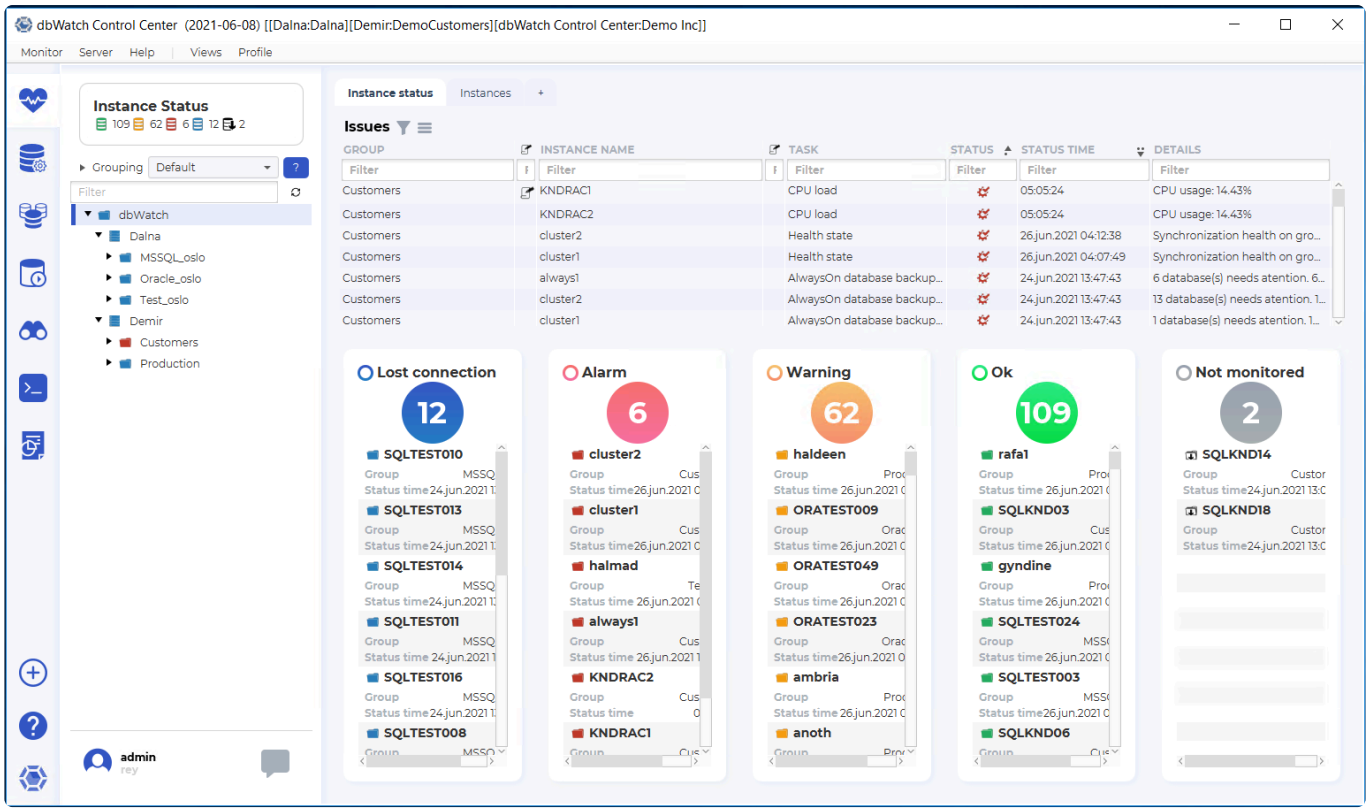
<< Security and Compliance package on SQL Server / The modules of Control Center >>

The modules of Control Center

We have seven default modules available when you start dbWatch Control Center. This article will only introduce you to what these instances are and give a brief description of them. You need to be familiar with the navigation tree to better understand how these modules work.

Monitoring Module

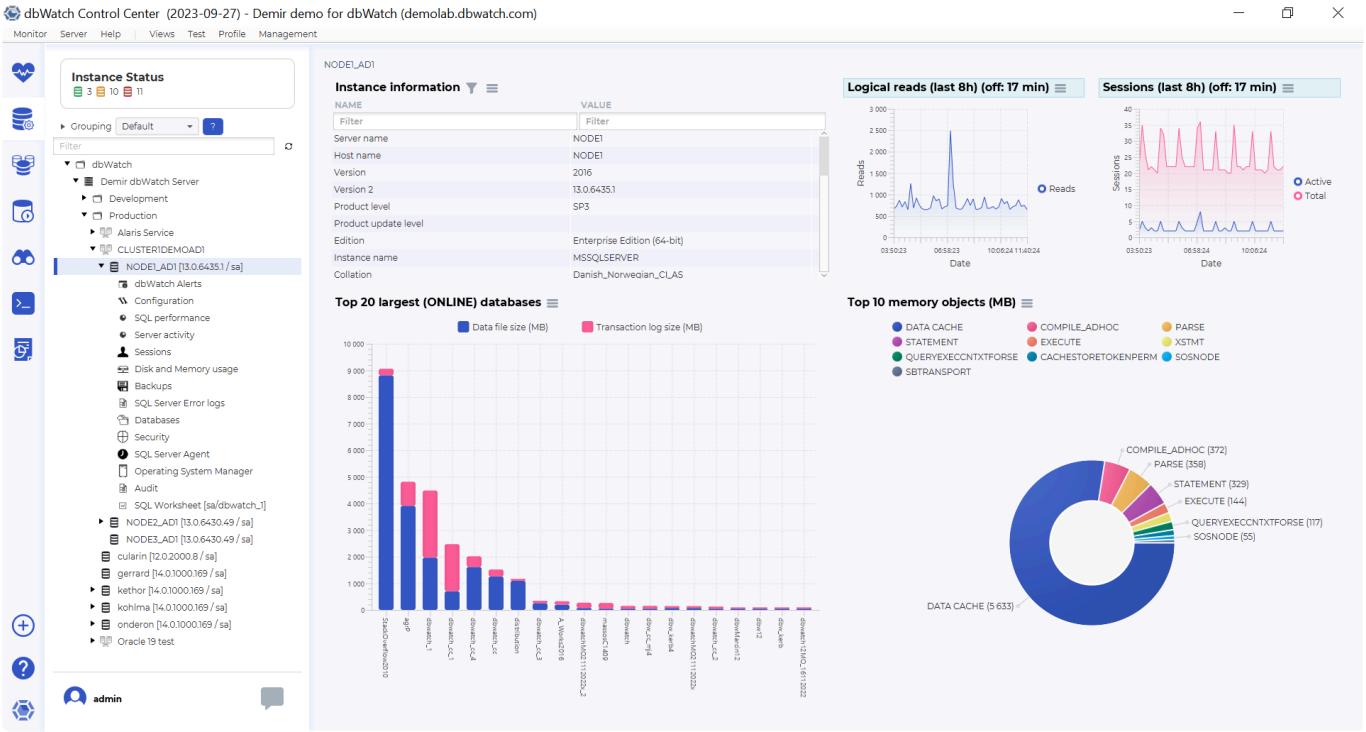
In this module, you can monitor all your registered instances. When you select a server group, it will display all the instances under that server group. For example, you wanted to view a server called Dalna in group MSSQL_oslo. You only need to select the MSSQL_oslo group represented by a folder icon.



In this example, we are currently seeing the dbWatch Server with different dbWatch servers found under it. We are currently in the Instance status tab, which shows the count of instances with different statuses and glaring issues found in all of our instances.

Management Module

The management module is where you can directly deploy changes to your databases. If you click on an instance, Control Center will display options for you. You can check the performance, sessions, databases, error logs, audit logs, and memory. You are also given the option to configure the database's current configuration and access control. Plus, you can set up backups and jobs.



In this example, we are given an overview of the server we are currently looking at.

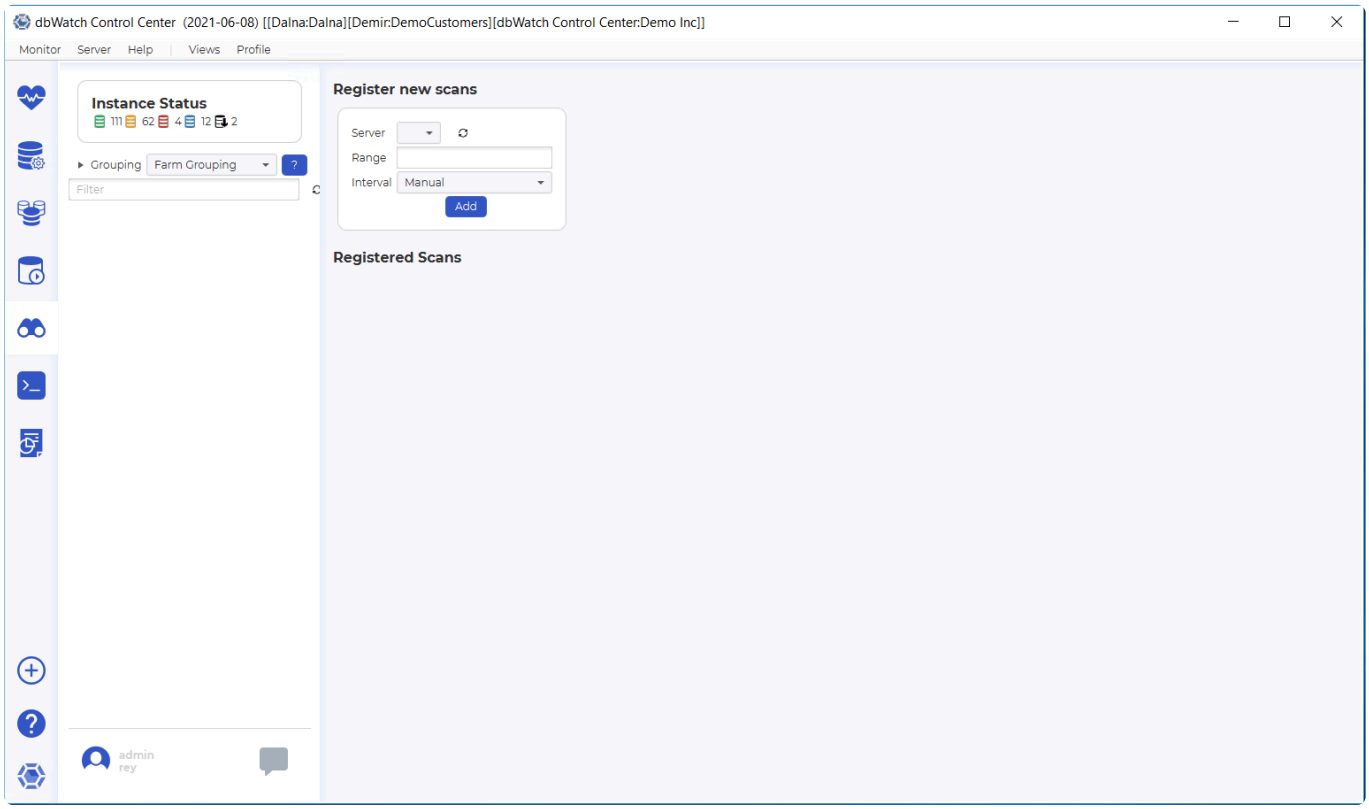
Farm Module

The farm module is an expanded version of the monitoring module and management module. In essence, you have a more powerful tool to monitor and manage your database farm.

Currently, we are looking at the job status on our entire database farm. If you want to learn more about Farm Management, you can check out our library on [database farm management]

Auto-discover Module

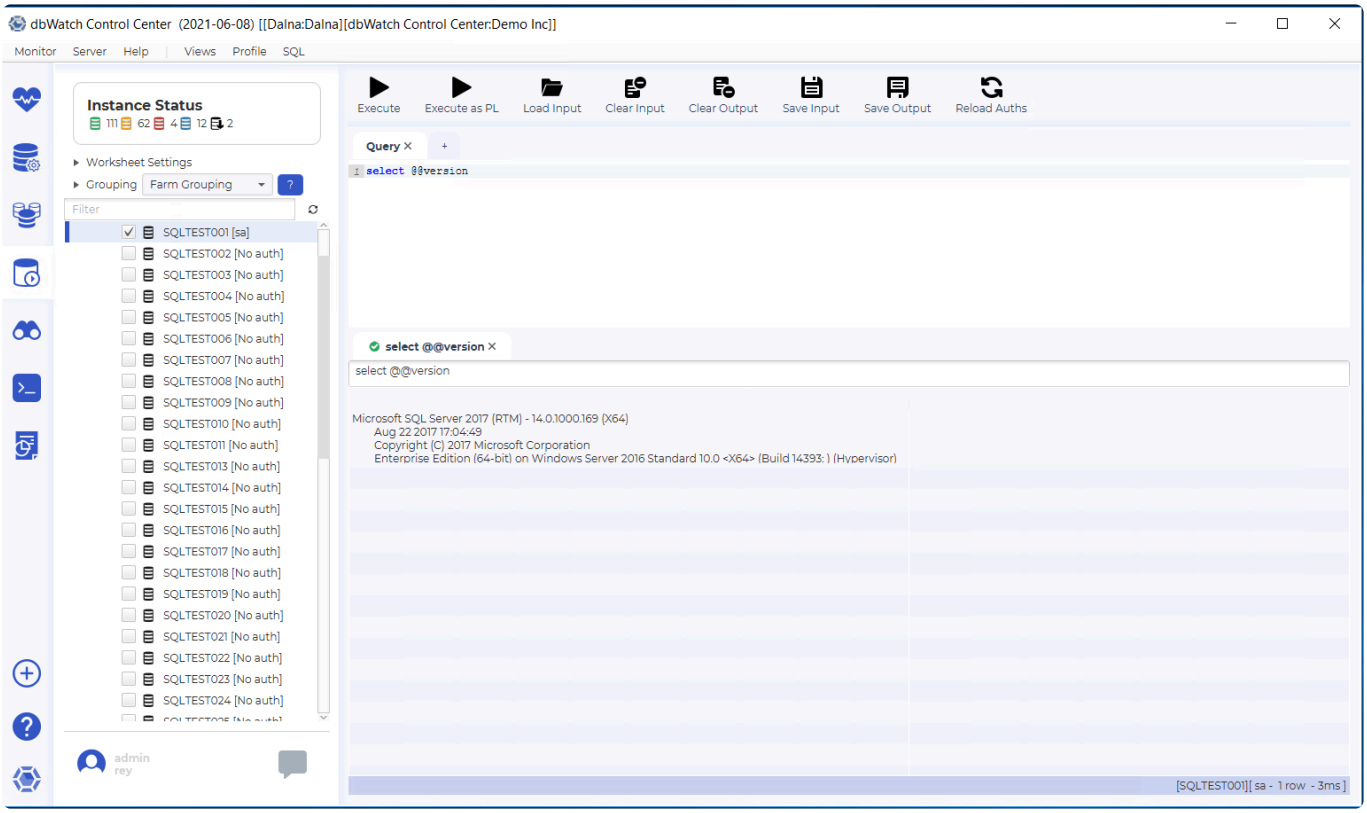
This module scans a specified server for active database instances. Afterward, it will list down all database servers found.



You can learn more about auto-discover by clicking on the link provided.

Worksheet Module

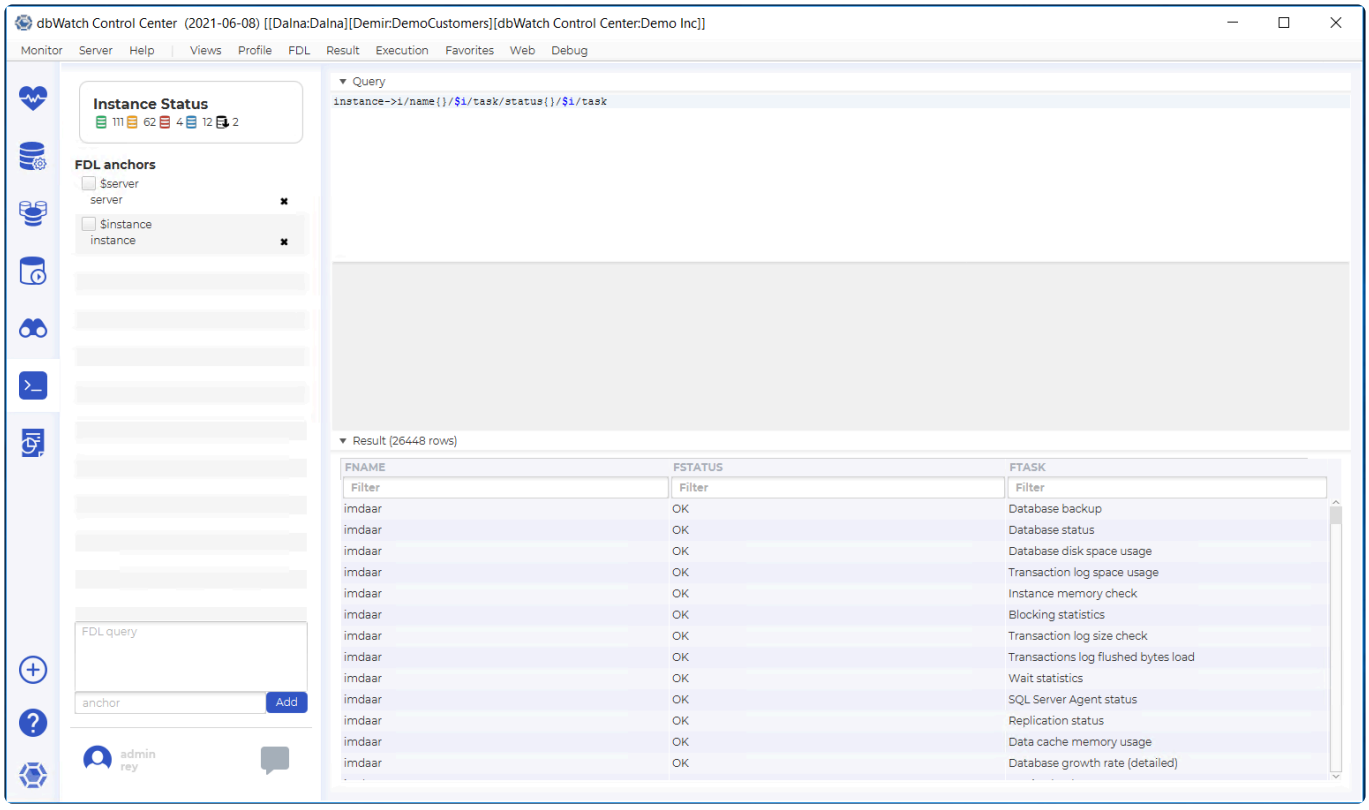
You can script queries under this worksheet. You can cross-check them manually under one window.



The worksheet follows the native query language of your database. To learn more about this, you can check SQL Worksheets.

FDL Console Module

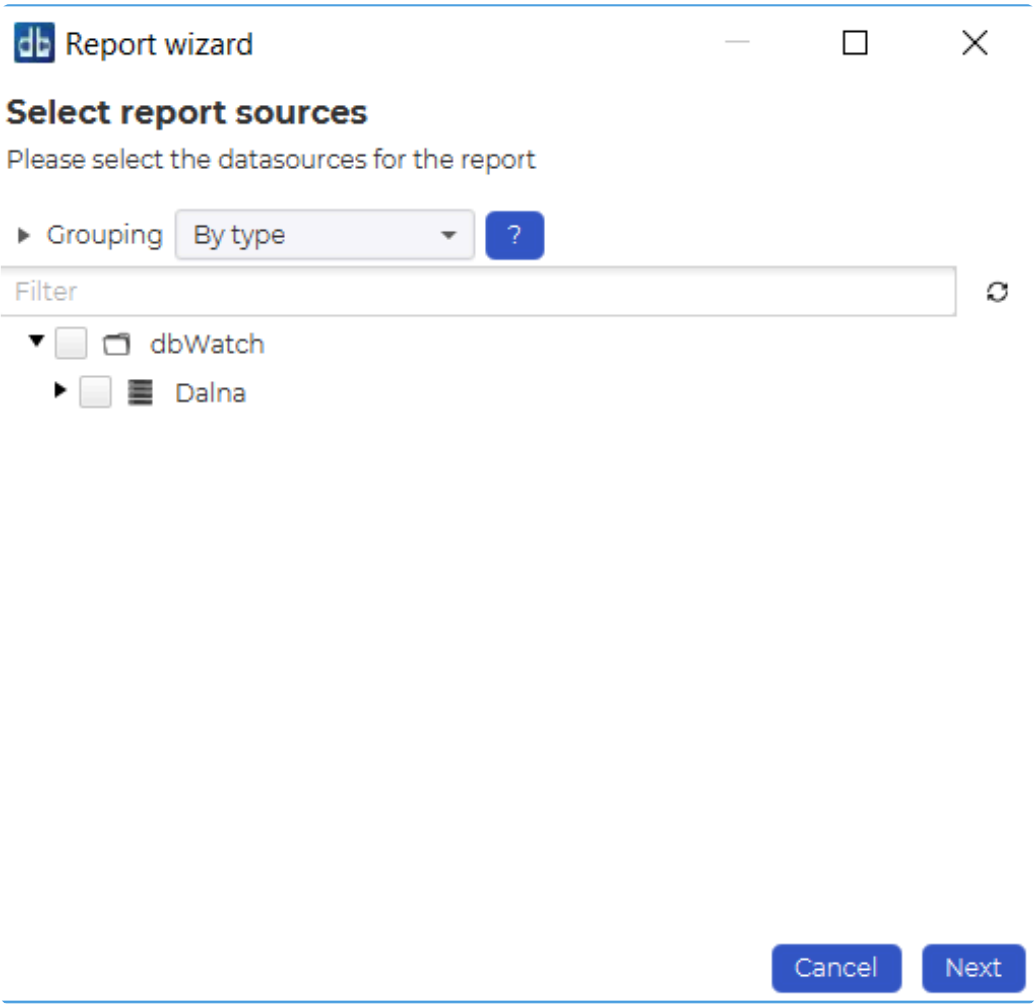
You can script FDL queries in this module. This works well for database farms and it's not applicable for single instances.



In the example above, we are checking the jobs and status for each instance. Check out the section of FDL to learn more.

Report Module

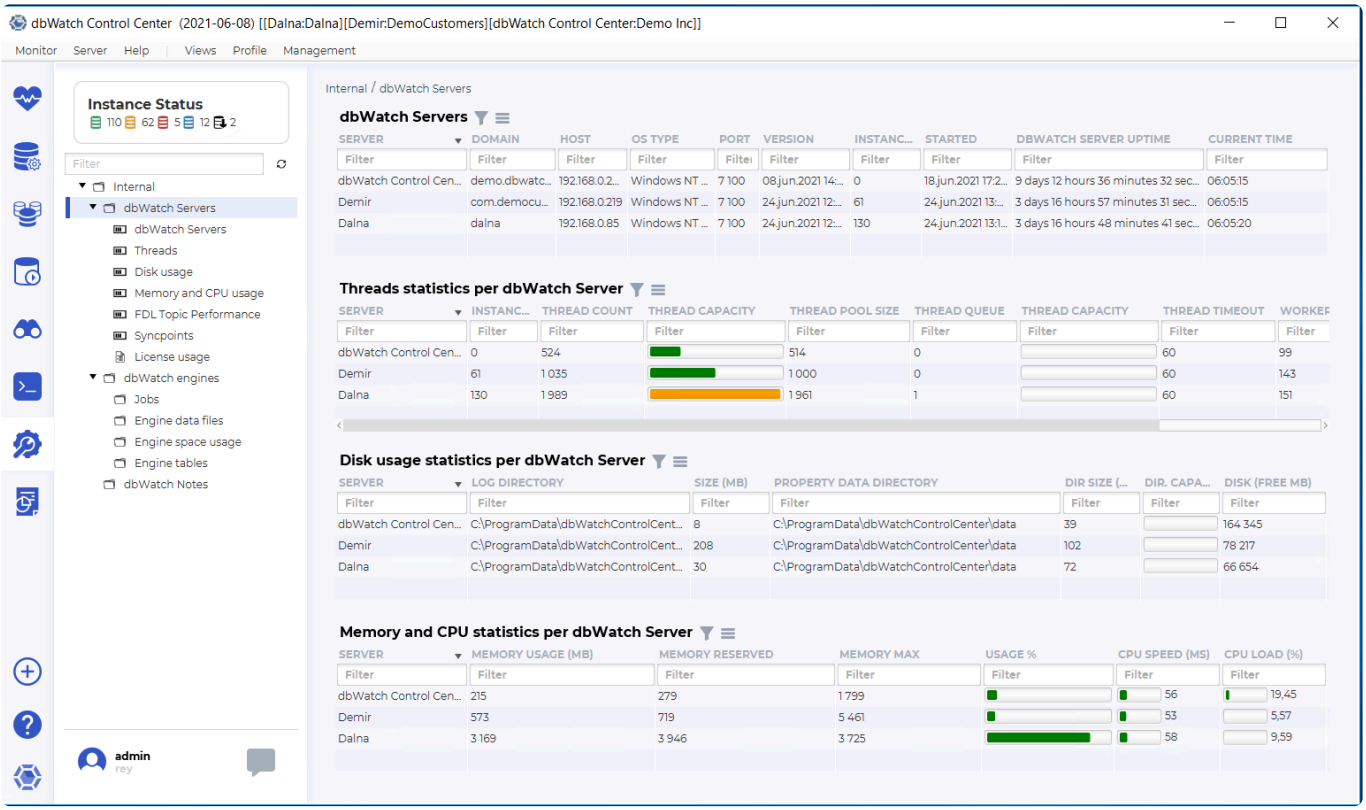
When you click on the report module, a report wizard will appear and guide you in generating the report.



You can learn more about reports in our section on Reporting

dbWatch Internal

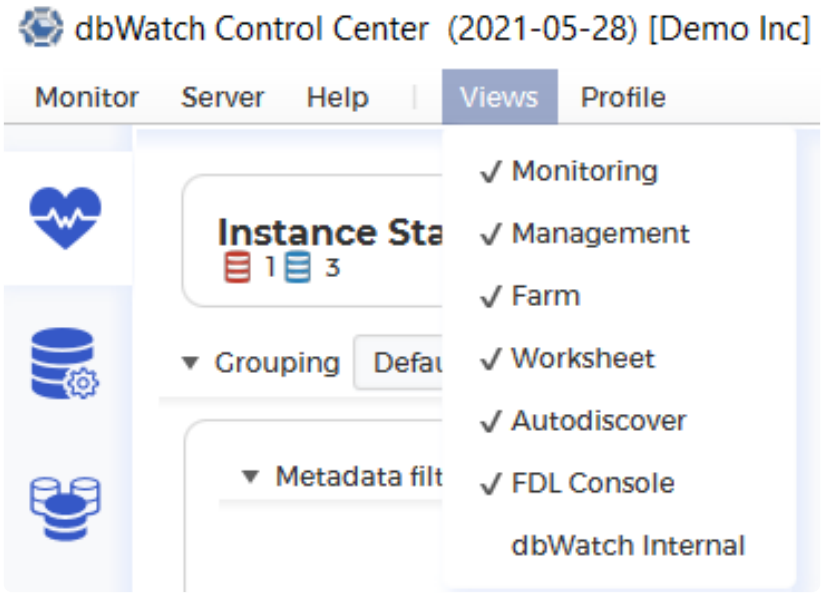
Finally, we have a special module that tracks the information of your dbWatch server and other dbWatch servers currently controlled by your dbWatch monitor.



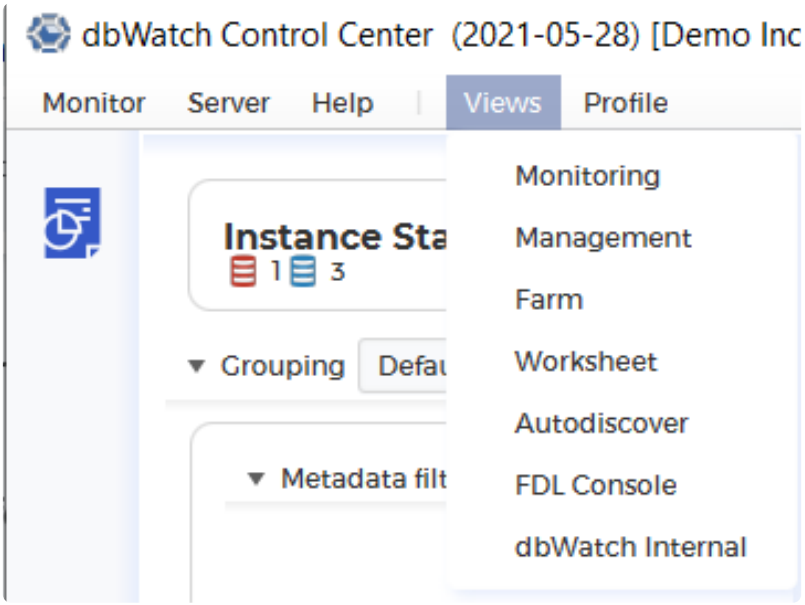
<< Using Control Center / Disabling and Enabling Views >>

Disabling and Enabling Views

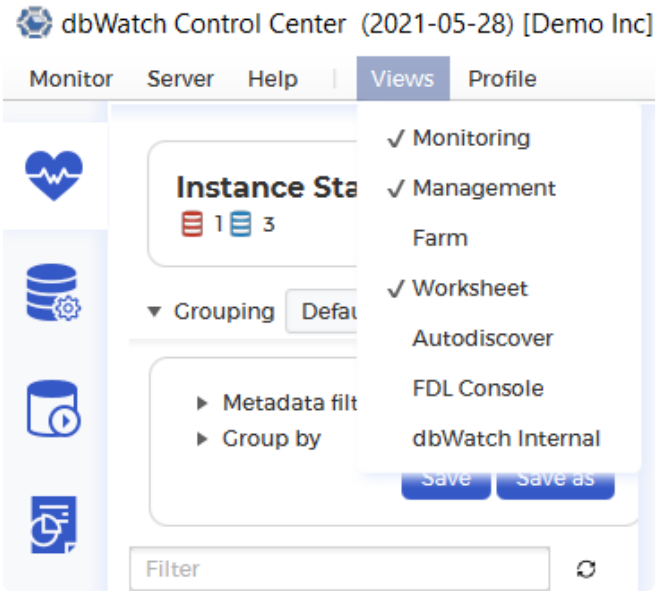
You can enable or disable modules from your dbWatch Control Center by unticking their check boxes. If you check Views, you will see the default set up of modules that are visible.



Once you disable all the modules, you'll only be left with the Report Module.



Otherwise, you can customize your dbWatch UI with the modules you prefer.



[<< The modules of Control Center](#) / [Instance management >>](#)

Instance management

In instance management we focus on the process of adding, removing and managing instances and instance groups.

You will need to add database instances to dbWatch Control Center in order to monitor, manage and create reports from your database instances.
There are 4 ways to start adding database instances to dbWatch.

1. Adding instances using the wizard

This is the most common way to get started. It is simple and works great for adding instances for testing and for scenarios where you are adding less than 50 database instances at once.

Wizard step by step for MS SQL Server
Wizard step by step for Oracle
Wizard step by step for PostgreSQL
Wizard step by step for MySQL

1. Adding instance from autodiscovery

dbWatch Control Center can scan networks, both once and periodically to detect database instances that are not already added. This is typically used on larger installations after initial installation to detect new database instances that you might not know of or that has been added without notifying the DBA's. Control Center will detect the database instances and pre-fill in details for the wizard process that is used to add the instances.

Using instance autodiscovery

1. Bulk import of instances

Bulk import of is used in larger deployments where a CSV formatted file with the details of database instances to be added to Control Center is processed in an automated fashion. This is a good method when adding more than 50 database instances. It is very quick once the CSV file is created, and database environments of around 200 database instances can be installed and configured in less than one hour.

Bulk adding instance

1. Using Control Center Commandline (CCC) scripts

The CCC is a commandline tool used to run scripts that interact with dbWatch Control Center or database instances that is added to it. This method is used to automate the process of adding database instances to Control Center, for example as part of automated deployment of database servers.

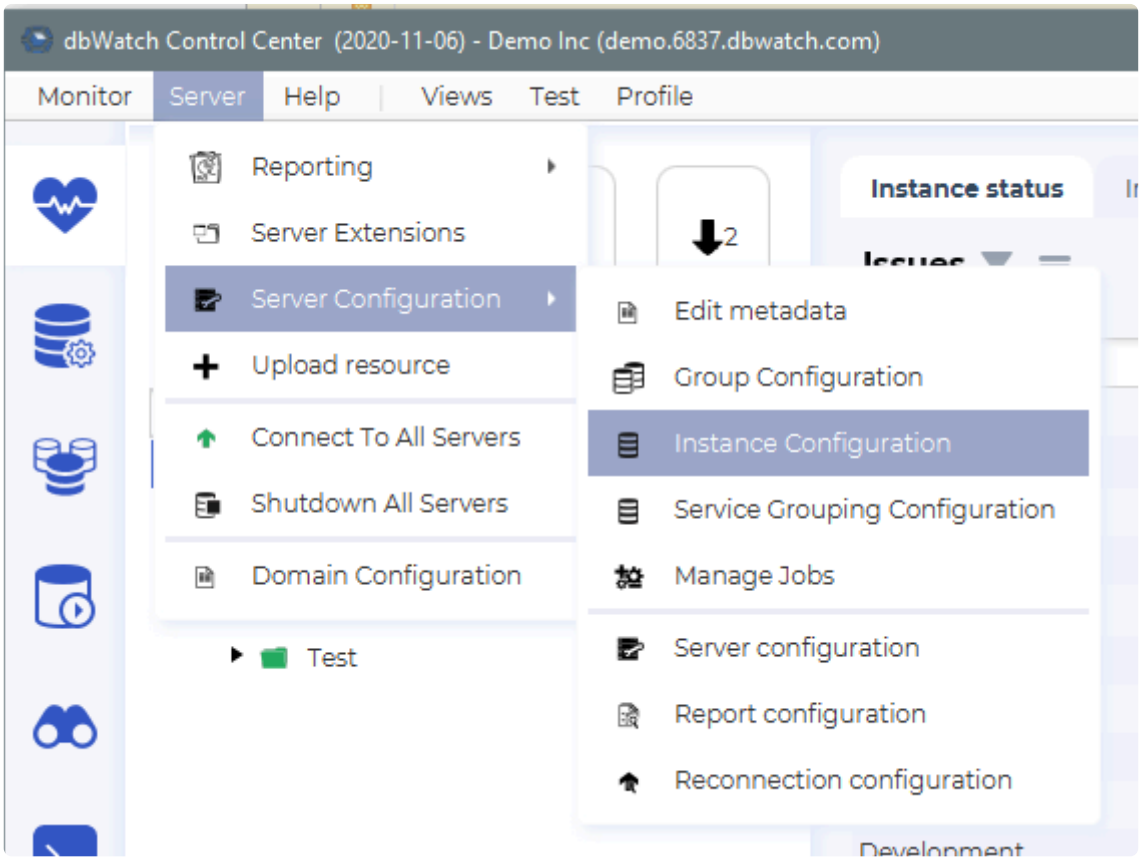
The Control Center Commandline

<< Disabling and Enabling Views / Instance Configuration >>

Instance Configuration

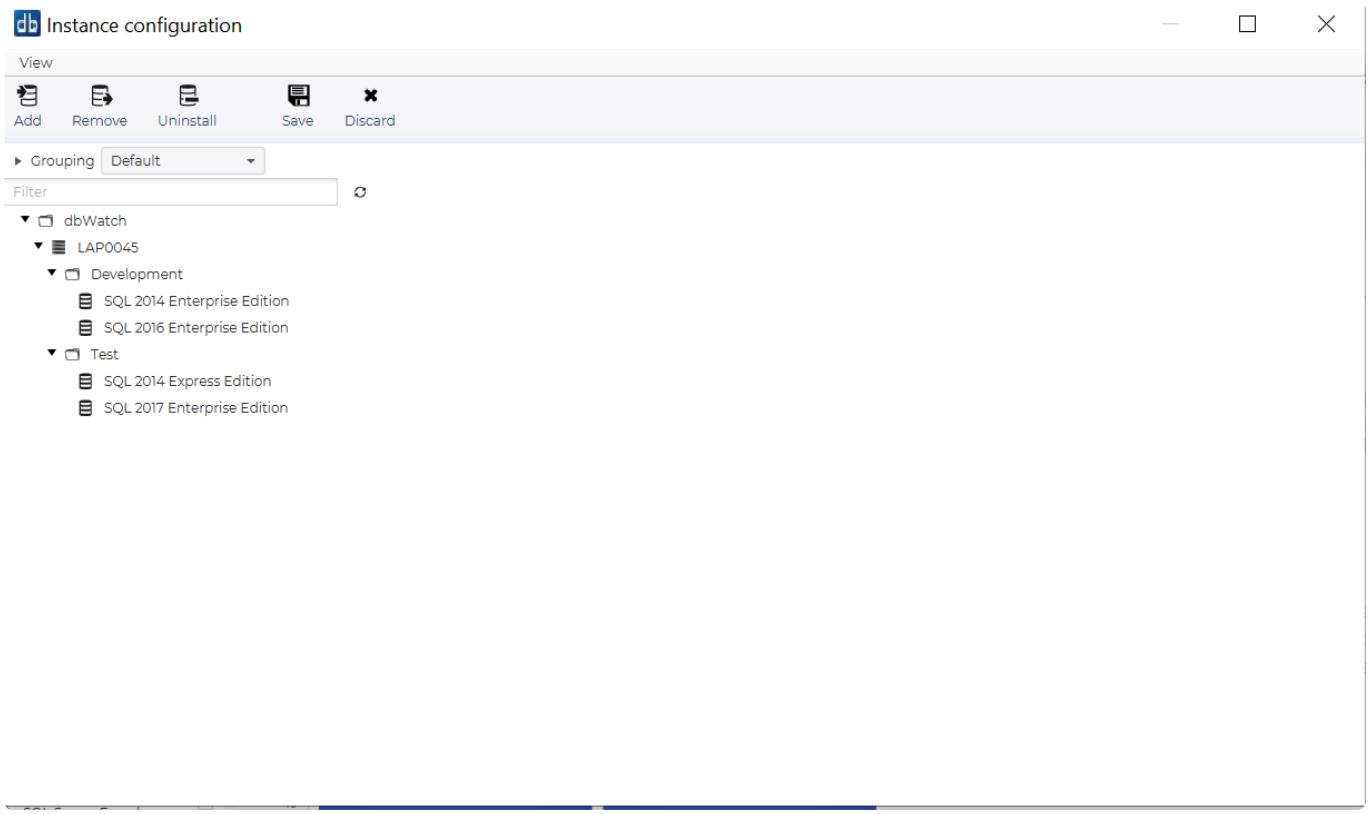
How to get here

Instance Configuration can be accessed by going to **Server > Server Configuration > Instance Configuration**, or by right clicking on an instance and selecting **Configure Instance** from the popup menu.



Main view

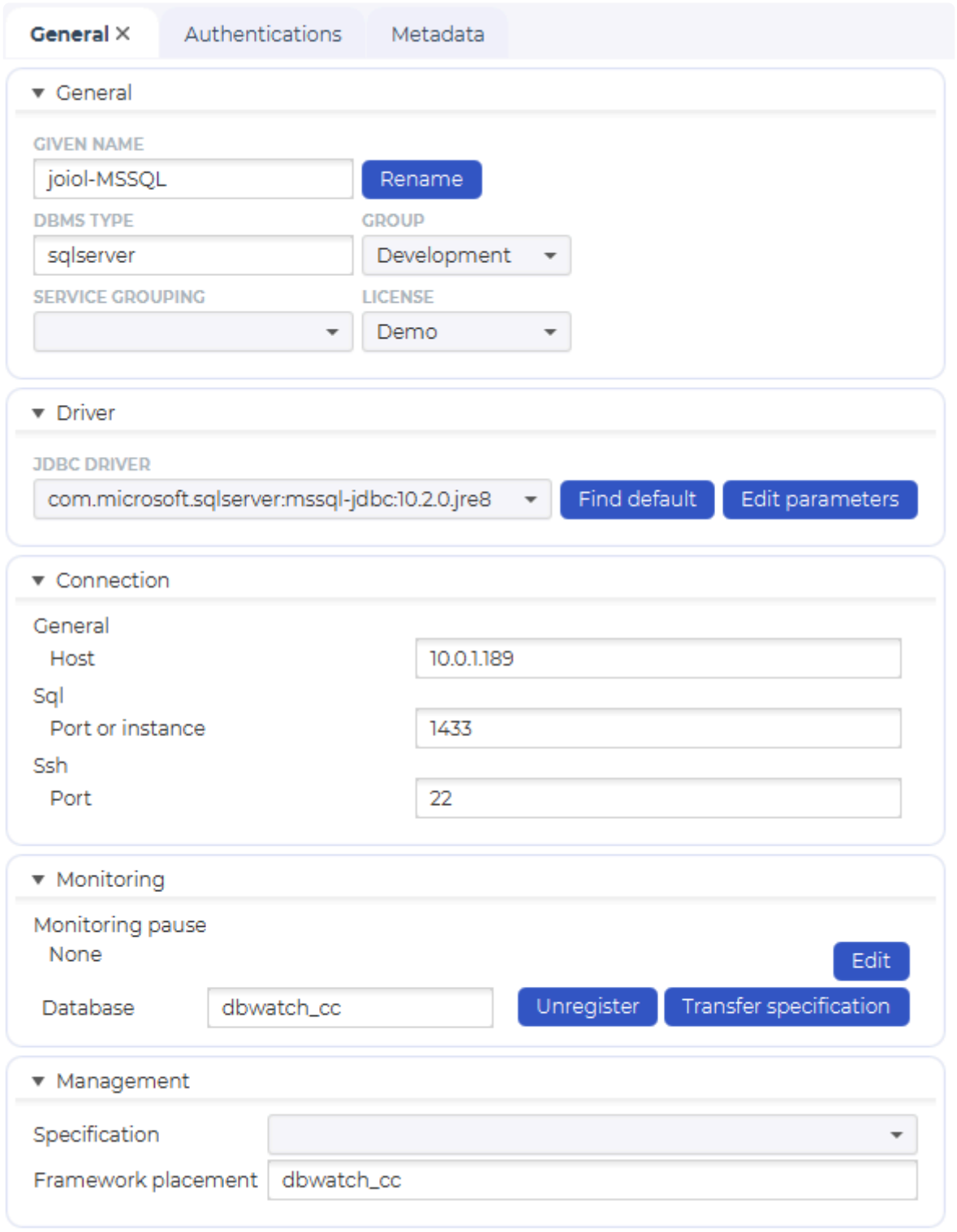
The instance configuration view consists of several parts. On the left side you see a tree are the list of your instances installed in CC. One the right pane, you can see the details of the highlighted instance. On the top are commands to add an instance, remove an instance, and uninstall a ninstance. You can also see the grouping filters.



The configuration panel is again divided into 3 main tabs.

General

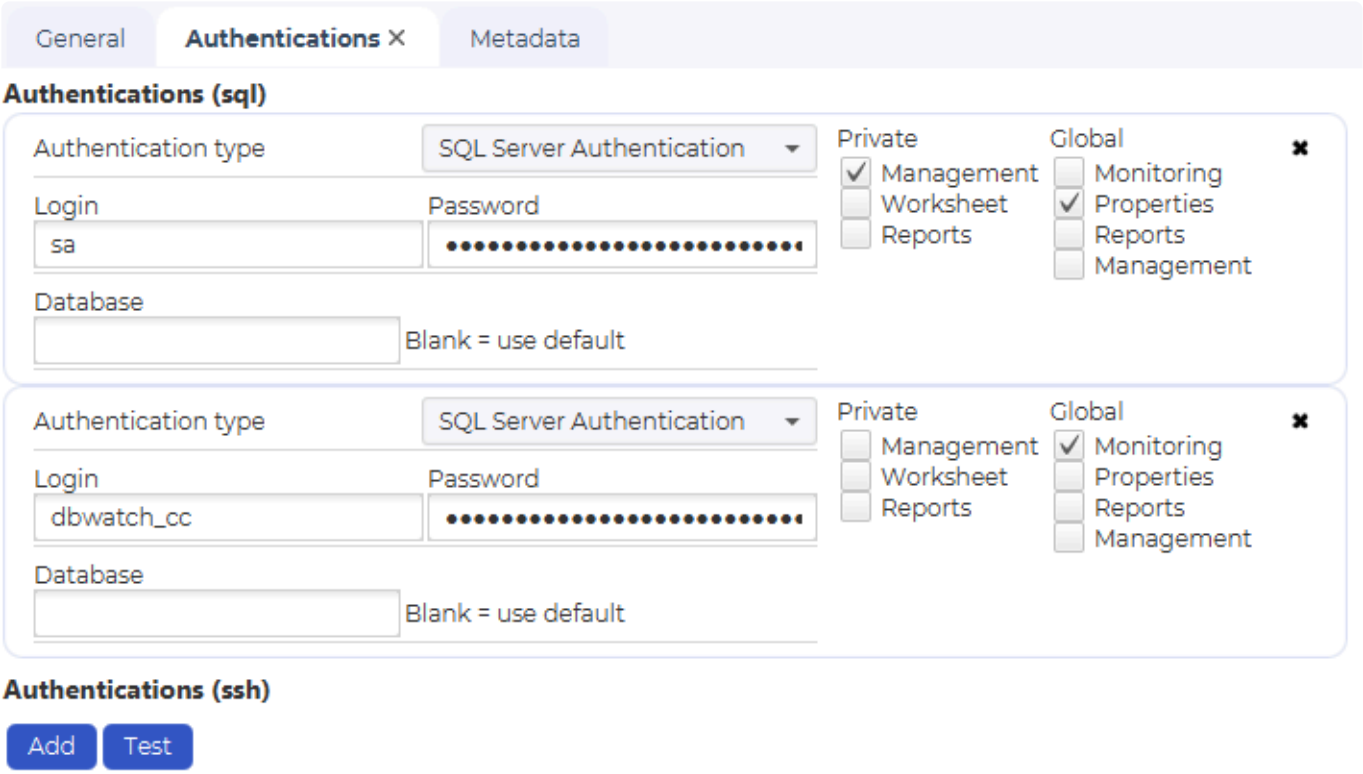
The first tab is **General**. This panel shows the Given name (named used to identify this instance in dbWatch), the DBMS type for the instance, and the group the instance is sorted under. The **Driver** panel specifies the jdbc driver used and lets you edit jdpb parameters (read more).The Connection panel shows connectivity details like host and port for the instance (this may vary a bit depending on the dbms type). Below this you have information about the database/schema used for advanced monitoring (more info) and monitoring polling pauses. Then on the bottom you have information used by the management module.



The information in these panels will also vary a bit depending on the dbms type.

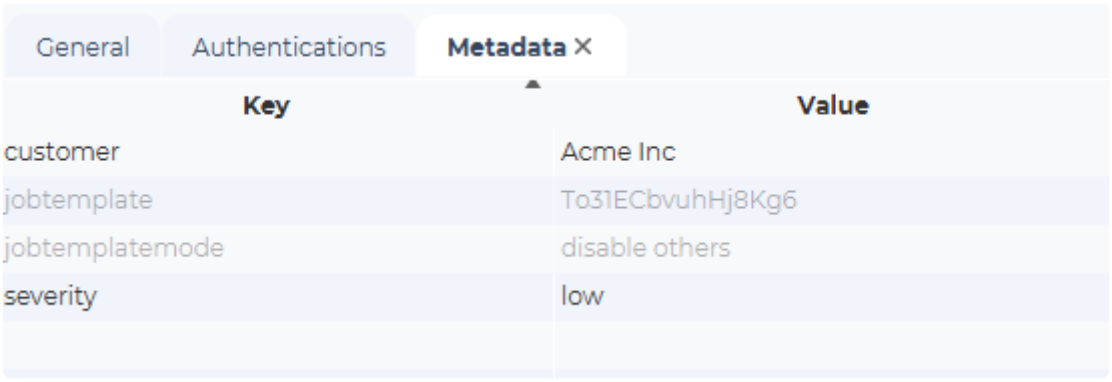
Authentications

In the Authentications tab you can specify the authentications used for the different modules in dbWatch. Management and Worksheet authentications are per Control Center user (changing these will only affect your user and not other DBA's using dbWatch). The Monitoring, Properties and Report authentications are global and thus effect all Control Center users.



Metadata

In the Metadata tab you can add/edit the metadata that is defined for this instance. More info on the property system here.

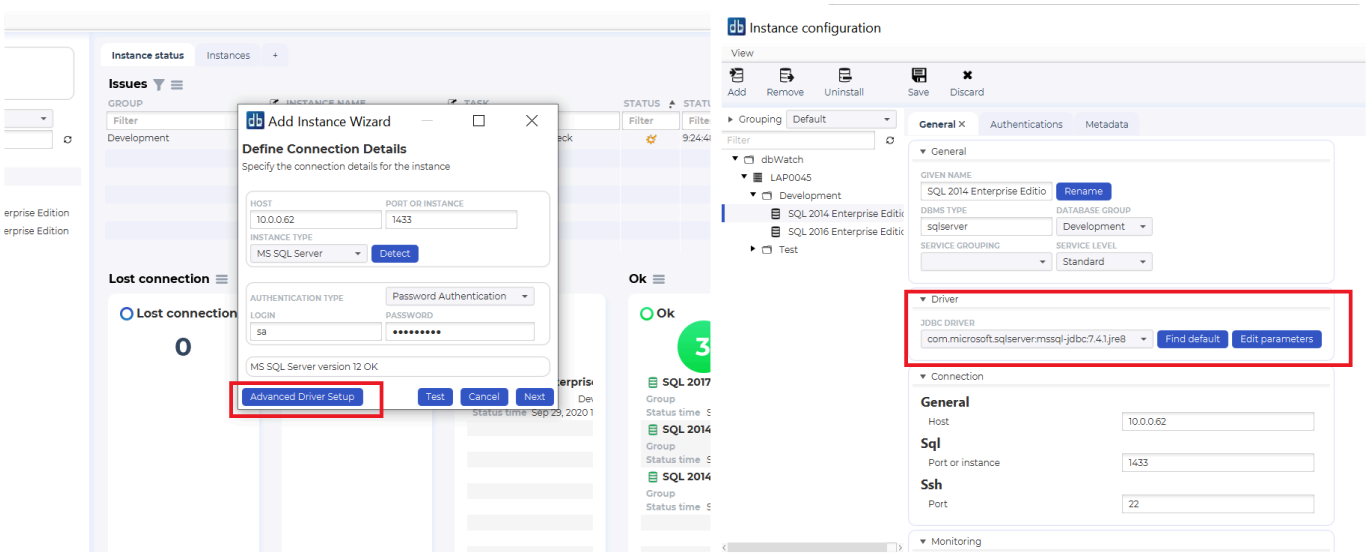


<< Instance management / JDBC Properties >>

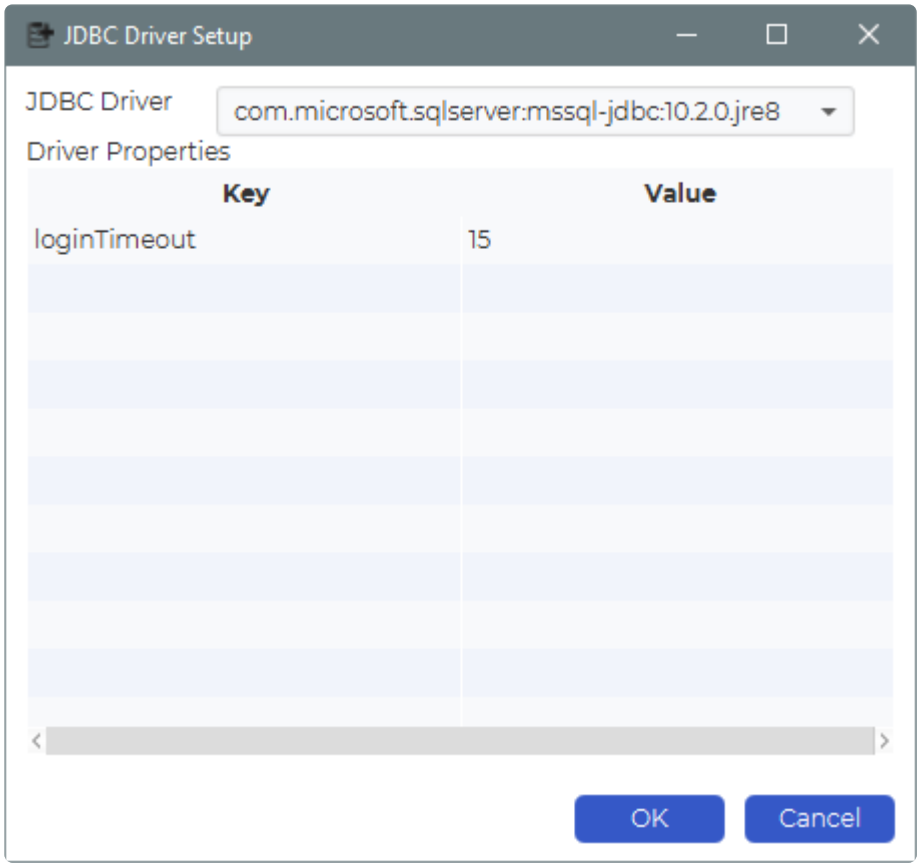
JDBC Properties

It is possible to set properties for the JDBC driver on specific instances.

You can find this under “Advanced Driver Setup” when adding SQL instances or under “Edit parameters” in the JDBC panel in the Configure Instance dialog.



The driver properties dialog looks like this.



The properties you set here will be applied to the JDBC connection on the database. The example above will set the login timeout to 15 seconds for an MS SQLServer instance.

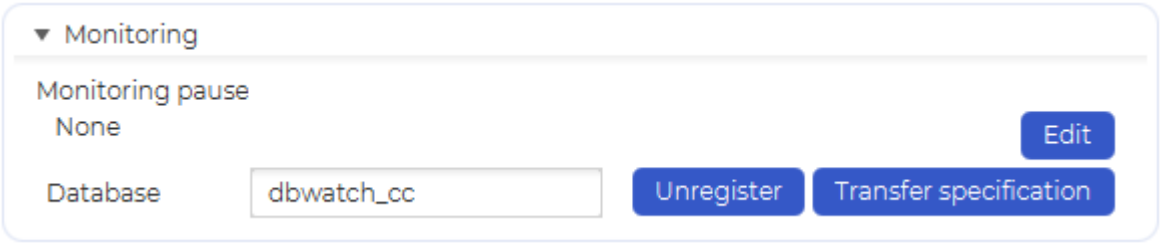
See documentation from the specific dbms platform for legal properties to set on the connection.

<< Instance Configuration / Monitoring Engine >>

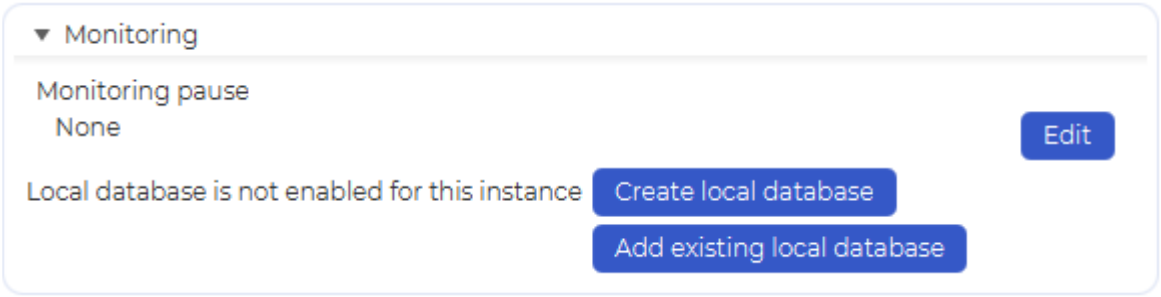
Monitoring Engine

The monitoring section of the **Configure Instance** view looks different depending on if you have a monitoring engine (i.e a database) defined on the instance.

With engine:



Without engine:



Text may vary a bit depending on the database type.

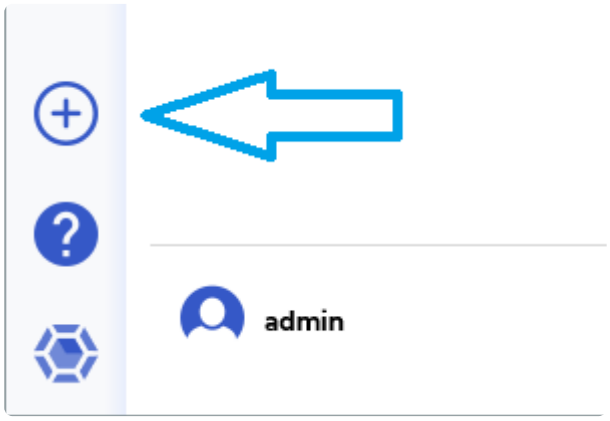
Button	Description
Unregister	Removes the reference to the monitoring database from this instance. (The database is not removed from the instance)
Transfer specification	Moves the reference to the monitoring database to another instance.
Create local database	Creates a monitoring database database on the instance
Add existing local database	Registers an existing monitoring database on this instance

<< JDBC Properties / Adding a SQLServer Instance >>

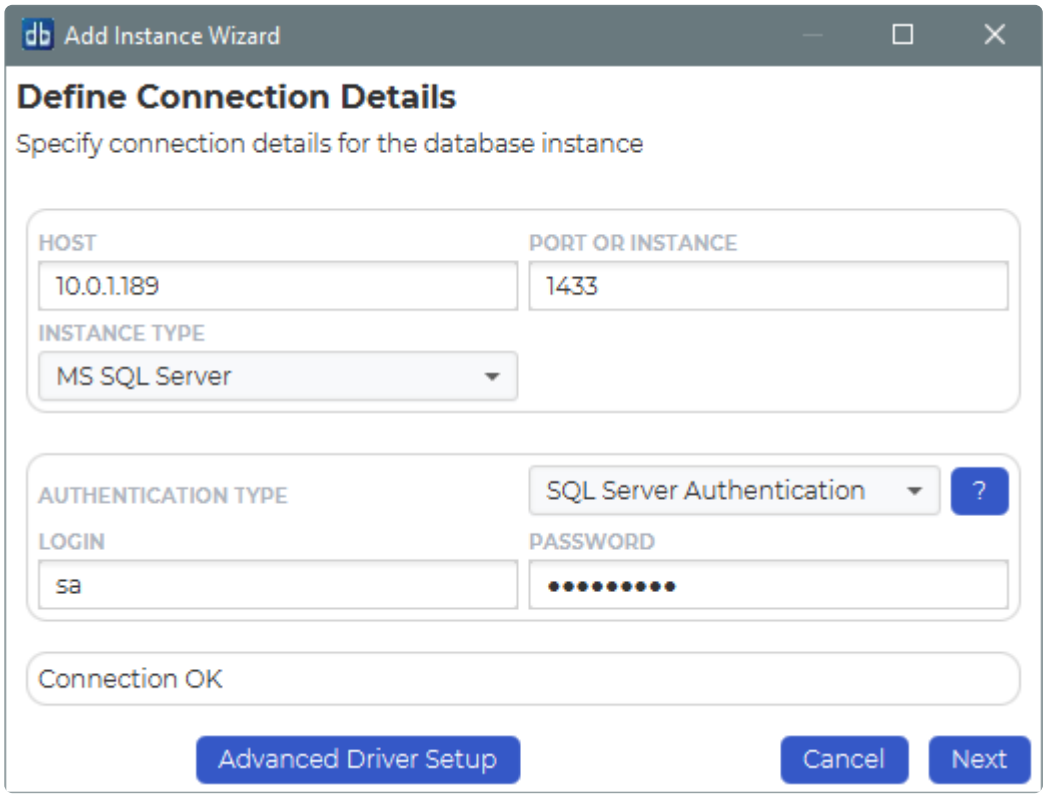
Adding a SQLServer Instance

How to get here

To open the “Add Instance Wizard”, click on the “Plus (+)” sign on the bottom left of the dbWatch Client.



Select Instance Type and Input Connection Details



Specify the host and port number.

dbWatch will attempt to select the correct instance type based on the entered port number. If the instance type is wrong, select the correct one in the drop down.
The authentication types available will depend on the selected instance type.

Choose an authentication type whether “SQL Server Authentication”, “OS Authentication” or “Kerberos”.

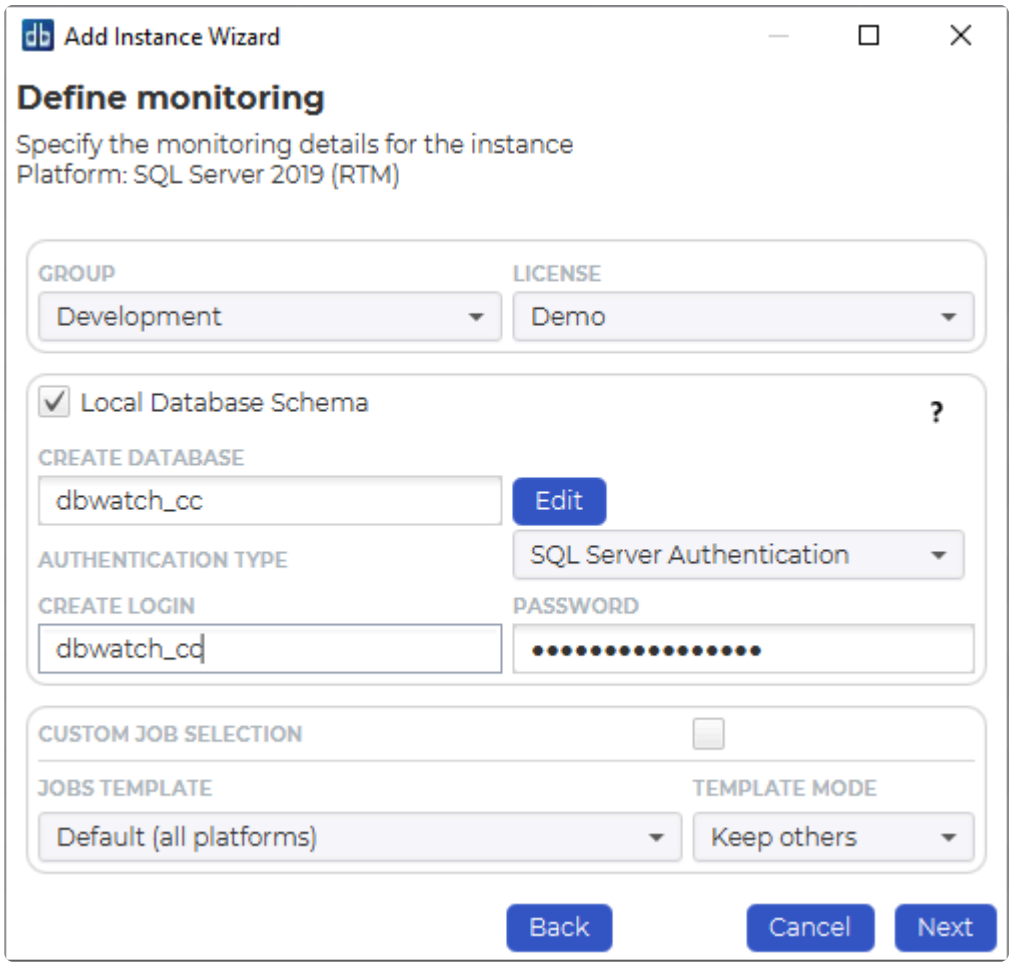
You need to specify a login with a server administrator role that will be used during the installation.

OS Authentication	Connects to the MS SQL Server using the account the dbWatch Server Services is running under. This could require additional configuration, more here
SQL Server Authentication	Connects using the provided login and password.
Kerberos	Connects through kerberos using the provided login and password. This could require additional configuration, more here

By default dbWatch will select the most apropiate JDBC driver for the database instance. It is possible to select a specific JDBC driver and set driver properties by clicking on “Advanced Driver Setup”. Read more.

Click “Next”.

Define Monitoring



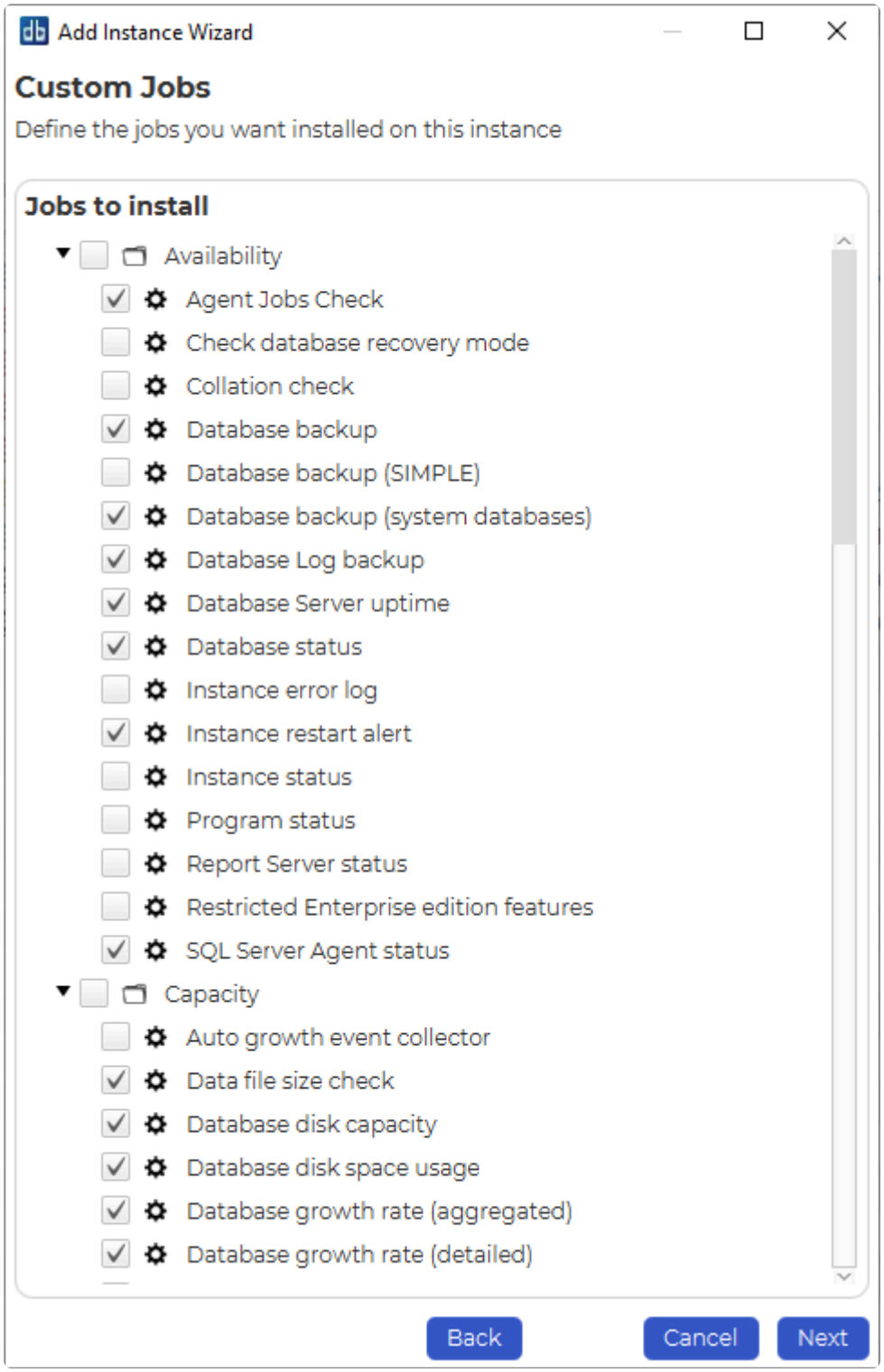
Specify the group and license for the instance. Default groups are **Development**, **Test** and **Production**, but you can easily create and customize your own groups based on your requirements.
The licensing options available will depend on your license. The selected option will determine what monitoring jobs are available.

You can choose to register this instance with or without a local database schema. This choice will influence the available monitoring jobs. Some jobs require objects in the database (procedures, tables etc.), others do not. We recommend installing a database schema, as this will give the most monitoring options.

By default you install the jobs that are part of a template. You can choose the template you use, and have your own custom templates. If you want to enable custom job selection while adding the instance, you can click on the “Custom job selection” radio button.

Click “Next”.

If you do that the next window will allow this selection:

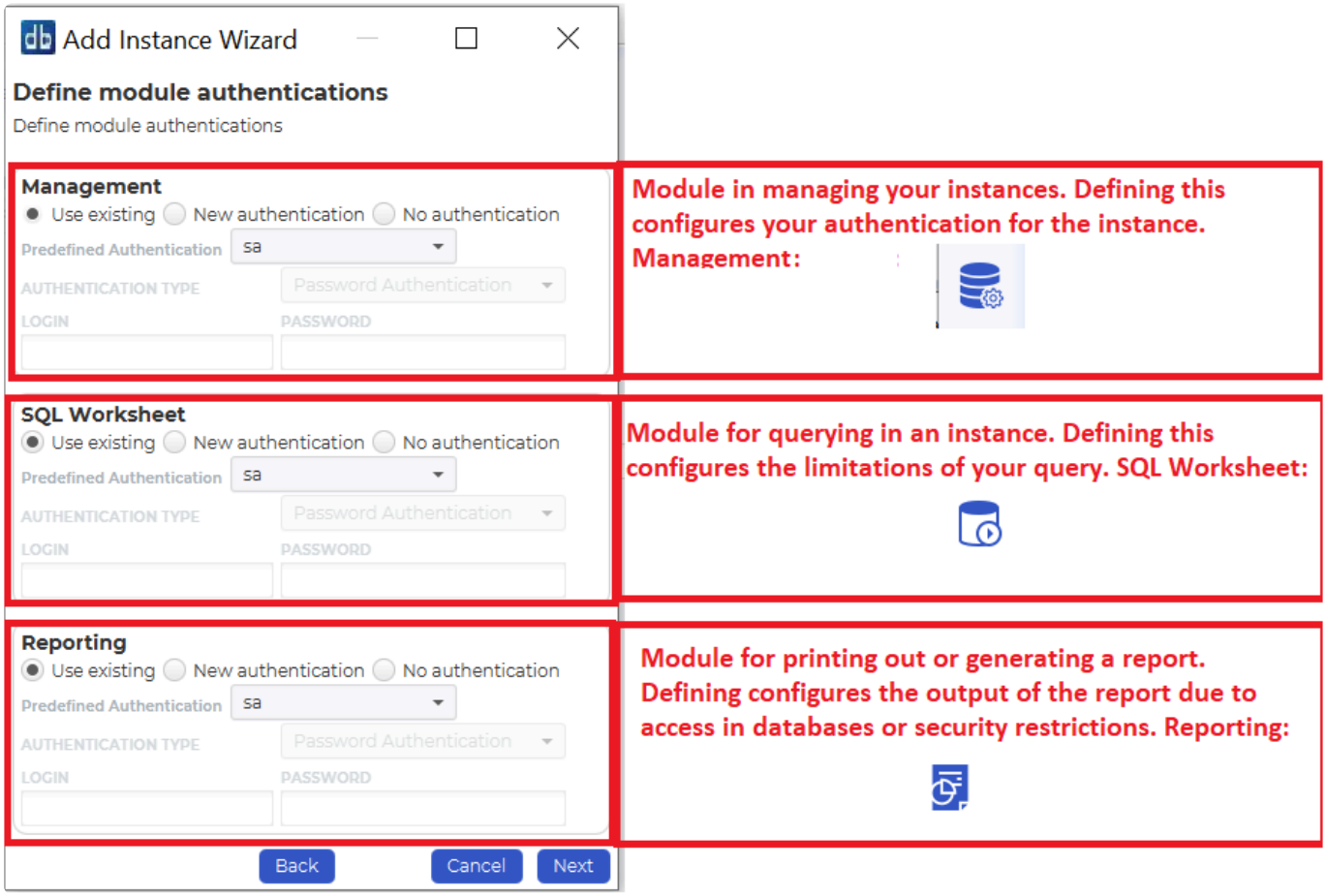


You can stick with the jobs that are selected by default or customize the selection as needed.

Click “Next”.

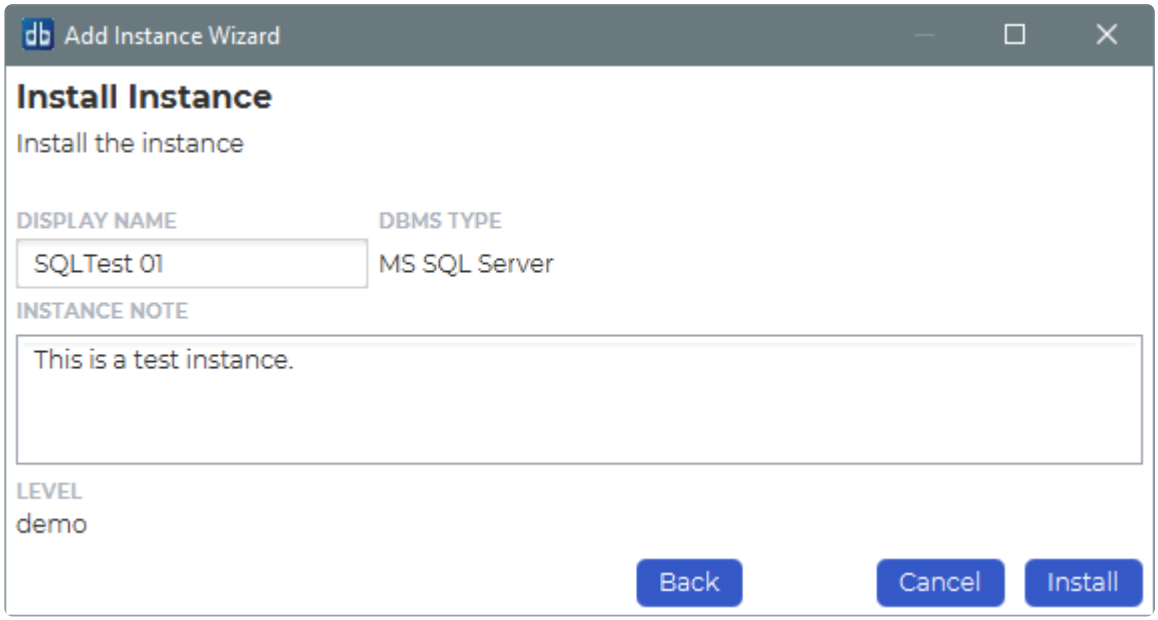
Define Module authentications

Within this window you have an option to choose which credential to use for each module within dbWatch.
You can use the existing credentials, or select specify a new one.



If you've finished specifying a credential for each module. Click Next.

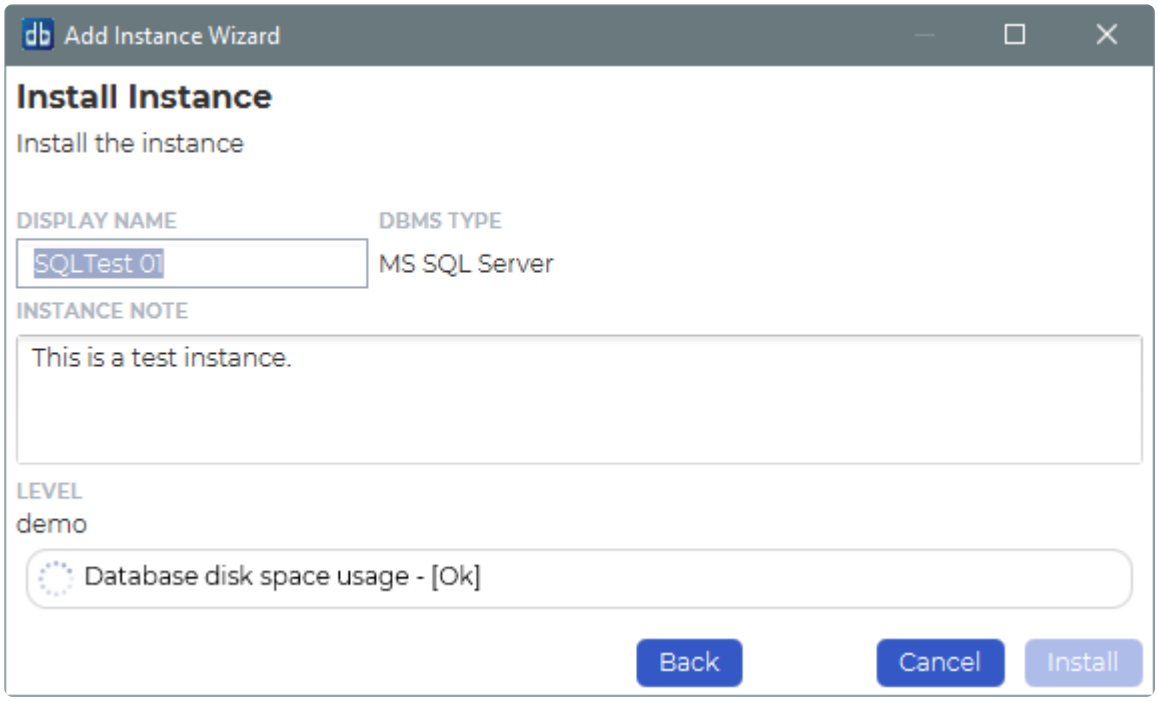
Instance Installation



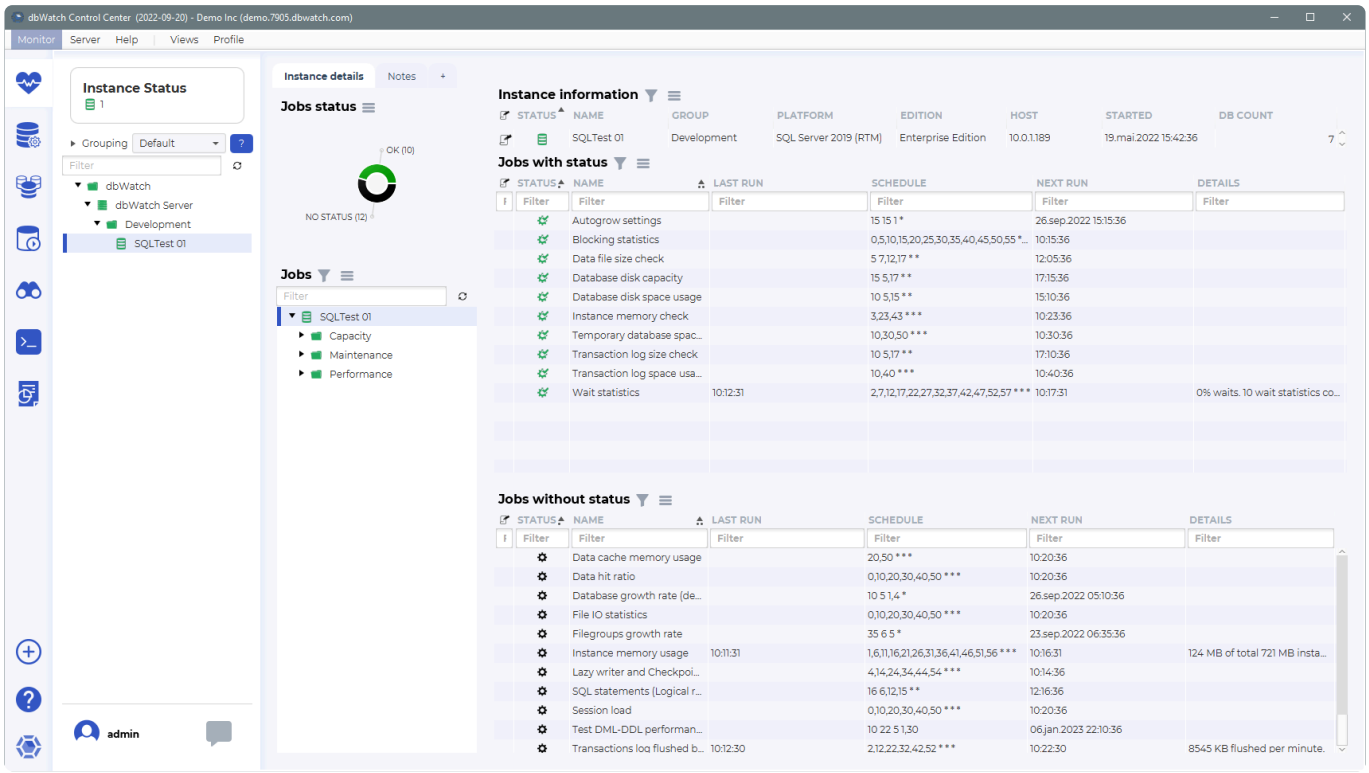
Specify a display name for this instance. You can also provide a description or any relevant information you want to remember about this instance.

Click Install to start installing.

Underneath you will see the status of various jobs being installed.



We've now successfully added the SQL Server instance in our dbWatch monitor.

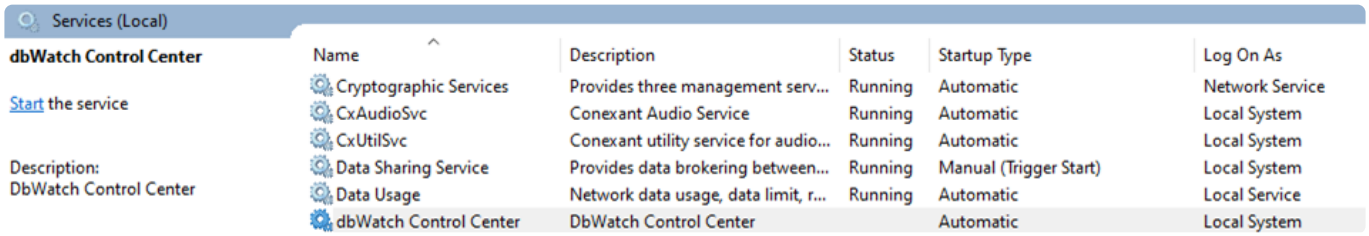


<< Monitoring Engine / Using OS Authentication >>

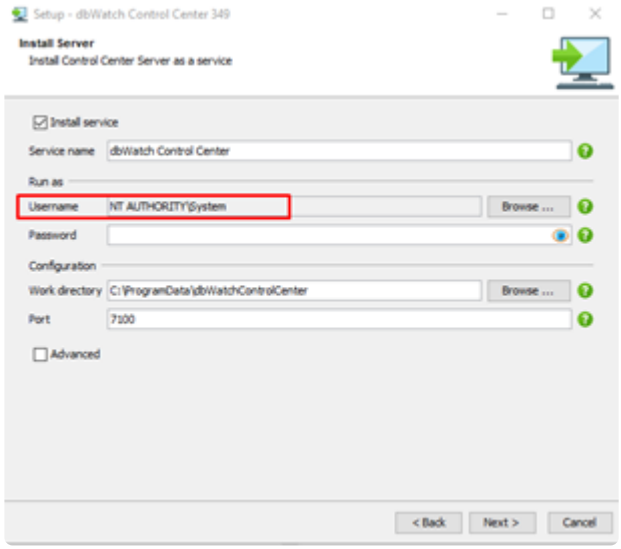
Using OS Authentication

When connecting to Microsoft SQL Server instances it is possible to login with Windows Authentication (called OS Authentication in dbWatch). The accout used is the same account as the dbWatch Control Center service runs as.

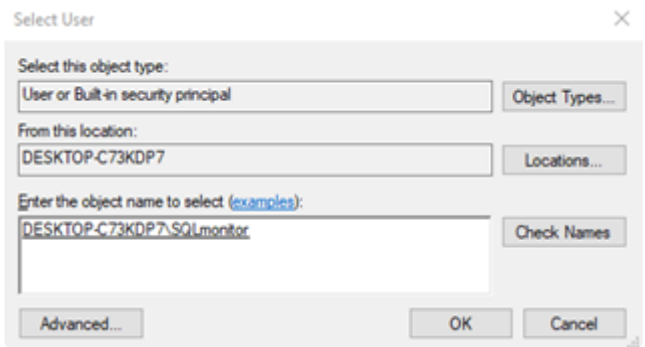
Per default the dbWatch Control Center Service running on Windows is configured using the system account. Running under this account will only make it possible to connect to SQL Server using SQL Server Authentication login.



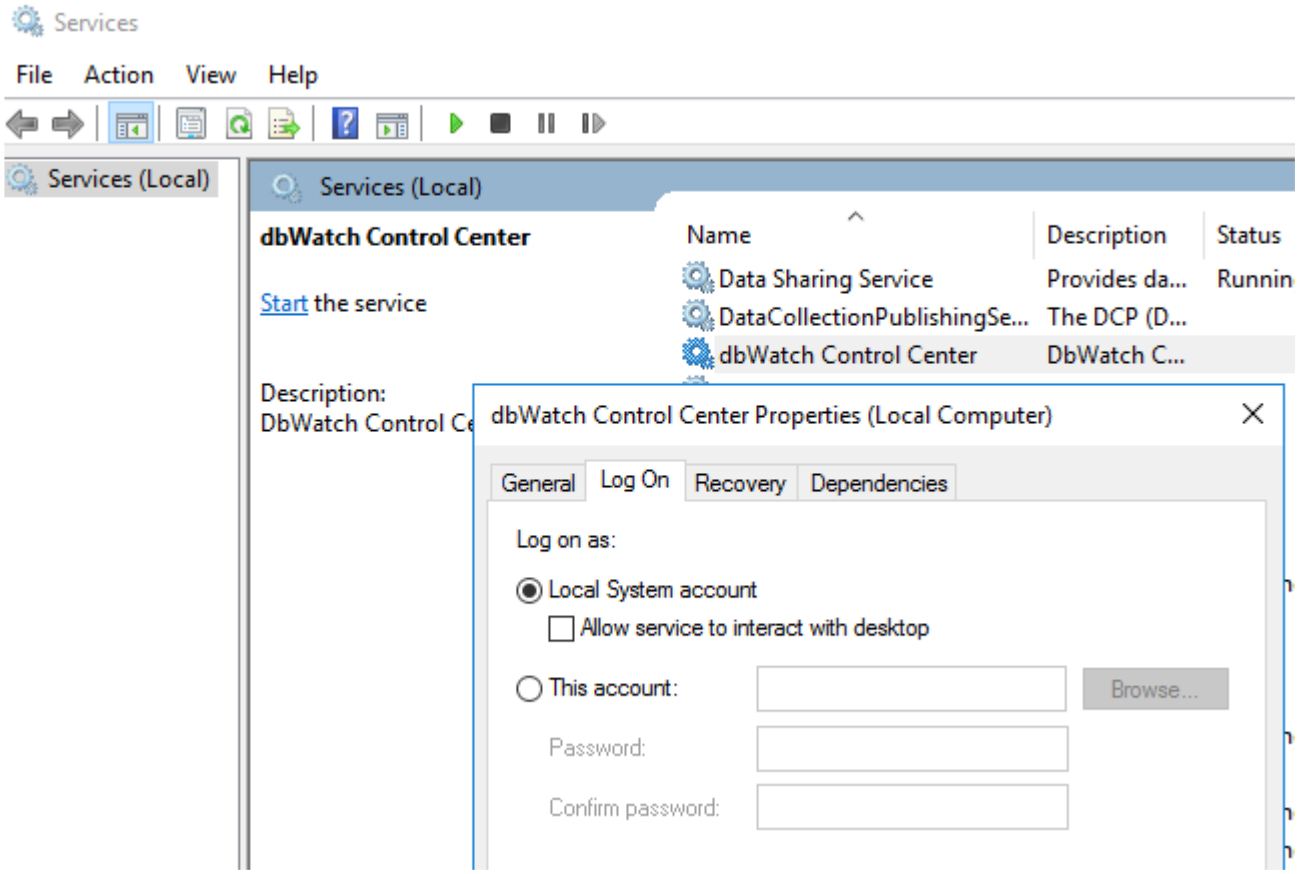
In the dbWatch installer “Install Server” step you can specify what account to use.



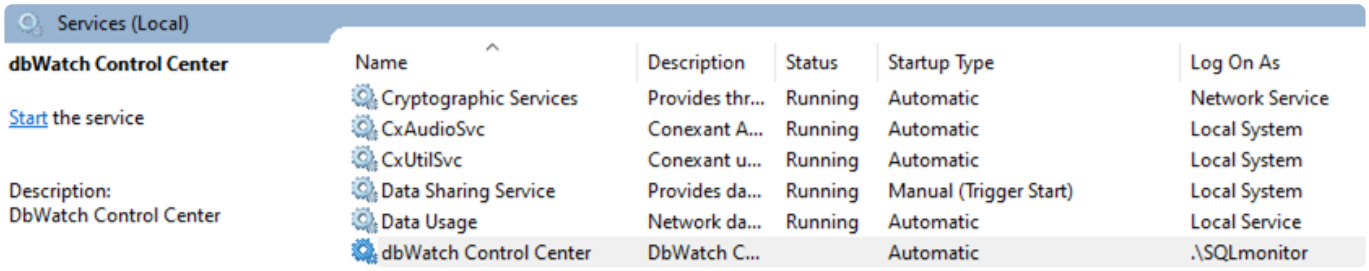
To be able to use Windows Authentication login (which use credentials from Windows) the dbWatch Control Center Service must run under an account with sysadmin privileges on the SQL Server instances.
Click “Browse” and specify the correct Windows account with correct credentials.



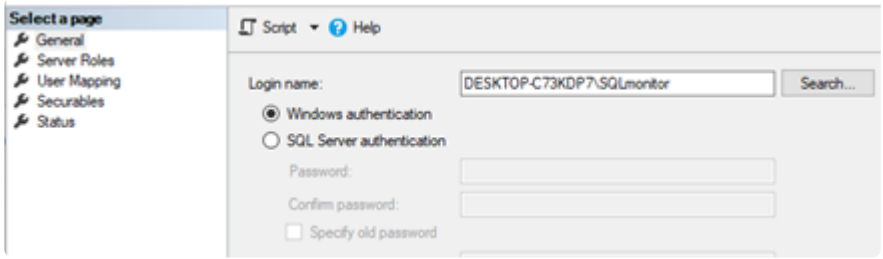
You can also change this later by right clicking on the dbWatch Control Center service in the “Services” tool, selecting “Properties” and then navigating to the “Log On” tab.



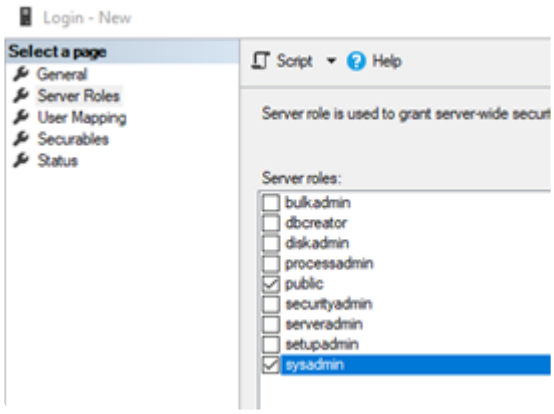
This example shows the dbWatch Control Center Service running under a Windows account “SQLmonitor”.



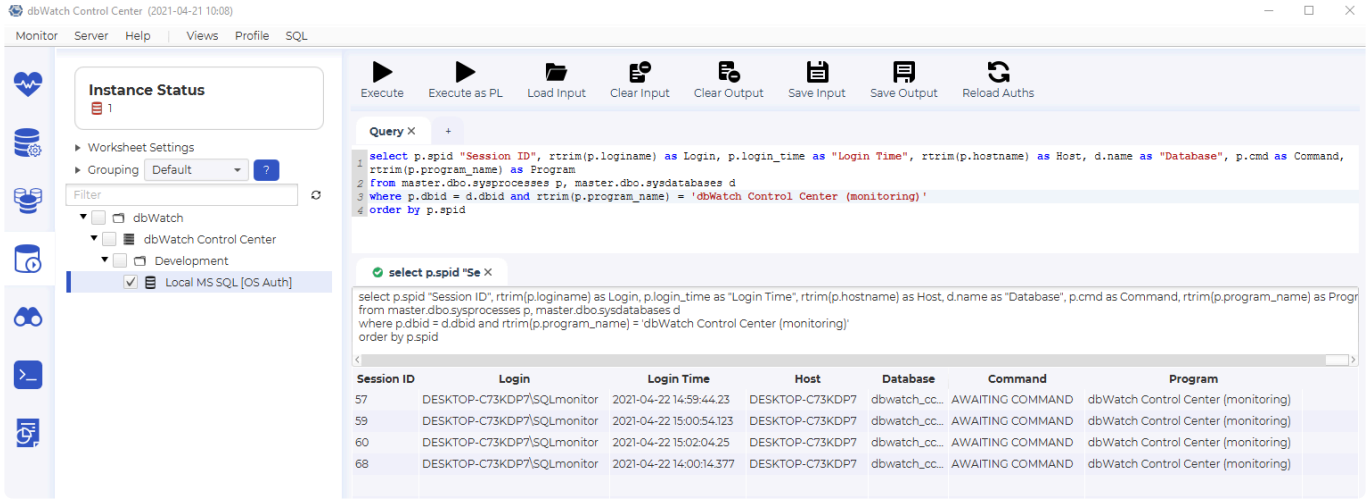
The “SQLmonitor” account is mapped in to a login on the SQL Server instance



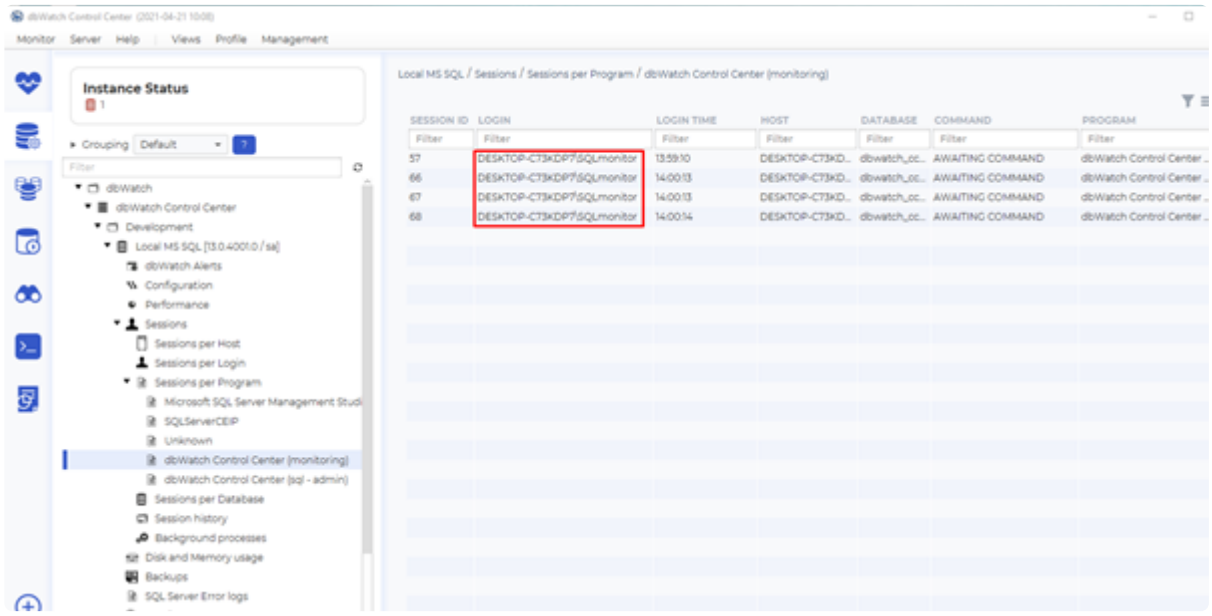
The “SQLmonitor” login needs the sysadmin server role to work properly.



After adding an MS SQL Server instanace to dbWatch Control Center you will be able to see a small connection pool (4 sessions) connected to the SQL Server.



You can also see all sessions from dbWatch Control Center Service (Server) in dbWatch Management GUI.

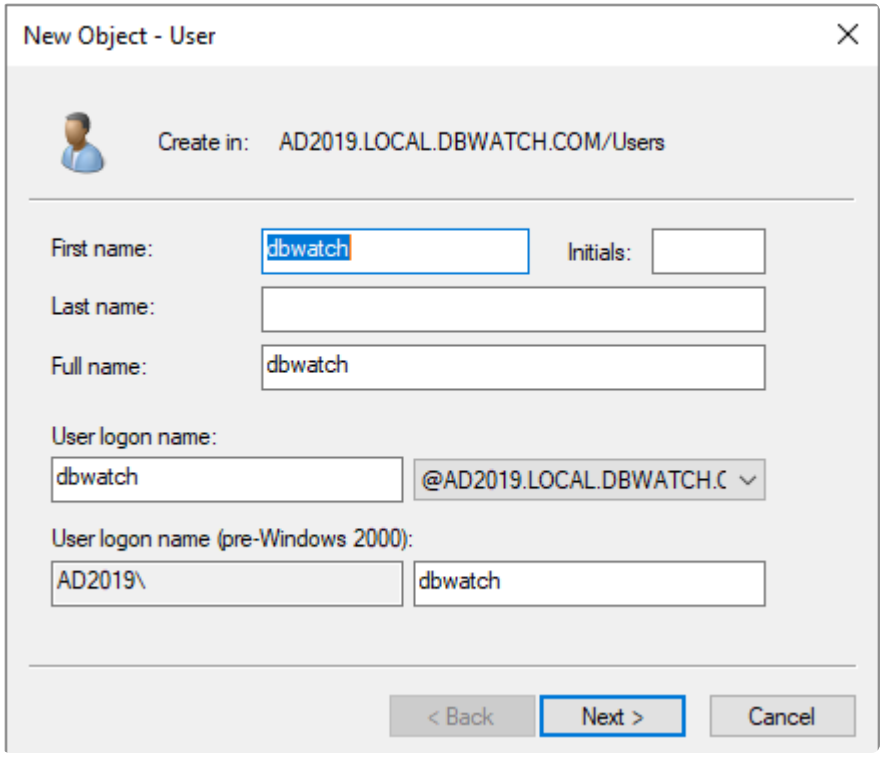


[<< Adding a SQLServer Instance] / [Adding SQLServer instances with kerberos >>]

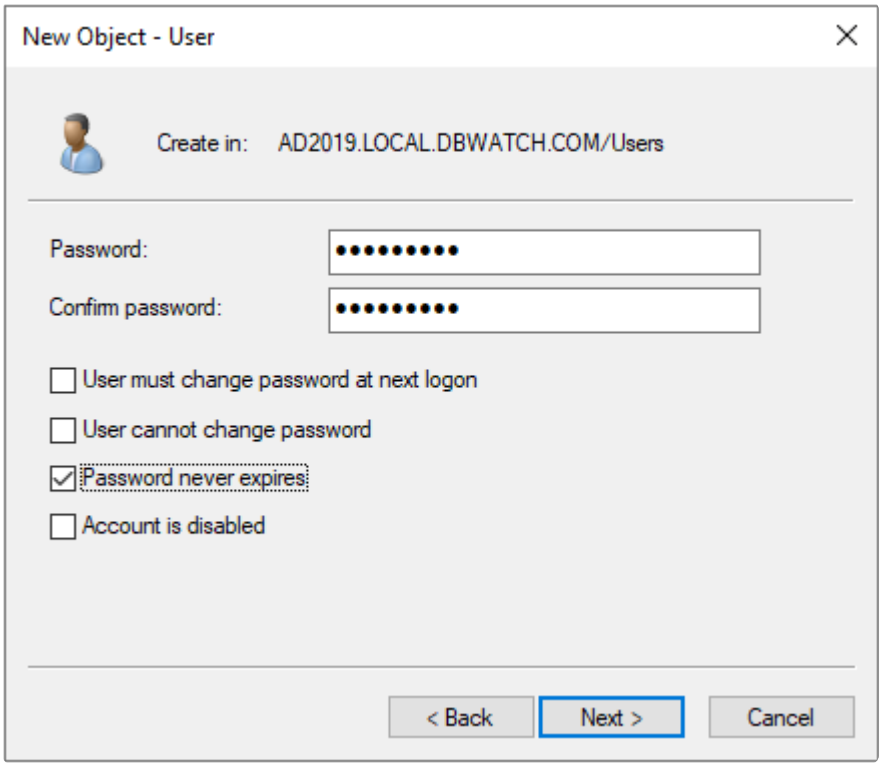
Adding SQLServer instances with kerberos

Using the kerberos authentication option requires some additional setup.

We need dbwatch user added to the domain:

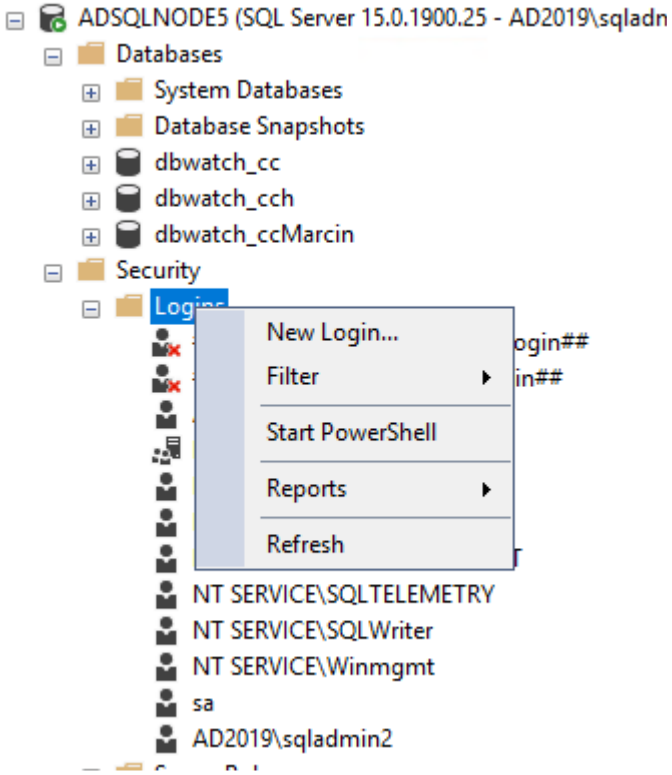


First we create a dbwatch user that is a normal domain user.

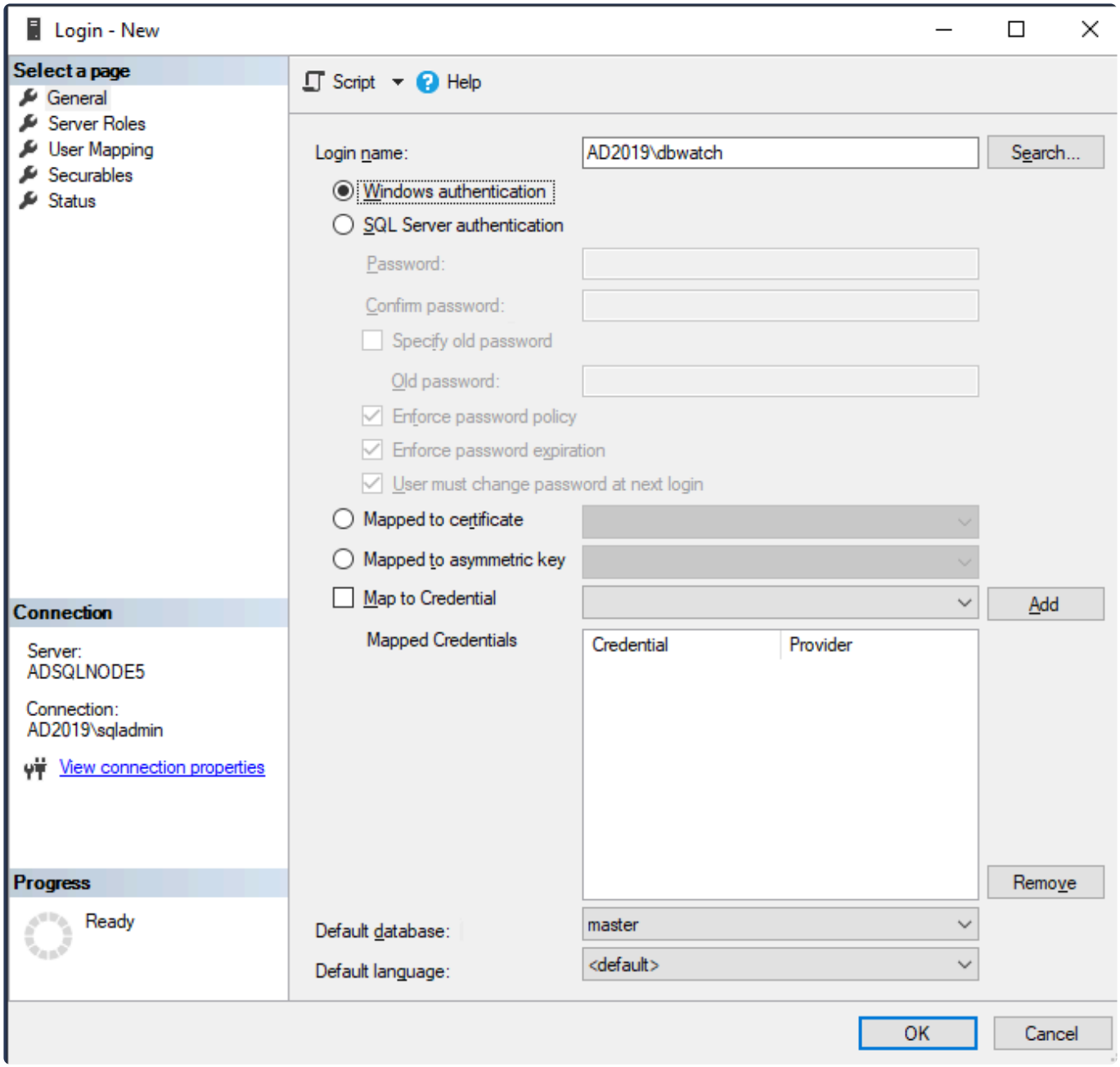


We set a passord, and make sure it dont expire.

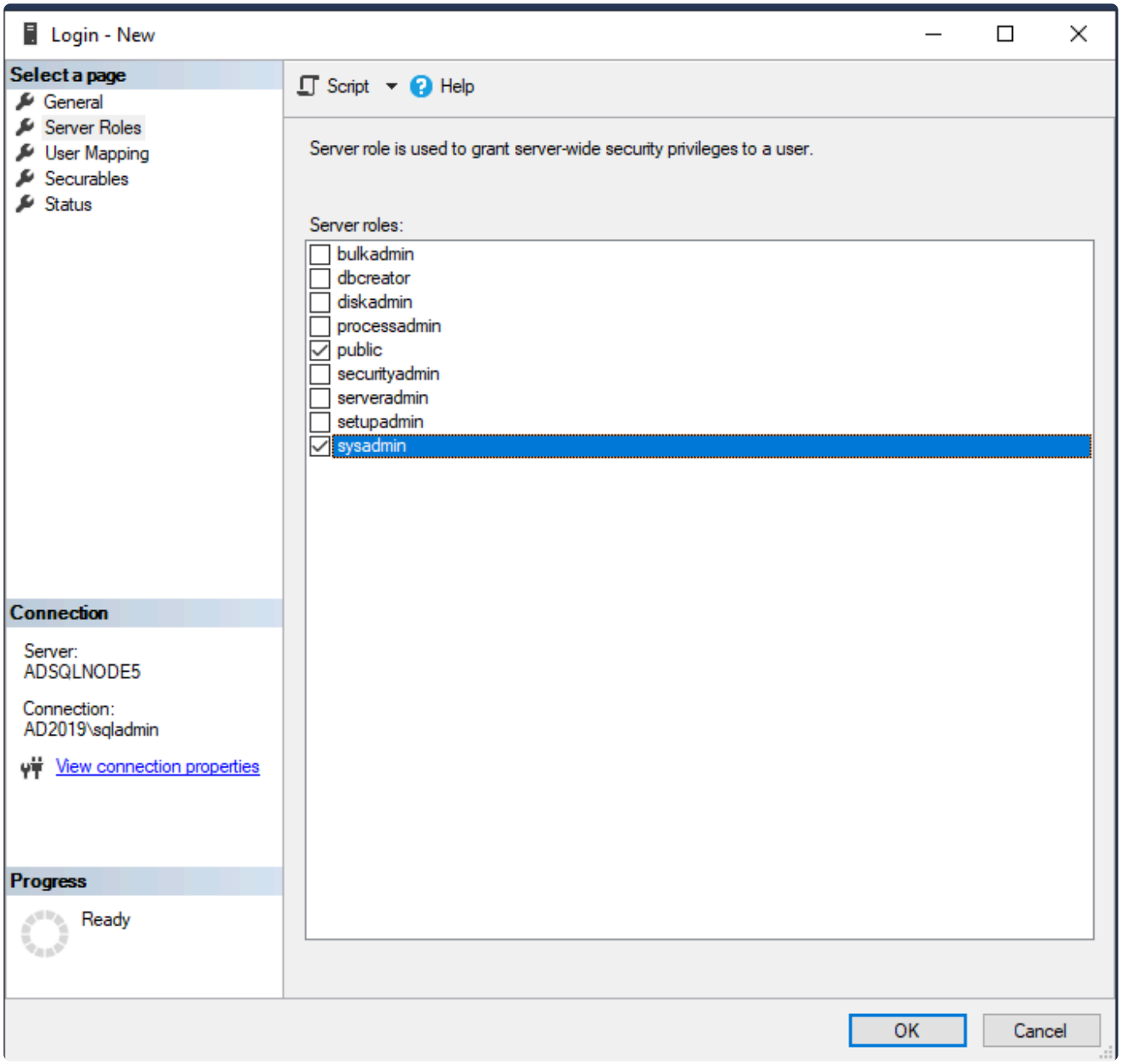
And we must add this user to the SQL Server database:



Create a new login



And select Windows authentication and use the domain user you created earlier

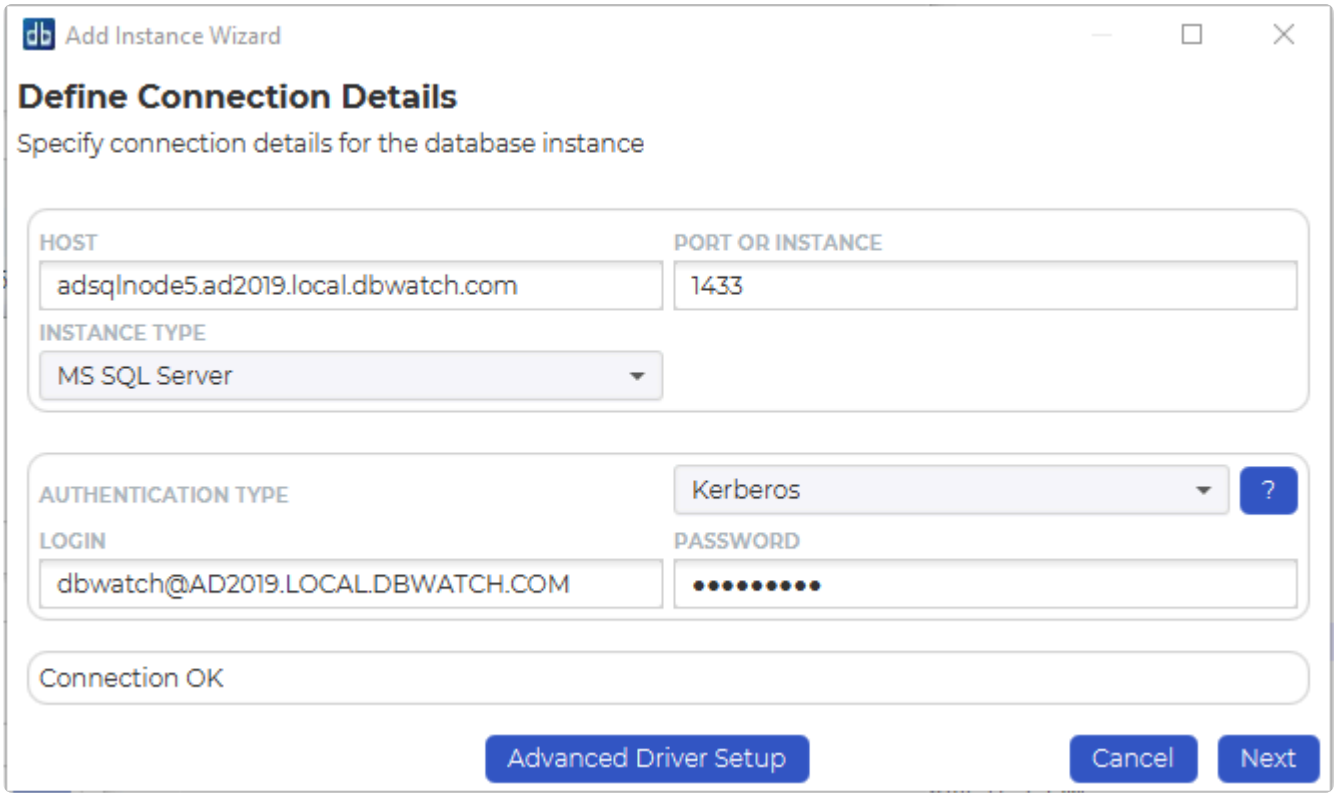


This user needs the sysadmin server role in the database instance.

krb5.conf

```
[libdefaults]
dns_lookup_kdc = false
dns_lookup_realm = false
ticket_lifetime = 86400
renew_lifetime = 604800
forwardable = true
udp_preference_limit = 1
realm_try_domains = 1

[realms]
AD2019.LOCAL.DBWATCH.COM = {
kdc = ADSQLNODE5.AD2019.LOCAL.DBWATCH.COM
admin_server = ADSQLNODE5.AD2019.LOCAL.DBWATCH.COM
default_domain = AD2019.LOCAL.DBWATCH.COM
}

[domain_realm]
.AD2019.LOCAL.DBWATCH.COM = AD2019.LOCAL.DBWATCH.COM
AD2019.LOCAL.DBWATCH.COM = AD2019.LOCAL.DBWATCH.COM

```

There also has to be a krb5.conf file located in C:\ProgramData\dbWatchControlCenter\ directory
This file must specify the different realms / active directory domains you want to log into. The example setup is for the domain AD2019.LOCAL.DBWATCH.COM, and your domain name and the ad server will differ.
Multiple domains can be added, with their own realm and domain_realm tags. Changes in this file required restart of the dbWatch Control Center service.



After this is configured you should be able to add a instance with the normal add instance wizard. Note that the host has to match the domain record, so IP address will not usally work. Also the username you created followed by a @ sign and the domain name. The domain name is case sensitive and usally will need to be provided in uppercase. Drivers are selected automatically.

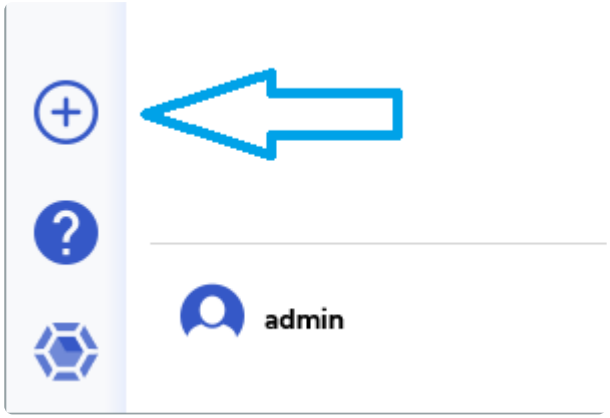
Adding an Oracle Instance

Oracle Prerequisites

1. Oracle databases need to have a listener process running and also allow for an external login with sysdba user or user with enough privileges.
2. You need a privileged user, such as sys as sysdba, another user with sysdba privileges or another user with enough privileges depending on what you want to install.
3. This privileged user must be able to grant privileges to a monitoring user, by default dbwatch.
4. The monitoring user **must be a different user** from the privileged user.
5. There is no need to create the monitoring user before you start this install, its created by the privileged user as part of the process.

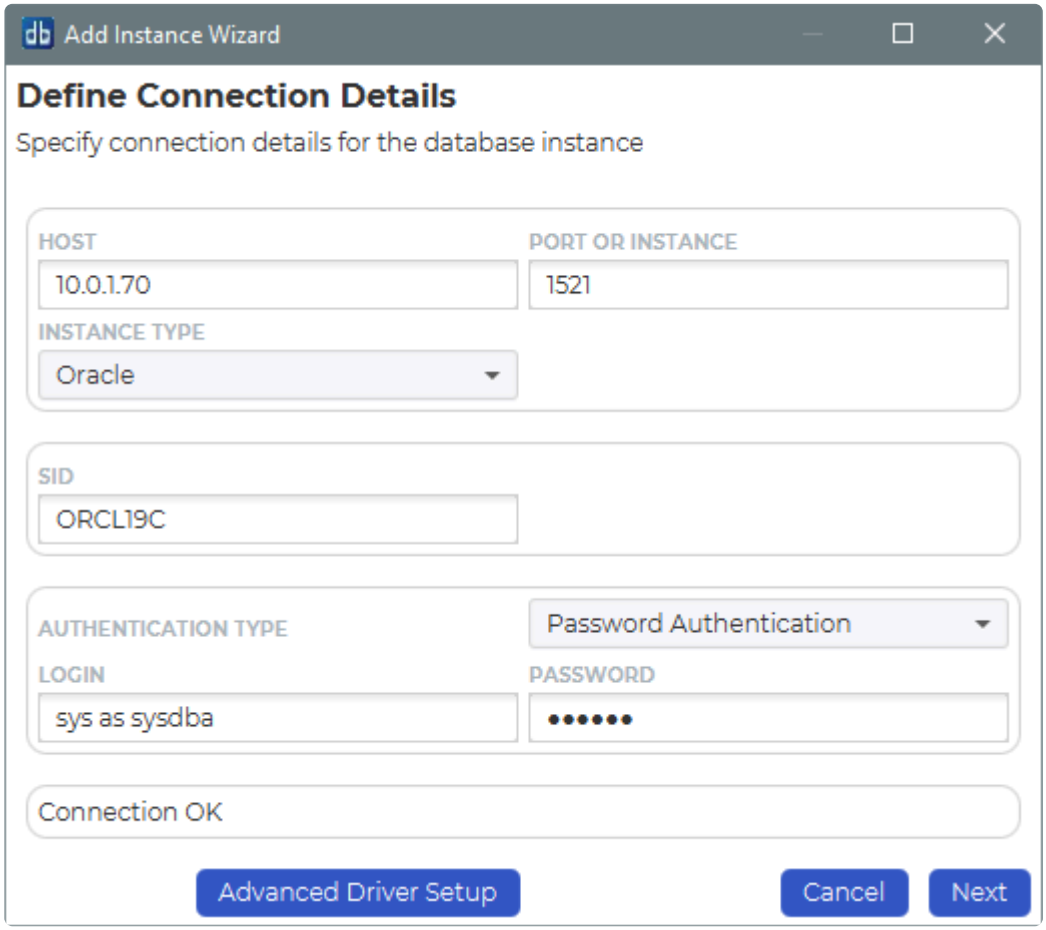
How to get here

To open the “Add Instance Wizard”, click on the “Plus (+)” sign on the bottom left of the dbWatch Client.



This will bring up the “Add Instance” wizard, which will guide you through the following steps.

Select Instance Type and Input Connection Details



The **Host** info is the hostname or ip address of the database instance.
The **Port** is the port number where the listener process is listening on.

In the **SID / Service name** field allows you to specify the SID or Service name for the Oracle instance you are about to add.

dbWatch will attempt to select the correct instance type based on the entered port number. If the instance type is wrong, select the correct one in the drop down.
The authentication types available will depend on the selected instance type.

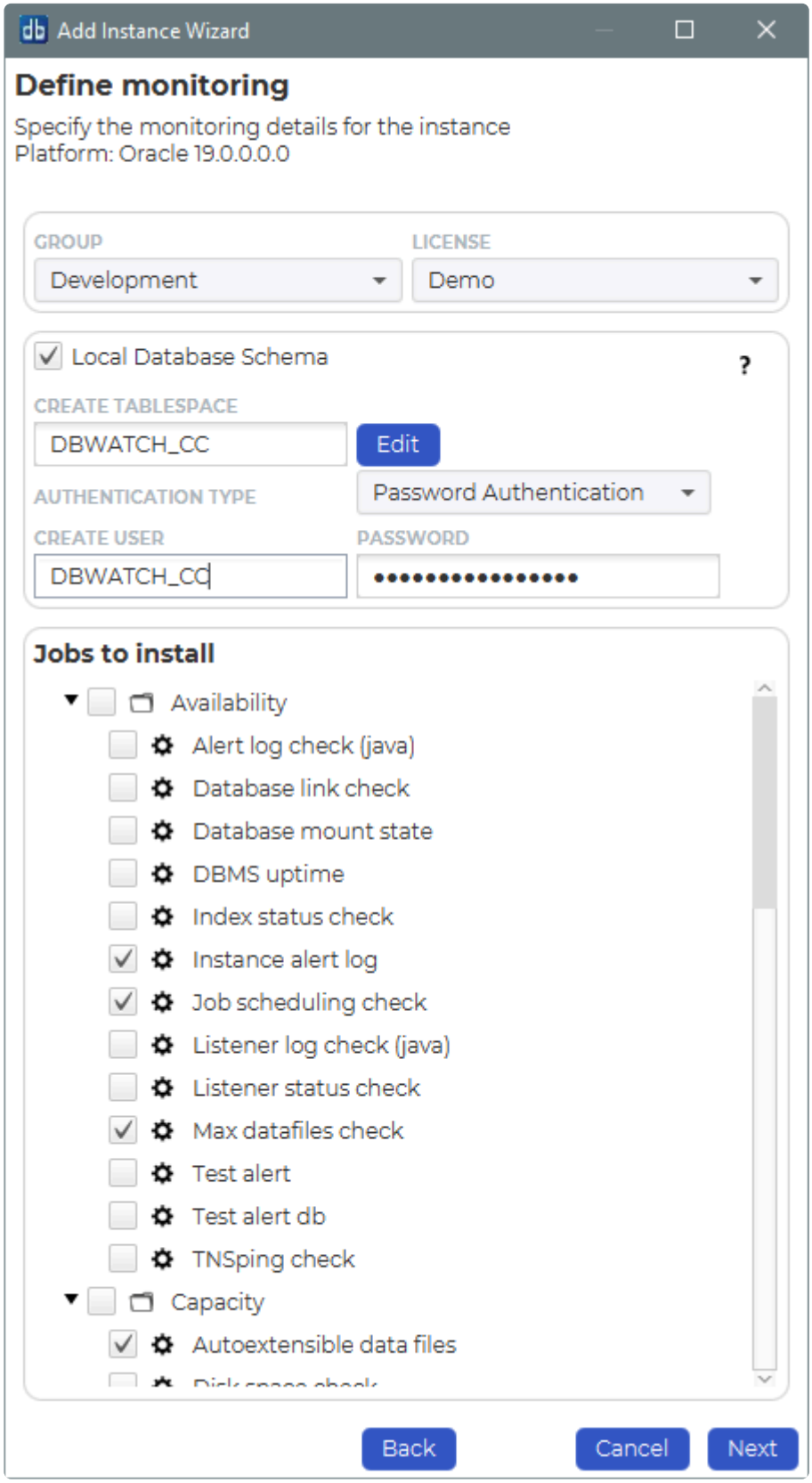
Choose an authentication type whether a password or Kerberos Authentication.
You need to specify a user with sysdba privileges that will be used during the installation.

Password Authentication	Connects using the provided username and password.
Kerberos Authentication	Connects using kerberos.

By default dbWatch will select the most apropiate JDBC driver for the database instance. It is possible to select a specific JDBC driver and set driver properties by clicking on “Advanced Driver Setup”. Read more.

Click “Next”.

Select Instance Type and Connection Details



Specify the group and license for the instance. Default groups are **Development**, **Test** and **Production**, but you can easily create and customize your own groups based on your requirements.
The licensing options available will depend on your license. The selected option will determine what monitoring jobs are available.

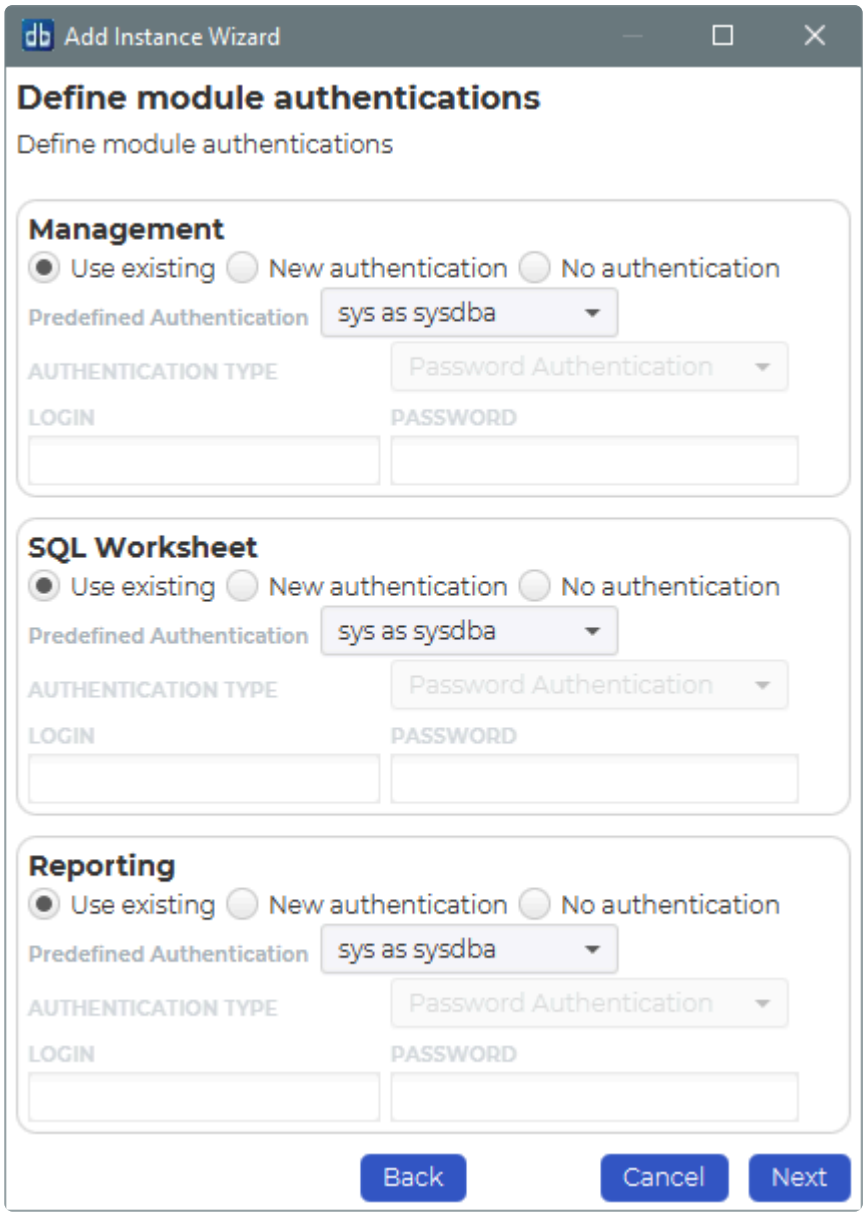
You can choose to register this instance with or without a local database schema. This choice will influence the available monitoring jobs. Some jobs require objects in the database (procedures, tables etc.), others do not. We recommend installing a database schema, as this will give the most monitoring options.

You can stick with the jobs that are selected by default or customize the selection as needed.

Click “Next”.

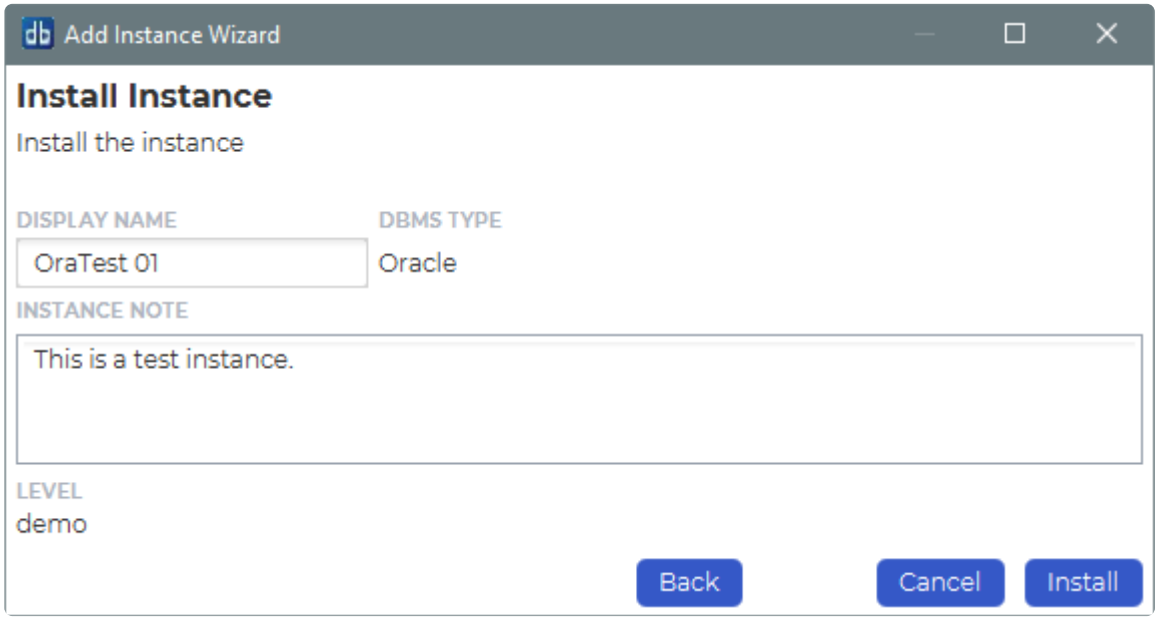
Define Module authentications

Within this window you have an option to choose which credential to use for each module within dbWatch.
You can use the existing user or select a new user whether password or OS authentication.



If you’ve finished specifying a credential for each module. Click Next.

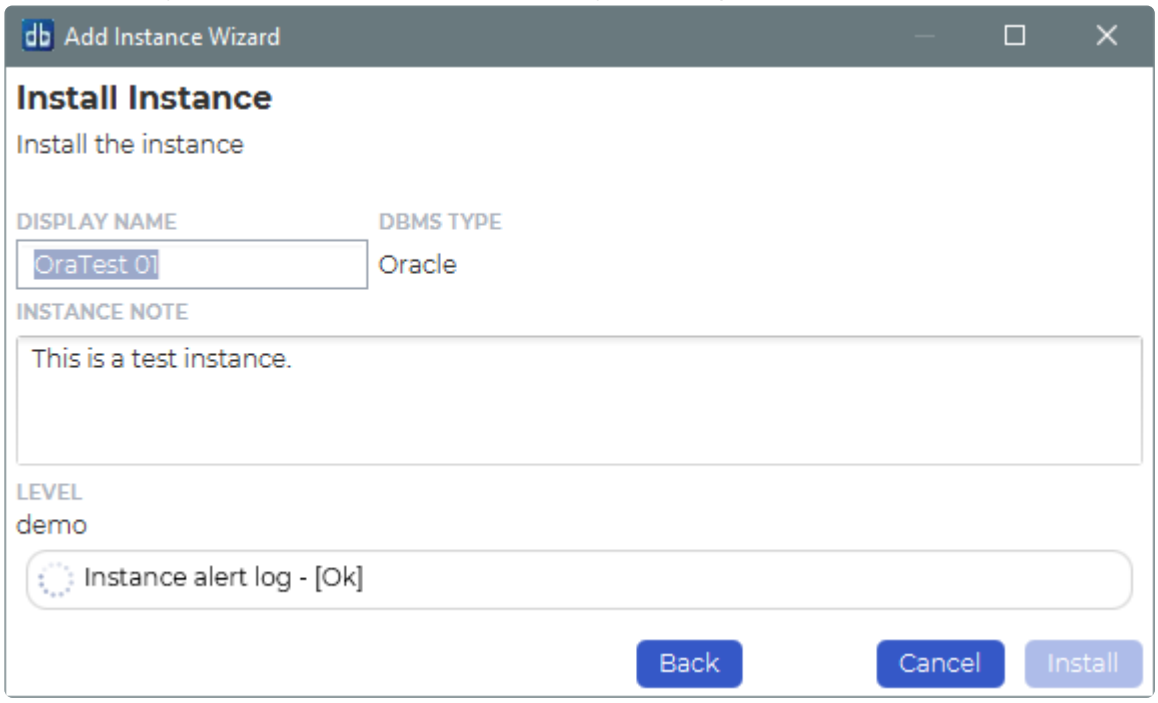
Instance Installation



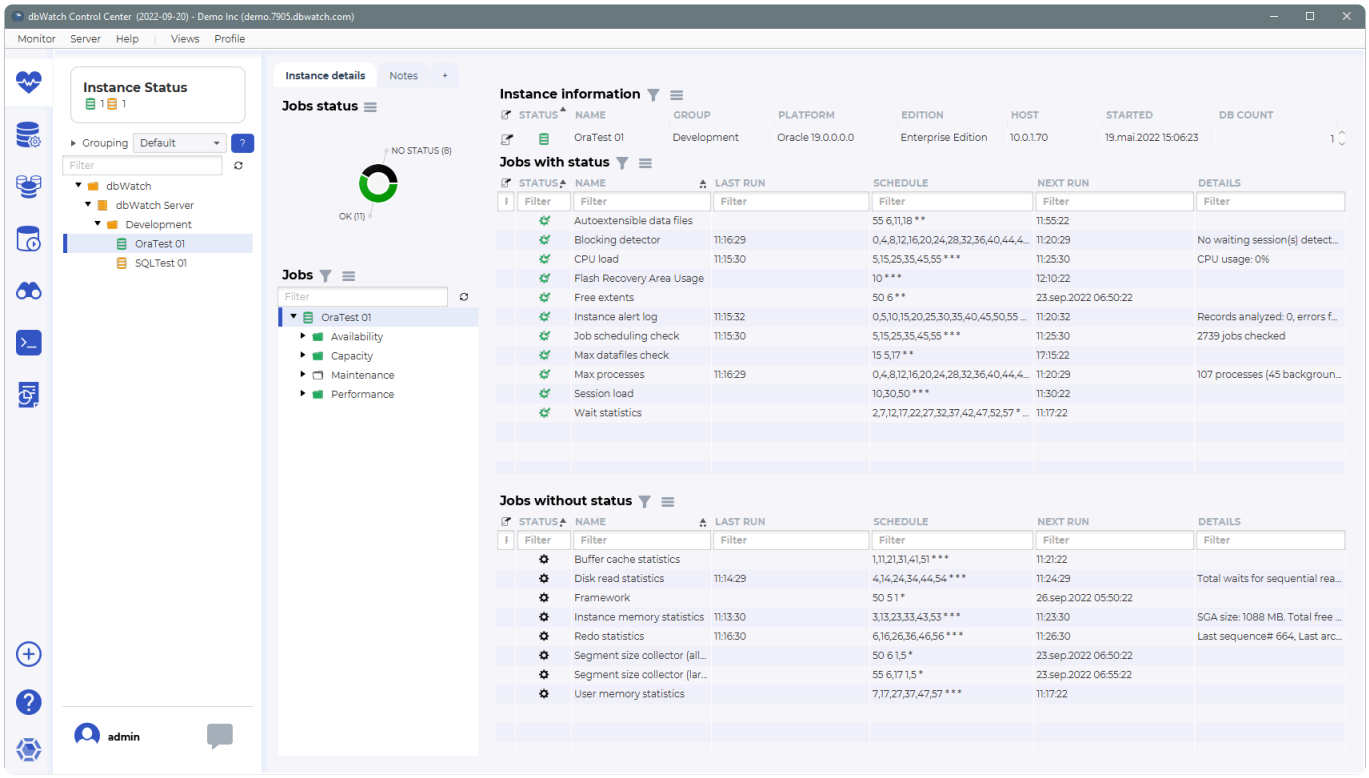
Specify a display name for this instance. You can also provide a description or any relevant information you want to remember about this instance.

Click Install to start installing.

Underneath you will see the status of various jobs being installed.



We've now successfully added the Oracle instance in our dbWatch monitor.

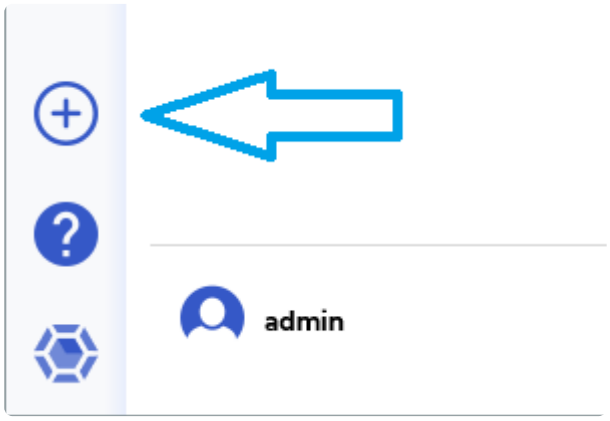


<< Adding SQLServer instances with kerberos / Adding a PostgreSQL Instance >>

Adding a PostgreSQL Instance

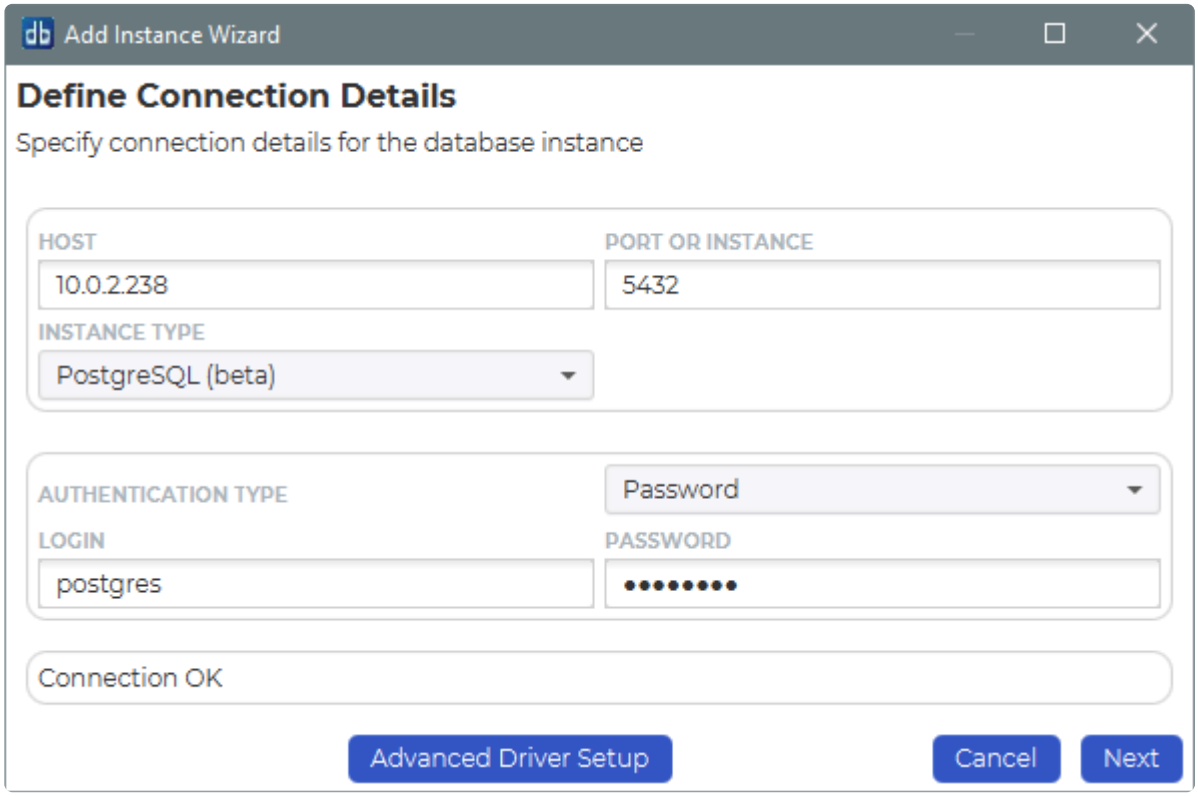
How to get here

To open the “Add Instance Wizard”, click on the “Plus (+)” sign on the bottom left of the dbWatch Client.



This will bring up the “Add Instance” wizard, which will guide you through the following steps.

Select Instance Type and Input Connection Details



Specify the host and port number.

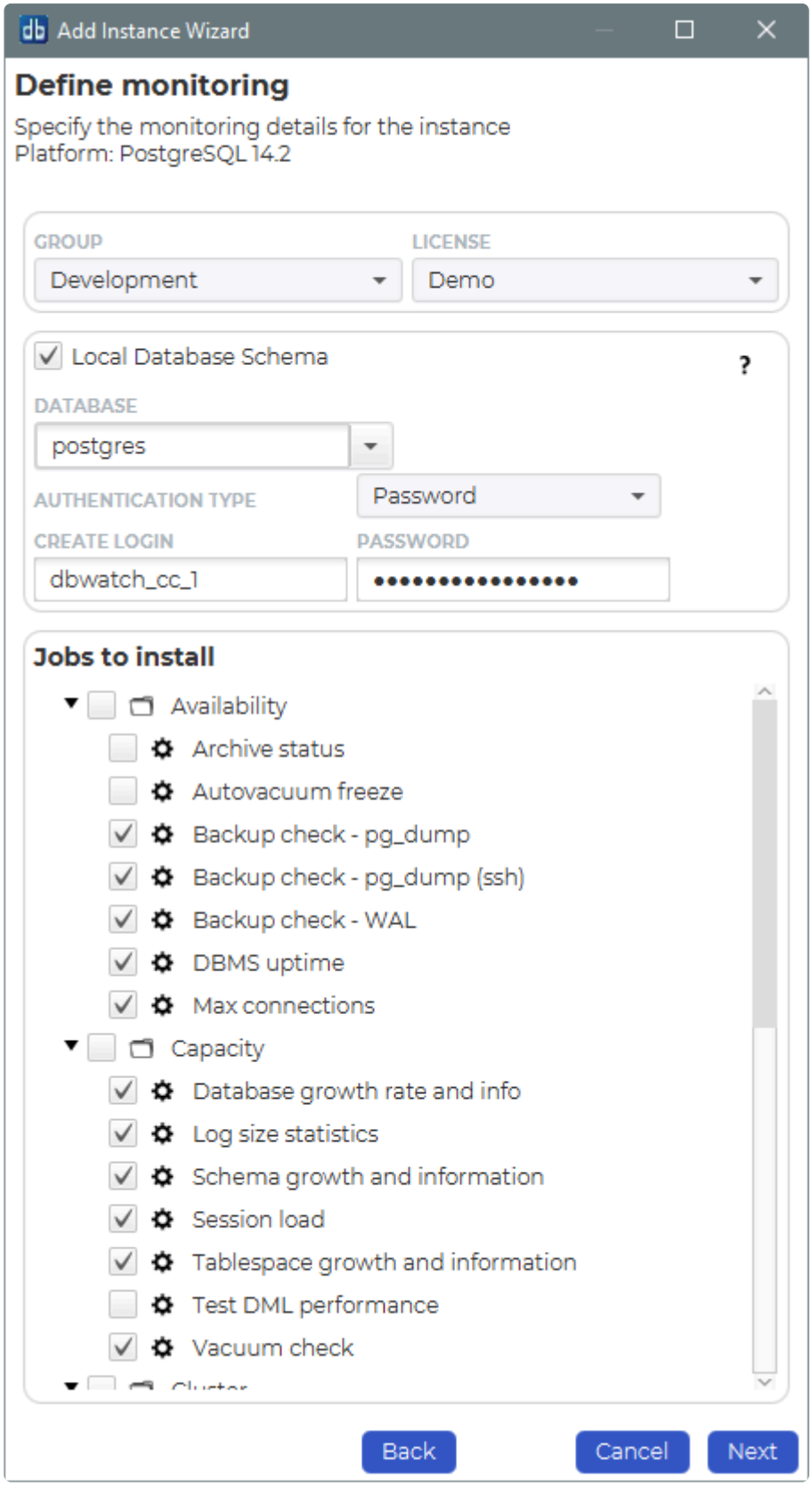
dbWatch will attempt to select the correct instance type based on the entered port number. If the instance type is wrong, select the correct one in the drop down.
The authentication types available will depend on the selected instance type.

Currently, PostgreSQL will only allow Password authentication.
You need to specify a login with a sysadmin role that will be used during the installation.

By default dbWatch will select the most apropiate JDBC driver for the database instance. It is possible to select a specific JDBC driver and set driver properties by clicking on “Advanced Driver Setup”. Read more.

Click “Next”.

Define Monitoring



Specify the group and license for the instance. Default groups are **Development**, **Test** and **Production**, but you can easily create and customize your own groups based on your requirements.
The licensing options available will depend on your license. The selected option will determine what monitoring jobs are available.

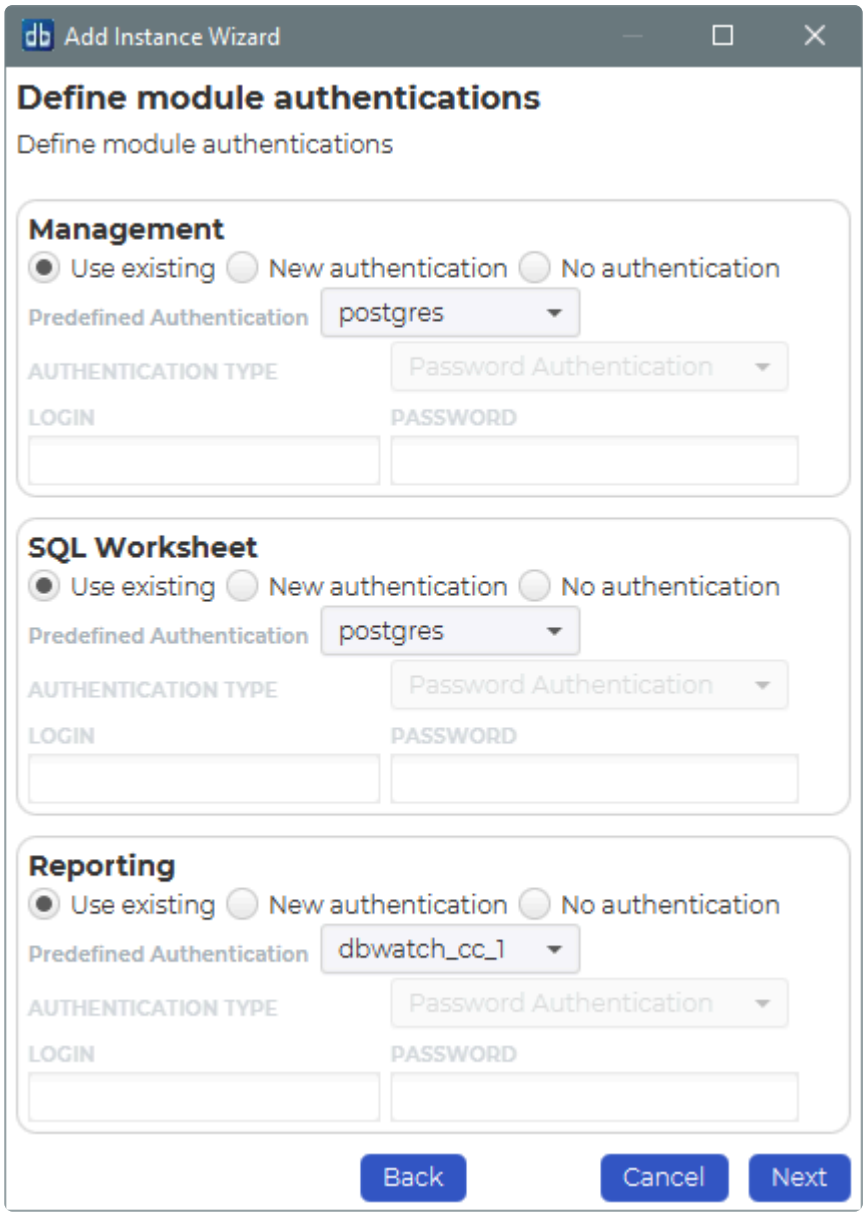
You can choose to register this instance with or without a local database schema. This choice will influence the available monitoring jobs. Some jobs require objects in the database (procedures, tables etc.), others do not. We recommend installing a database schema, as this will give the most monitoring options.

You can stick with the jobs that are selected by default or customize the selection as needed.

Click “Next”.

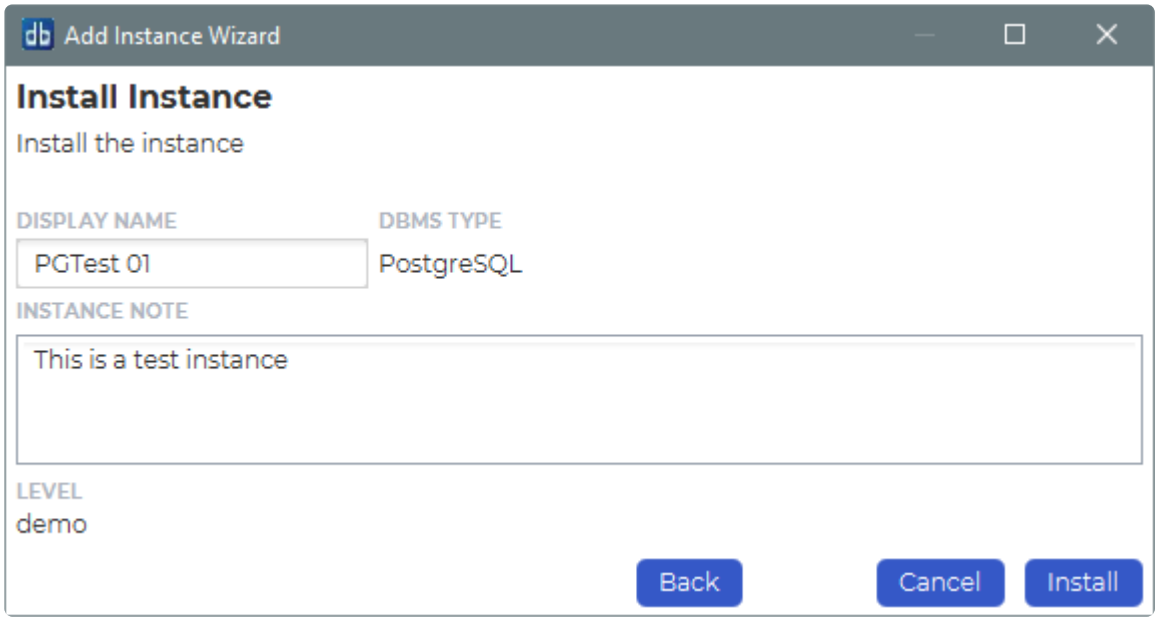
Define Module authentications

Within this window, you have an option to choose which credential to use for each module within dbWatch.
You can use the existing user or select a new user whether a password or OS authentication.



If you’ve finished specifying a credential for each module. Click Next.

Instance Installation



Specify a display name for this instance. You can also provide a description or any relevant information you want to remember about this instance.

Click Install to start installing.

Underneath you will see the status of various jobs being installed.

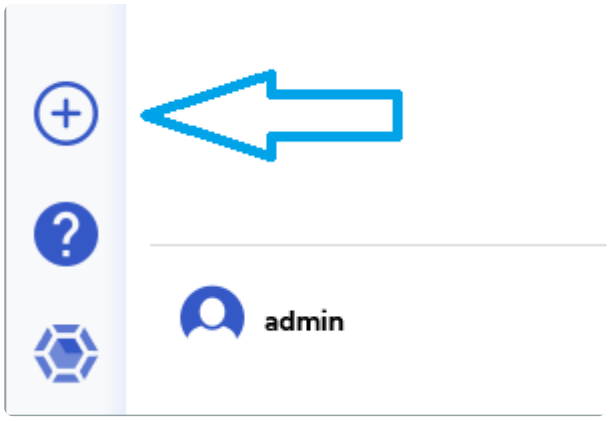
We've now successfully added the Postgres instance to our dbWatch monitor.

<< Adding an Oracle Instance / Adding a MySQL Instance >>

Adding a MySQL Instance

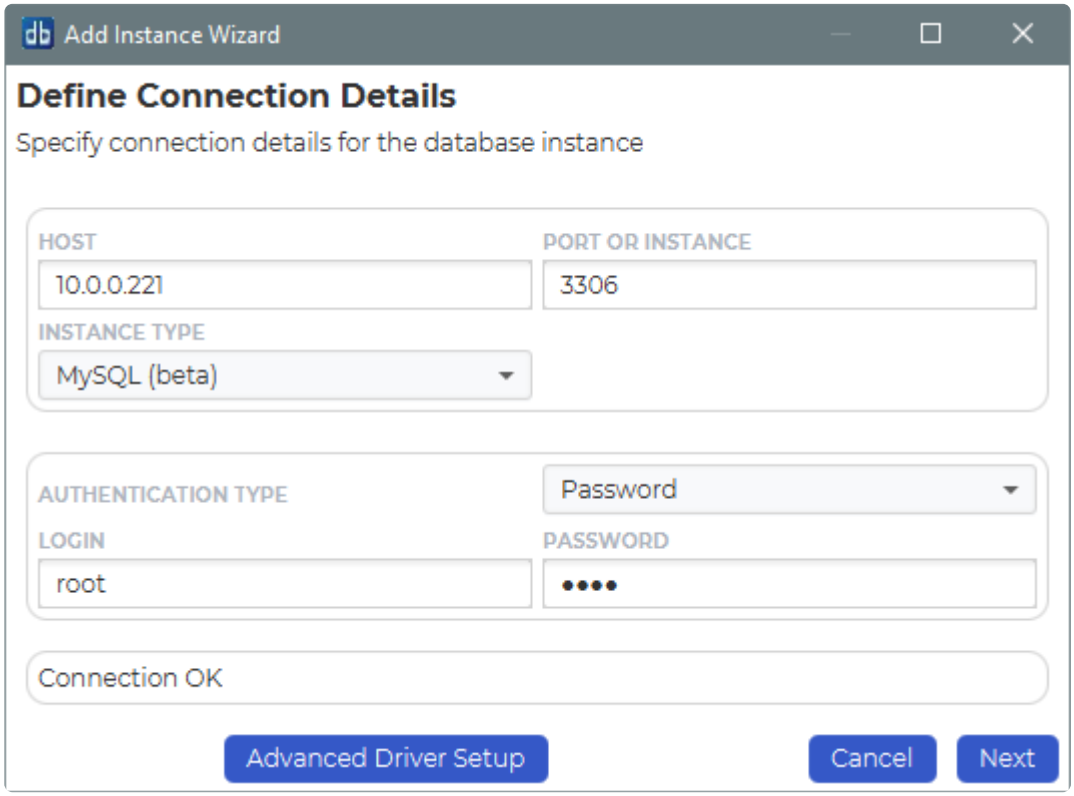
How to get here

To open the “Add Instance Wizard”, click on the “Plus (+)” sign on the bottom left of the dbWatch Client.



This will bring up the “Add Instance” wizard, which will guide you through the following steps.

Select Instance Type and Input Connection Details



The **Host** info is the hostname or ip address of the database instance.
The **Port** is the port number where the listener process is listening on.

dbWatch will attempt to select the correct instance type based on the entered port number. If the instance type is wrong, select the correct one in the drop down.

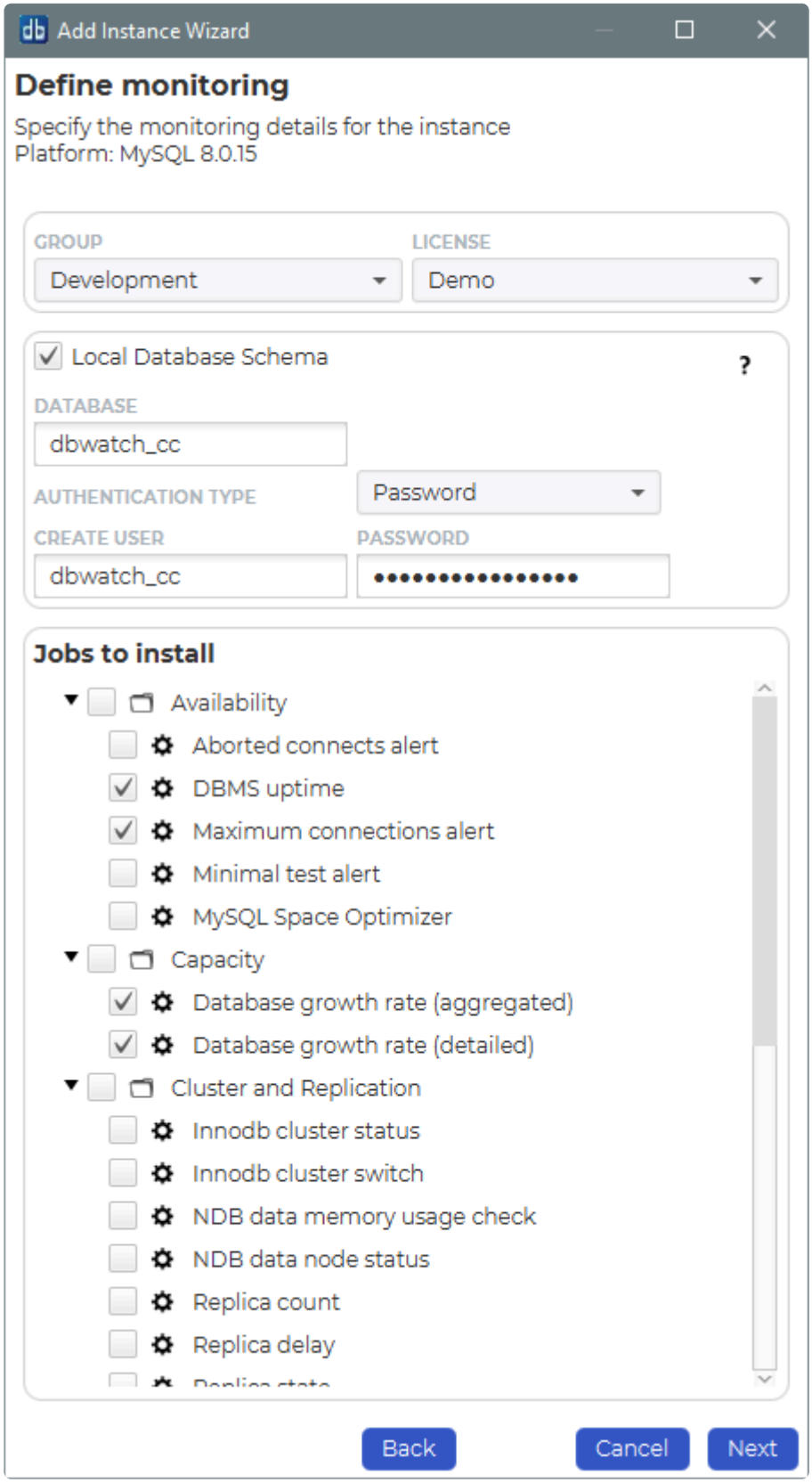
The authentication types available will depend on the selected instance type.

Currently, MySQL will only allows Password authentication.
You need to specify a username with admin privileges that will be used during the installation.

By default dbWatch will select the most apropiate JDBC driver for the database instance. It is possible to select a specific JDBC driver and set driver properties by clicking on “Advanced Driver Setup”. Read more.

Click “Next”.

Select Instance Type and Connection Details



Specify the group and license for the instance. Default groups are **Development**, **Test** and **Production**, but you can easily create and customize your own groups based on your requirements.
The licensing options available will depend on your license. The selected option will determine what monitoring jobs are available.

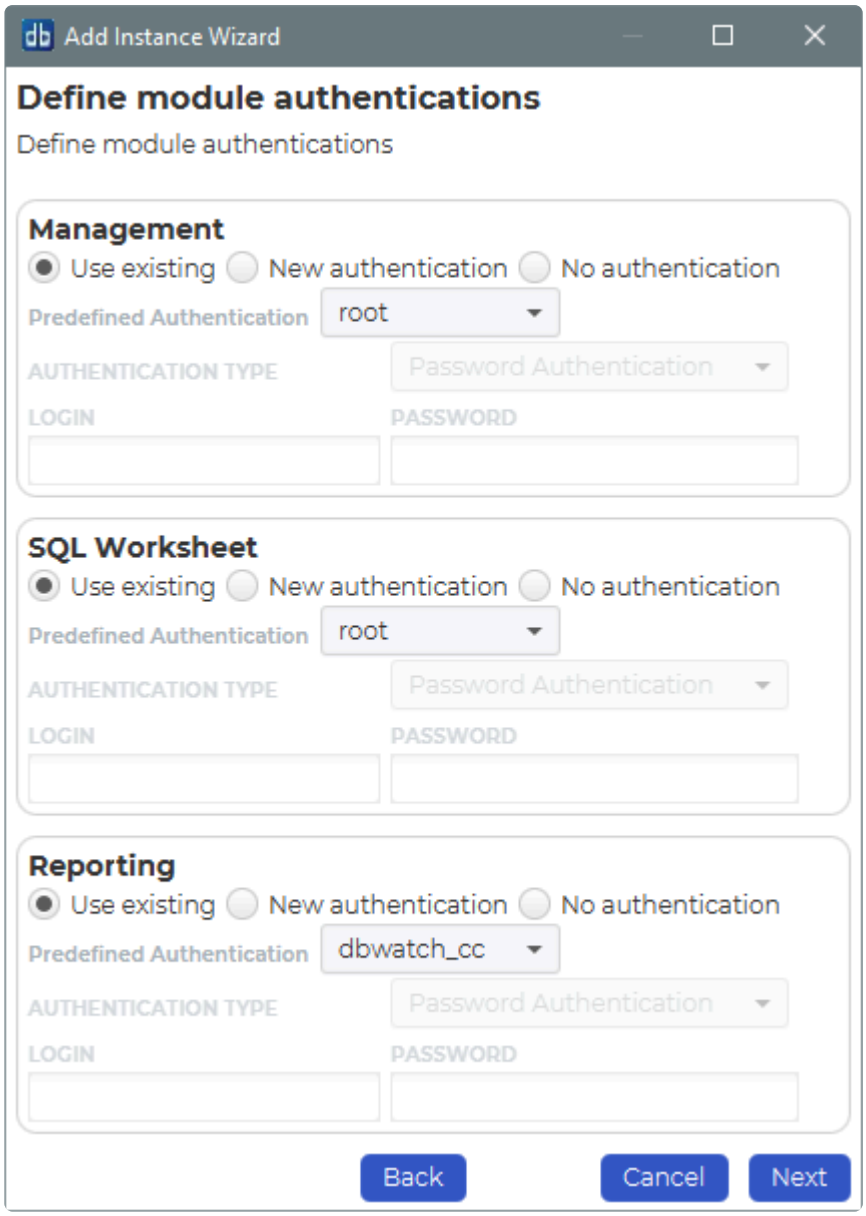
You can choose to register this instance with or without a local database schema. This choice will influence the available monitoring jobs. Some jobs require objects in the database (procedures, tables etc.), others do not. We recommend installing a database schema, as this will give the most monitoring options.

You can stick with the jobs that are selected by default or customize the selection as needed.

Click “Next”.

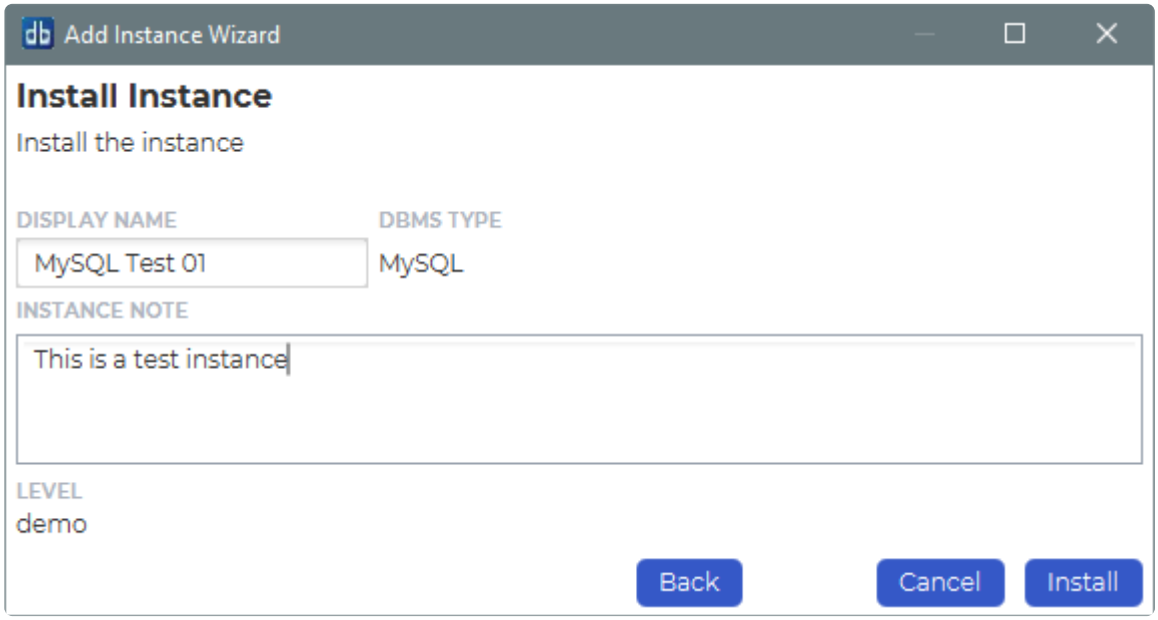
Define Module authentications

Within this window you have an option to choose which credential to use for each module within dbWatch.
You can use the existing user or select a new user whether password.



If you’ve finished specifying a credential for each module. Click Next.

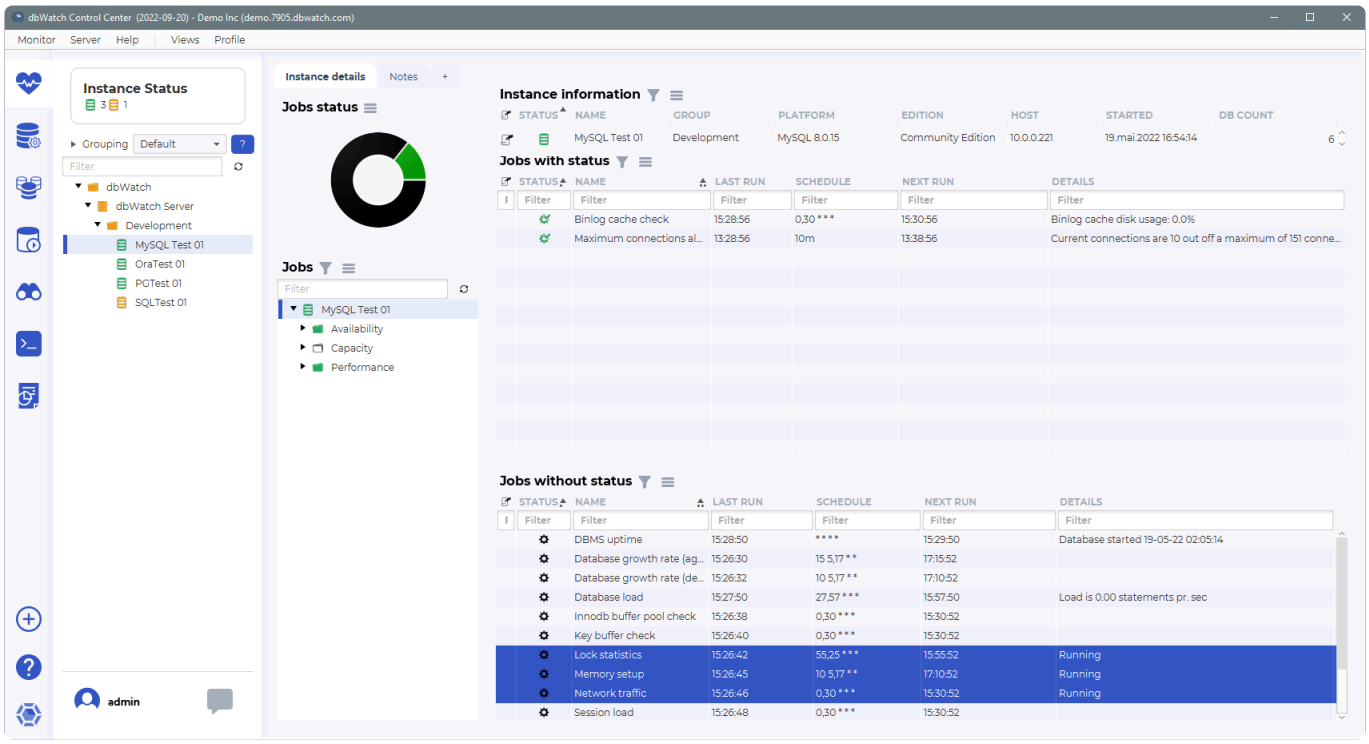
Instance Installation



Specify a display name for this instance. You can also provide a description or any relevant information you want to remember about this instance.

Click Install to start installing.

Underneath you will see the status of various jobs being installed.



We've now successfully added the MySQL instance in our dbWatch monitor.

If dbWatch cannot install the software due to a Super Privilege Error, see Super Privilege Error for steps in solving it.

[<< Adding a PostgreSQL Instance] / [Bulk installing instances >>]

Bulk installing instances

CSV File Format

You can add instances in bulk. The prerequisite is to have a CSV file with the following format:

```
"action:instance-import"
"dbms-type:sqlserver","given-name:SQLTEST001","host:local.dbwtest1.com","port:1433","admuser:sa","admpassword:Password1","group:Test","engine:true","service:demo"
"dbms-type:sqlserver","given-name:SQLTEST002","host:local.dbwtest2.com","port:1433","admuser:sa","admpassword:Password2","group:Test","engine:true","service:production_multiplatform"
"dbms-type:oracle","given-name:ORATEST001","host:local.dbwtest3.com","port:1521","sid:ORCL","admuser:sys as sysdba","admpassword:Password3","group:Test","engine:true","service:demo"
"dbms-type:mariadb","given-name:iridium","host:10.0.0.87","port:3306","group:mariadb","engine:true","adm-auth-mode:pwd","admuser:root","admpassword:9d500ec80e82789489ca770bd2d8b436","adm-all-global:true","service:production_multiplatform;all_performance_package"
"dbms-type:sqlserver","given-name:SQLPROD-001","host:SQLPROD-001.SECURE.DBWATCH.COM","port:1433","admuser:svc_dbwatch@SECURE.DBWATCH.COM","admpassword:SuperSecure001!","group:PROD","engine:true","service:demo","adm-auth-mode:kerberos","jdbc-driver:driver:com.microsoft.sqlserver:mssql-jdbc:9.4.0.jre8@connector:no.dbwatch.connectors:sqlserver"
```

Let’s explain the tags more with the following table:

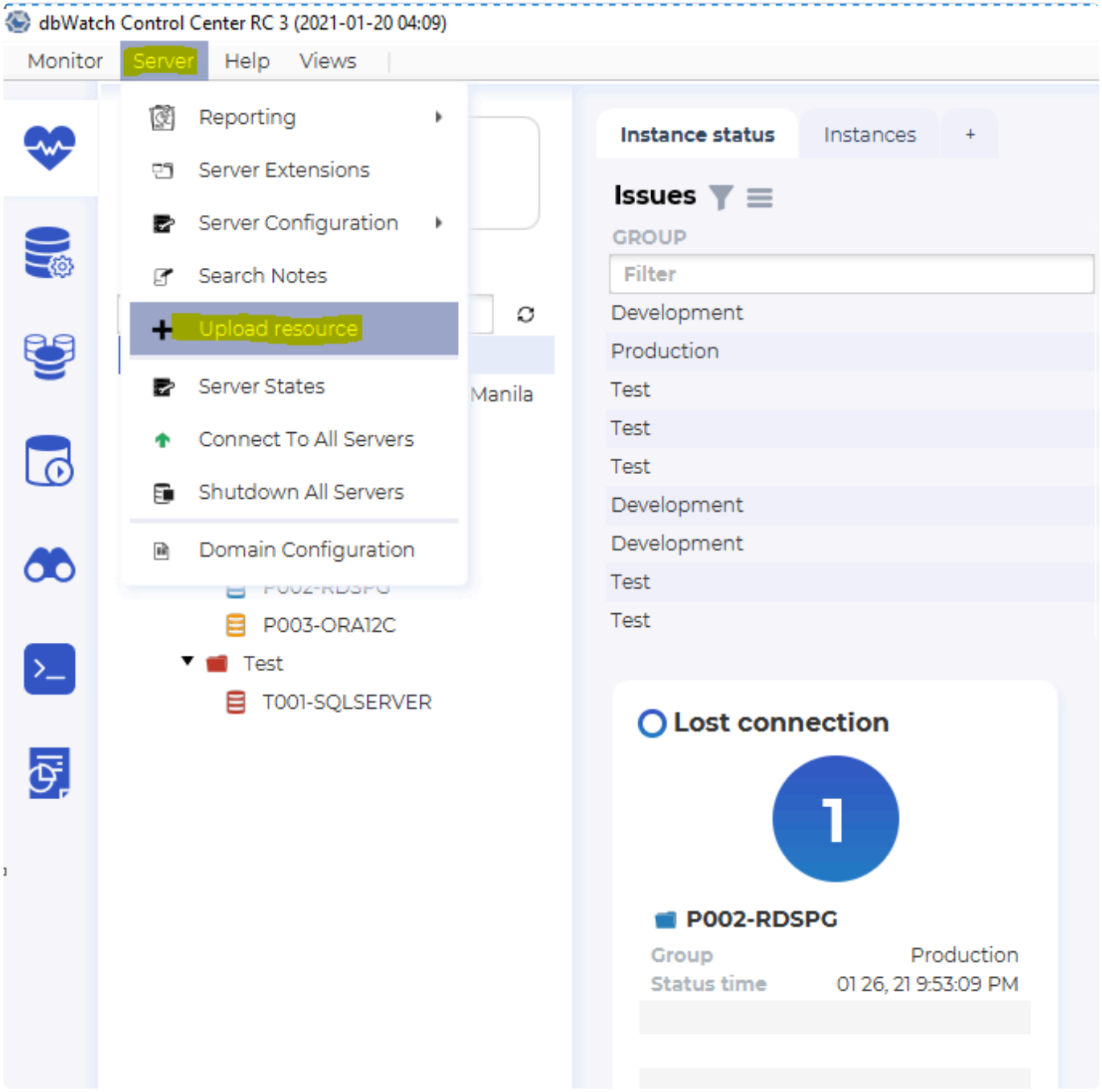
Tag	*Is mandatory*	*Description*
dbms-type	yes	The database type (sqlserver, oracle, postgres, mysql, sybase)
given-name	yes	The display name for the instance.
host	yes	Host IP
sid	only on Oracle	Specify Oracle SID
port	yes	Port number
admuser	yes	This user will be set as the properties user. If ‘engine’ is set to true, this user will be used to create the dbwatch schema
admpassword	yes	The password for the admin user

adm-auth-mode	no	What type of authentication is used, examples “adm-auth-mode:pwd” or “adm-auth-mode:kerberos”
adm-all-global	no	Sets management / administration user for all users with access to an instance to the admuser credentials. Default false, values true/false. Example “adm-all-global:true”
group	yes	The group to place the instance in. If ‘engine’ is set to true, the default values used during schema creation are from this group.
subgroup	no	The sub group to place the instance in.
engine	yes	If a dbwatch schema should be created (true, false).
engine-database	no	The name of the dbwatch database. Use if engine=true, and you want another database name than is specified in the group.
jobtemplate	no	Comma separated list of job templates.
jobtemplatemode	no	The job template mode (“keep others”, “disable others”, “uninstall others”).
service	yes	The licensing level to set for this instance. This can be a **semicolon** separated list of licenses to apply, lowercased. It is the same string as in the id in your license file. Examples: production_multiplatform, demo;production_sqlserver
jdbc-driver	no	It is possible to specify the jdbc driver to use, example: “jdbc-driver:driver:com.microsoft.sqlserver:mssql-jdbc:9.4.0.jre8@connector:no.dbwatch.connectors:sqlserver”

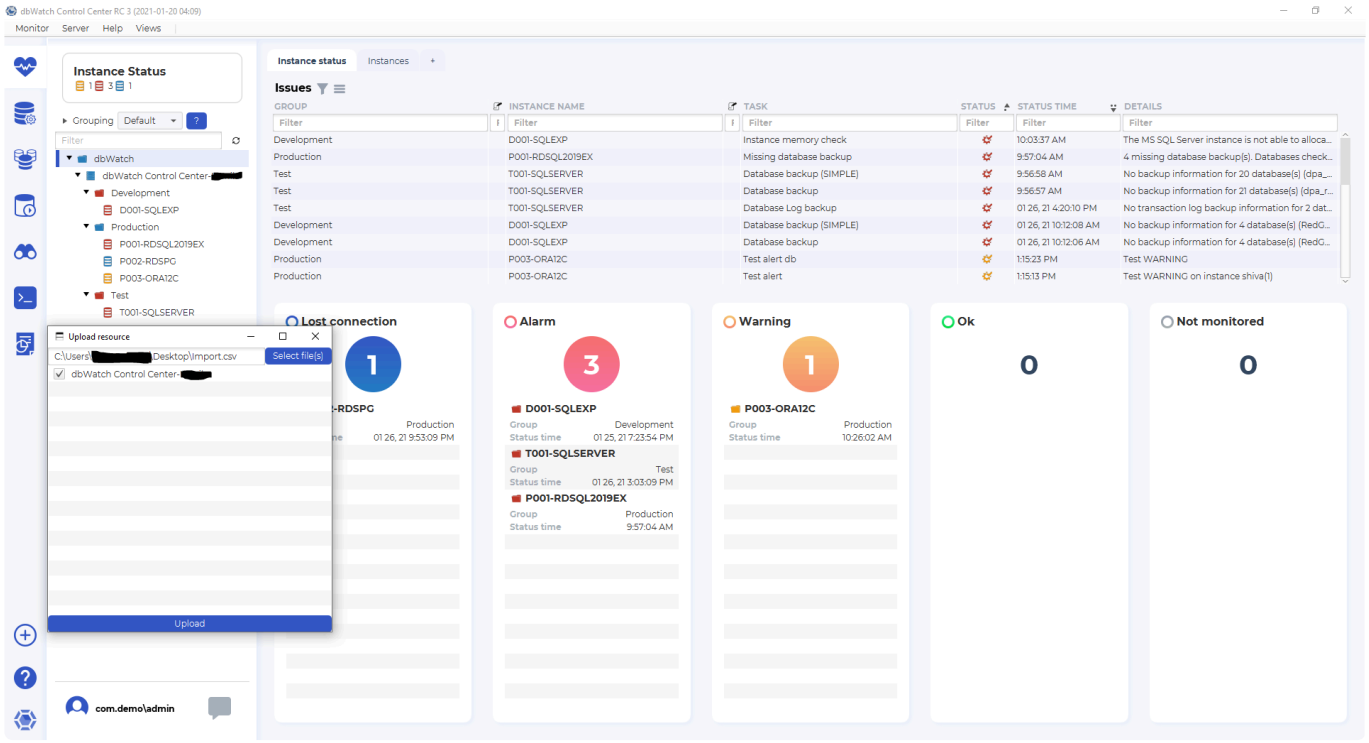
> ! Please take note that you should be using a user that has enough privileges as a super user or admin user. Otherwise, installing the instance may lead to problems in connecting and installing jobs.

Uploading Instance File

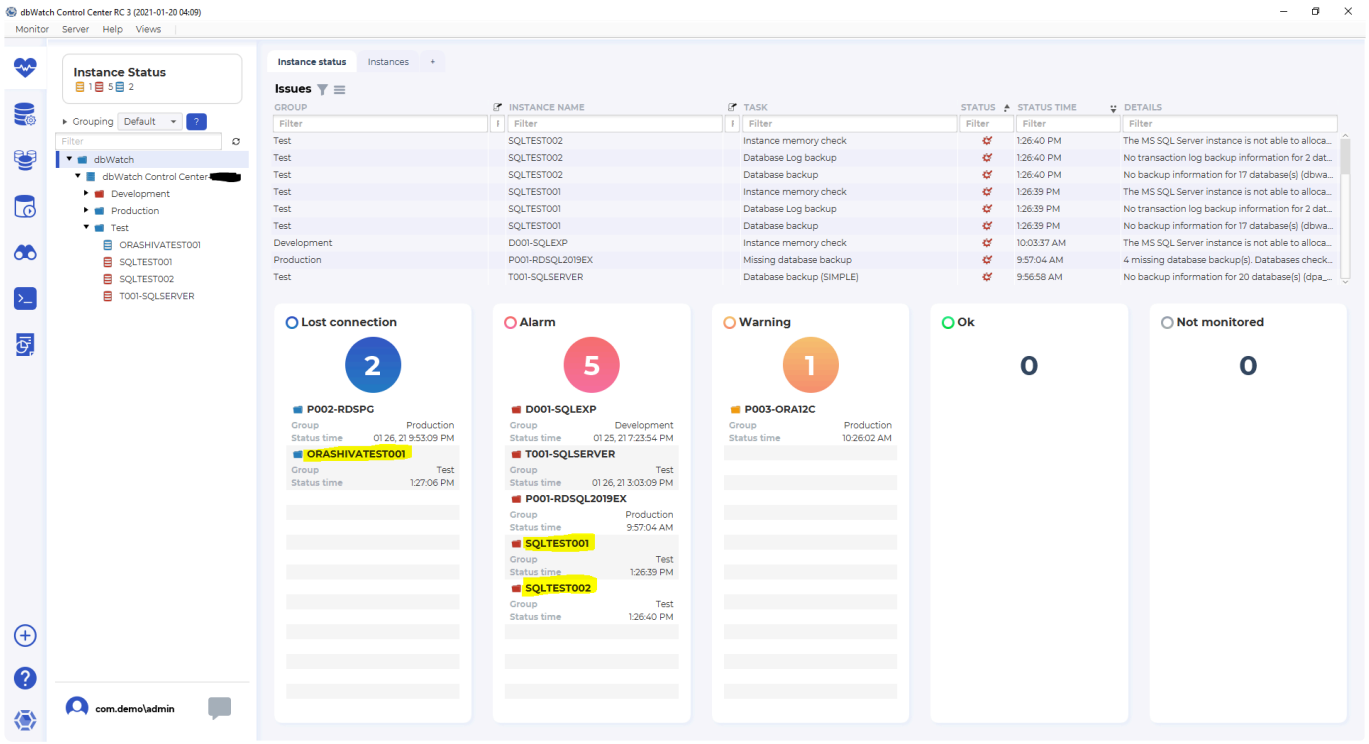
Open ‘Server’->‘Upload resource’ menu.



A window will appear where you can upload resource files. Click on “Select Files” then choose the CSV file for bulk installation.
Afterwards, click on “Upload”.



After successfully uploading the Import.csv file.
You will see the 3(highlighted in yellow) additional instances that we've added.

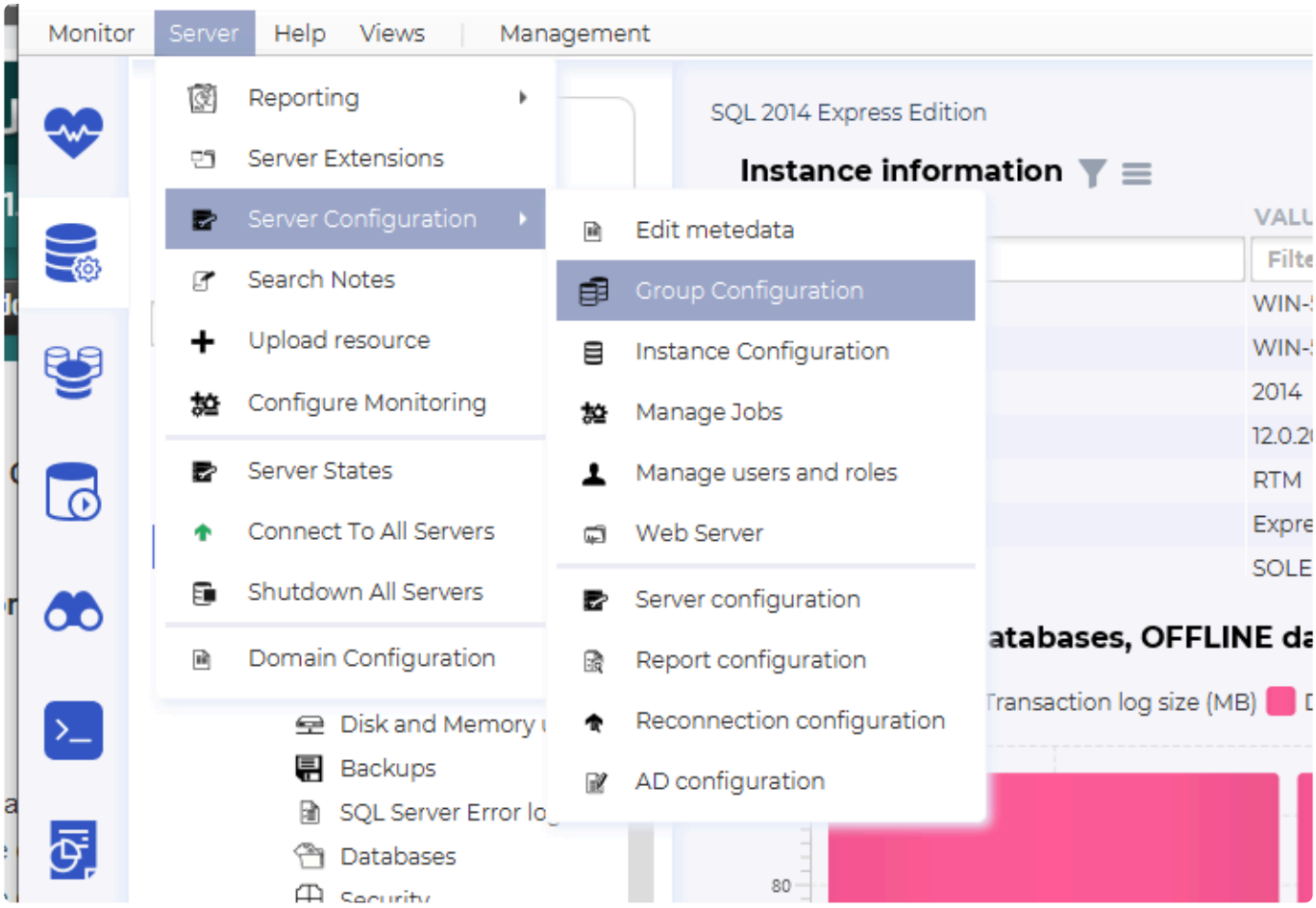


<< Adding a MySQL Instance / Groups Configuration >>

Groups Configuration

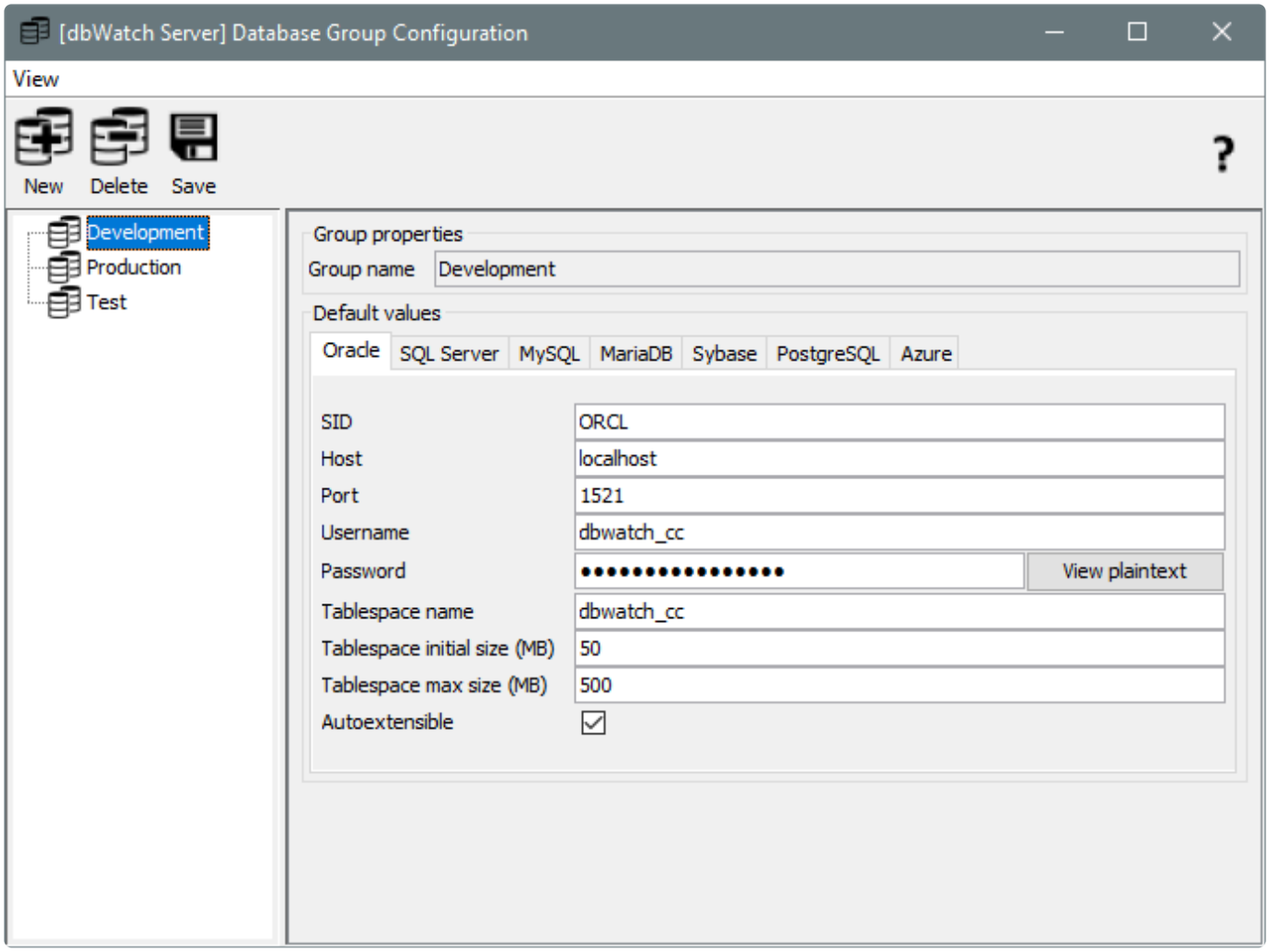
How to get here

You can access the group configurations going to **Server —> Server Configuration —> Group Configuration**.



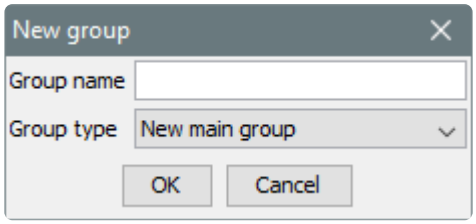
Overview

The Group Configuration view consists of two main parts. On the left side you can see a tree structure showing the groups, and on the right side you see the configured values for the selected group. These values are the defaults that will be used when you add a new instance to this group.



Adding a new group

To add a new group, click on the **New** icon on the toolbar. The dialog below will then appear.



Here you type in a name for the group and also choose if this should be a new main group, or if it should be a sub group of one of the existing groups.

Delete a group

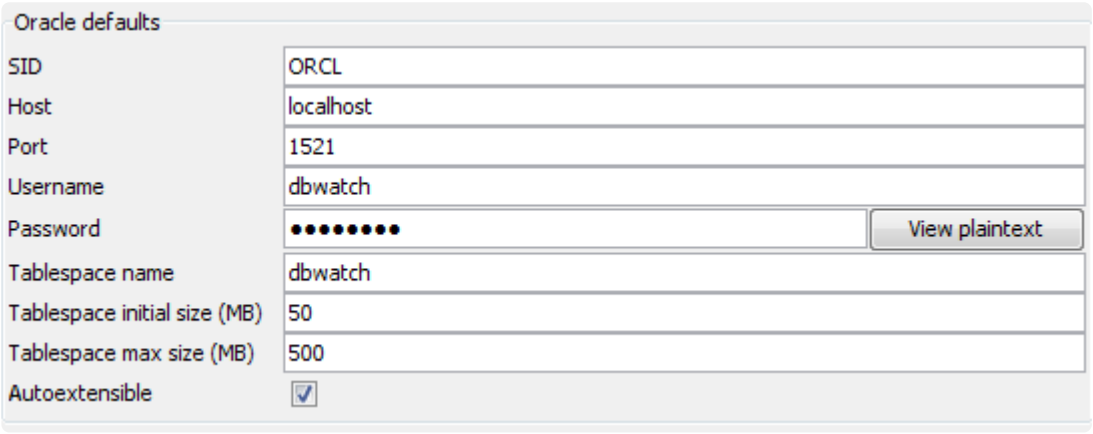
To delete a group, select it in the tree and click on the **Delete** icon on the toolbar.

A group cannot be deleted as long as it contains instances, therefore you have to move any instances to another group before deleting it.

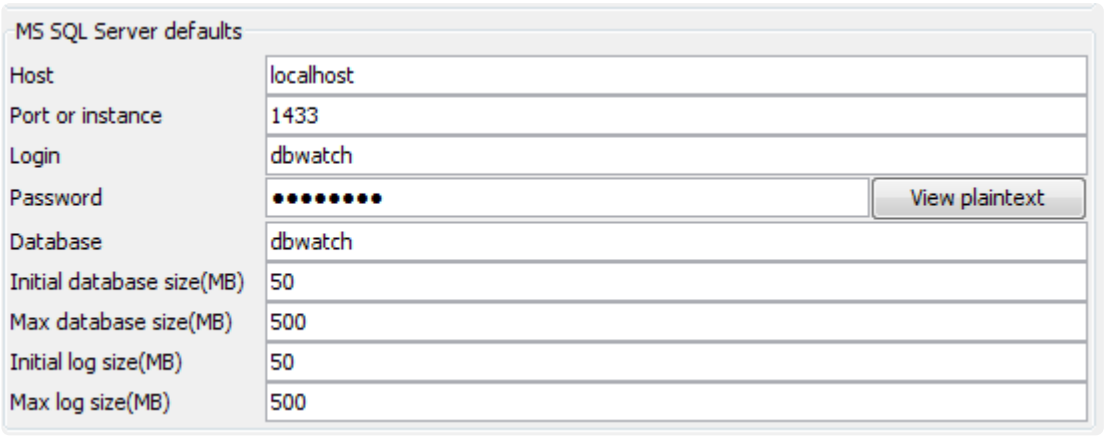
Default values

Each group has a set of default values for each DBMS type. Subgroups inherit the default values from their parent group.

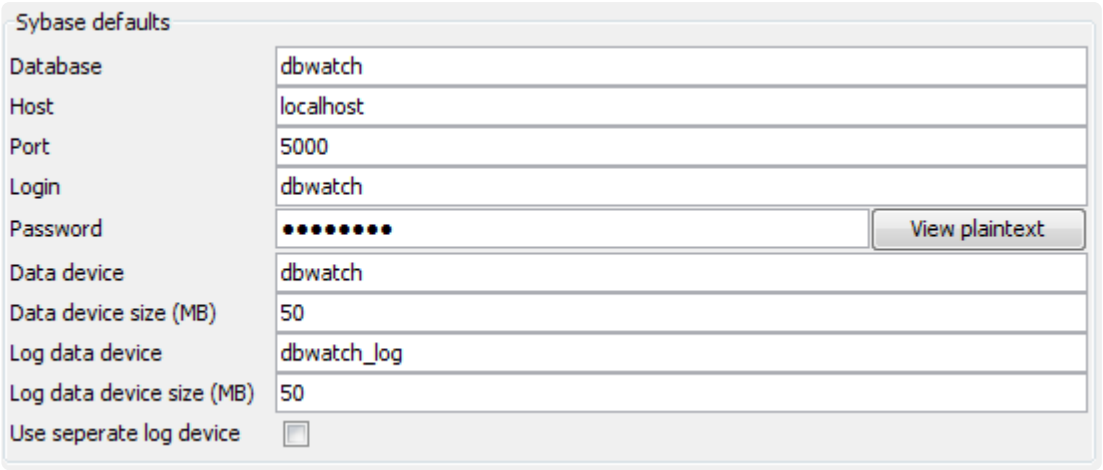
Oracle defaults



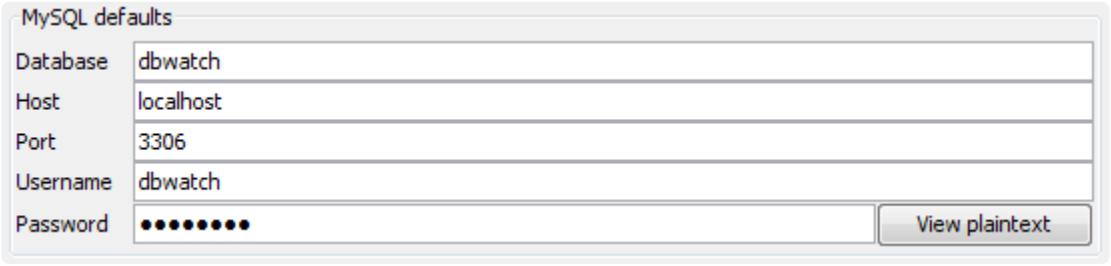
SQLServer defaults



Sybase defaults



MySQL defaults



PostgreSQL defaults

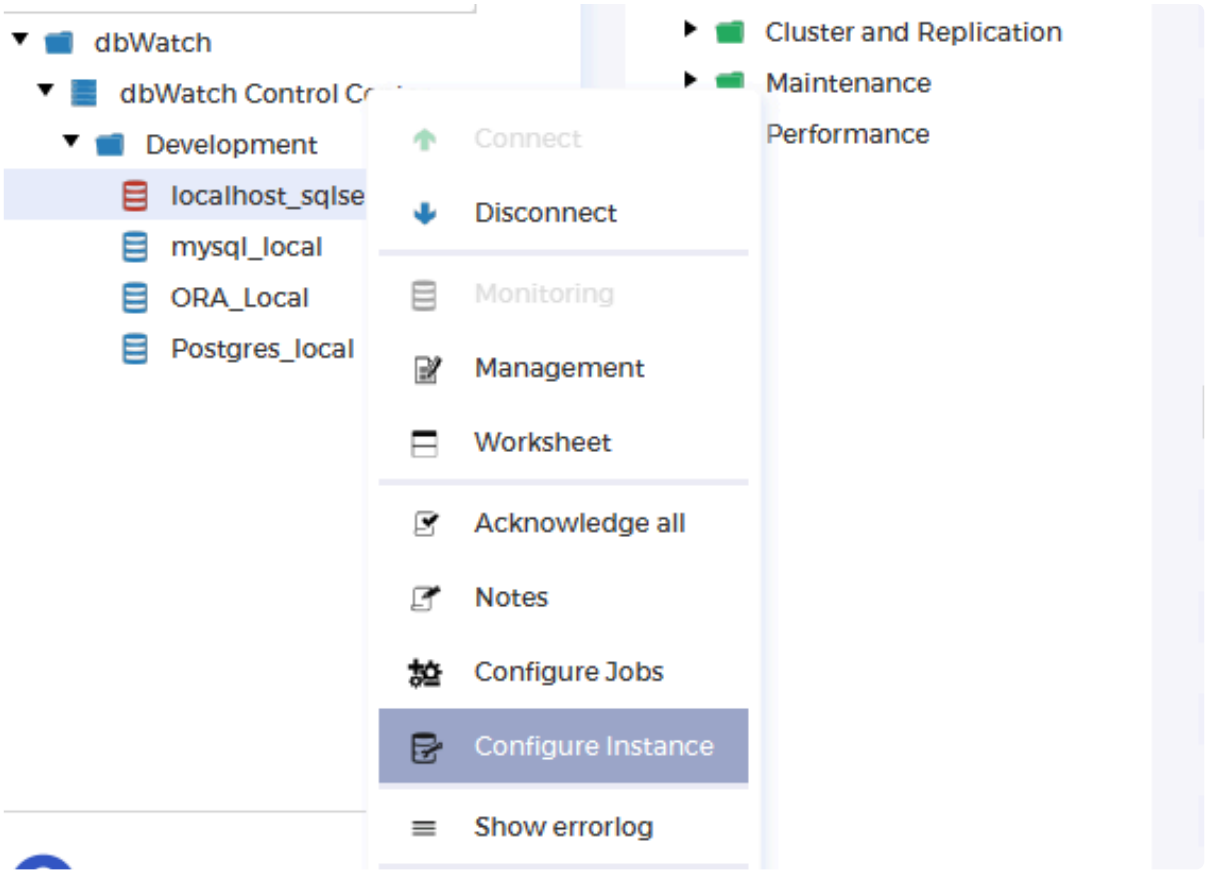
[<< Bulk installing instances] / [Changing Instance Group >>]

Changing Instance Group

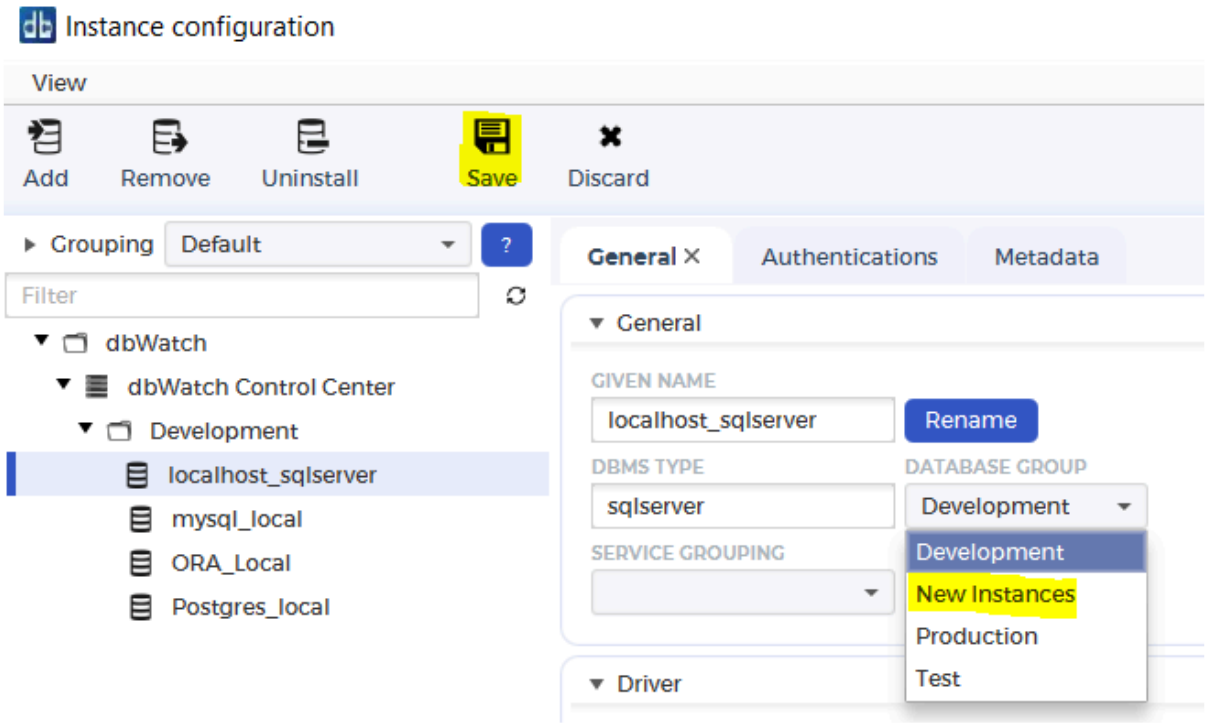
Changing a group can be done when you created an instance under a default group name and change it to another corresponding group. To change the groups base on your newly created group, you can check Group Configuration.

Changing Group of an Instance

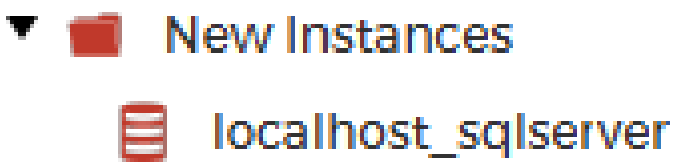
Once a group has been created, right-click on the instance and select “Instance Configuration”.



Proceed with the drop-down under the label “Database Group” and select your newly created instance group. For this example, we’ll be selecting “New Instances”. Click Save.



You should be able to see the instance be moved to the newly created instance group in the navigation tree.



<< Groups Configuration / Service groups >>

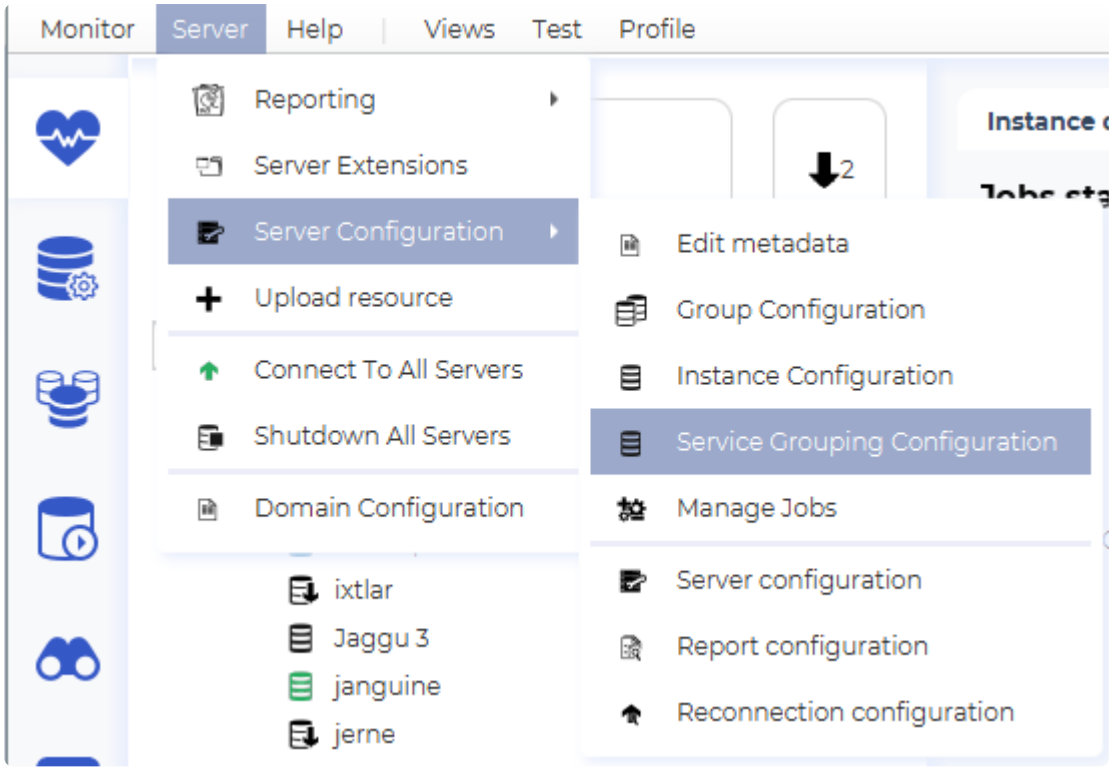
Service groups

In dbWatch you can organize instances in servicegroups

Instances placed in the same servicegroup should provide a common service of some sort. For example this can be instances belonging to the same cluster. Servicegroups may also be placed inside other servicegroups.

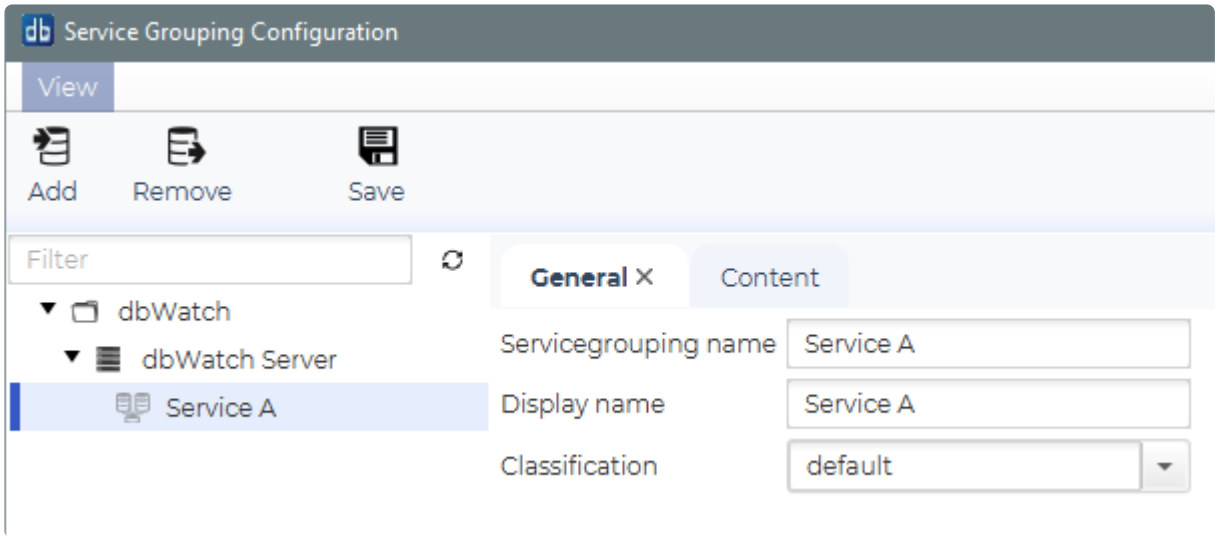
How to get here

You can access the service group configurations by going to **Server —> Server Configuration —> Service Group Configuration**.



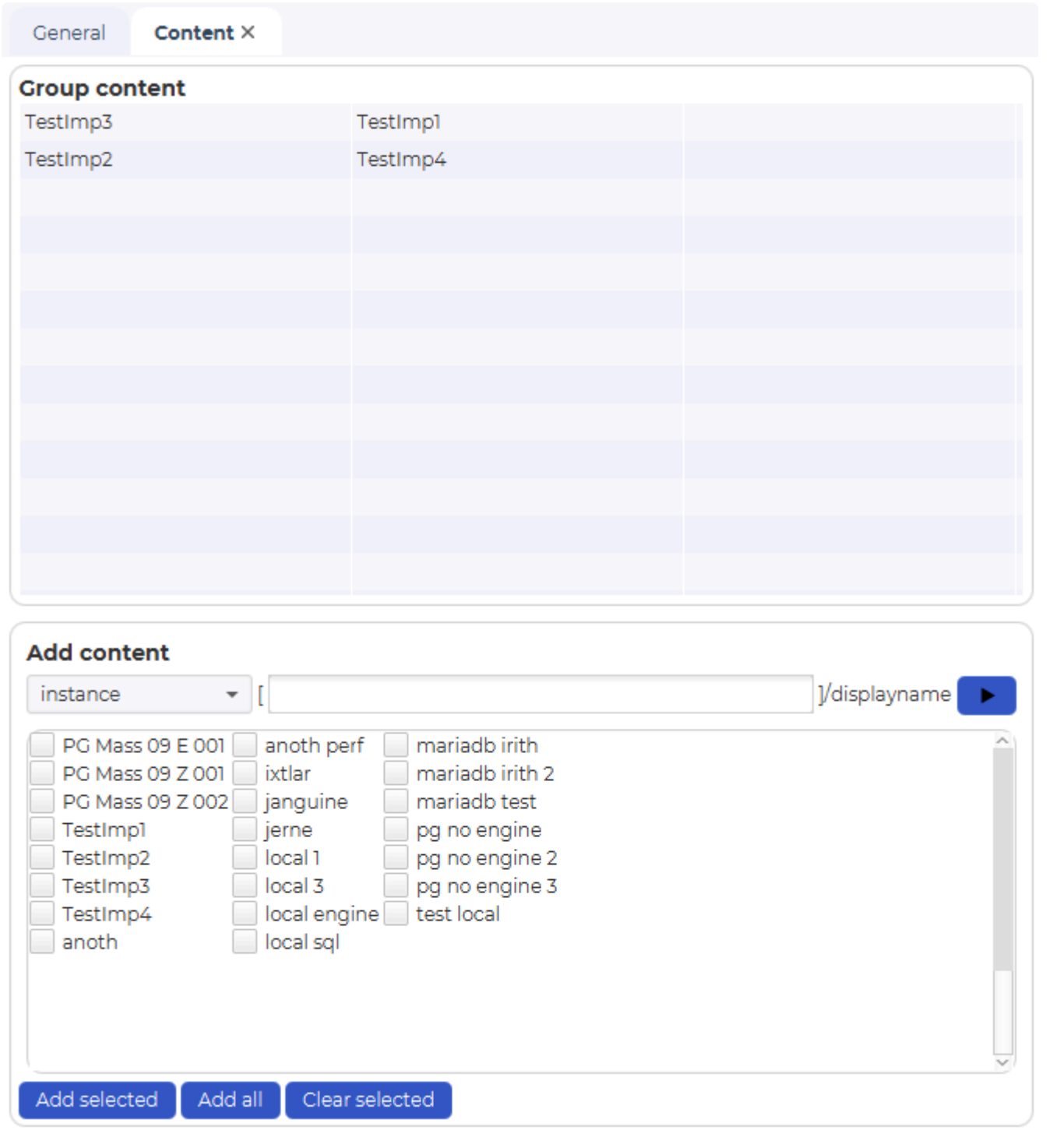
Overview

The Service Group Configuration view consists of two main parts. On the left side you can see a tree structure showing the service groups, and on the right side you see details about the selected group.



When you select a service group, the details on the right side consists of two panels, General and Content.

General lets you configure the displayname and classification for the Service Grouping.
Content lets you define the content of the Service Group (database instances and/or other Service Groups)

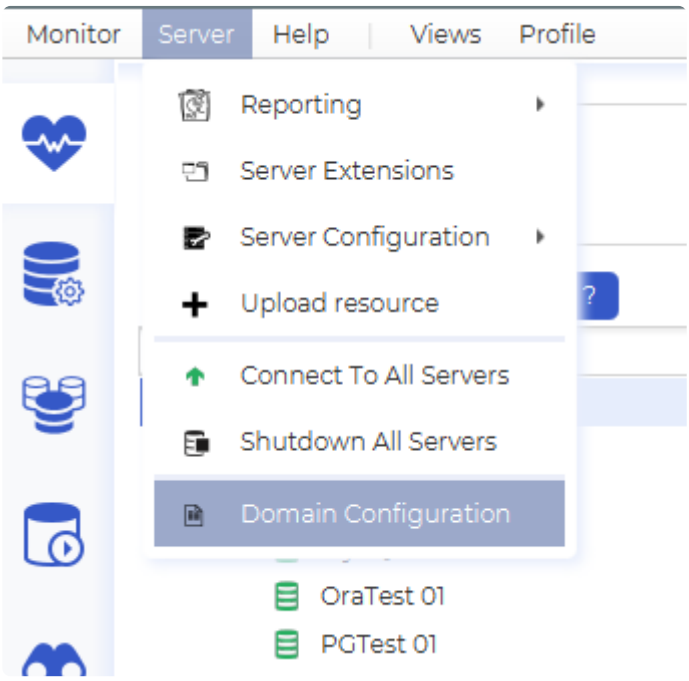


<< Changing Instance Group / Domain Configuration >>

Domain Configuration

How to get here

The Domain Configuration can be accessed by going to **Server > Domain Configuration**.

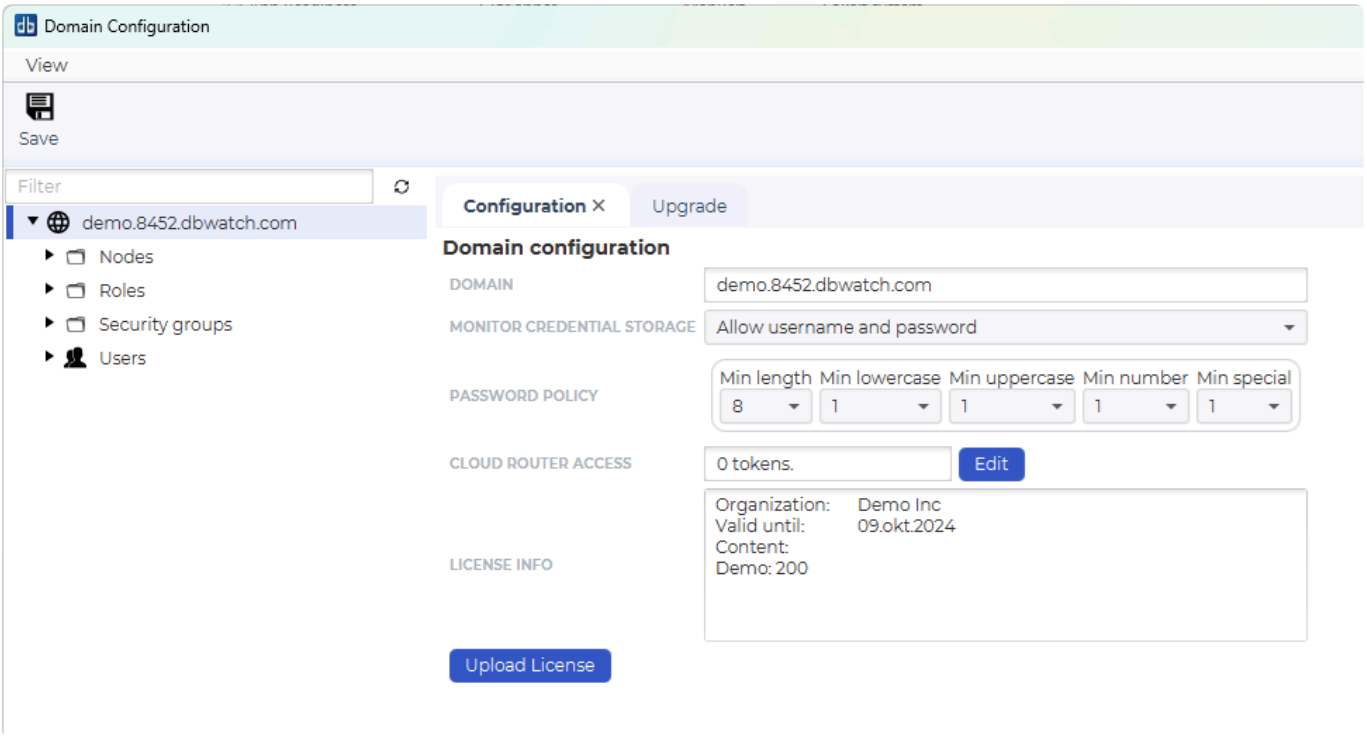


Overview

The Domain Configuration dialog lets you configure domain-wide settings.

The concepts listed are:

- **Domain** – Lets you configure domain settings.
- **Nodes** – Lets you control what dbWatch Servers are allowed to join the domain.
- **Roles** – Lets you configure the Roles and privileges that are availble to users.
- **Users** – Lets you configure the users.
- **Security groups** – Lets you group the instances that you wish to secure access to.

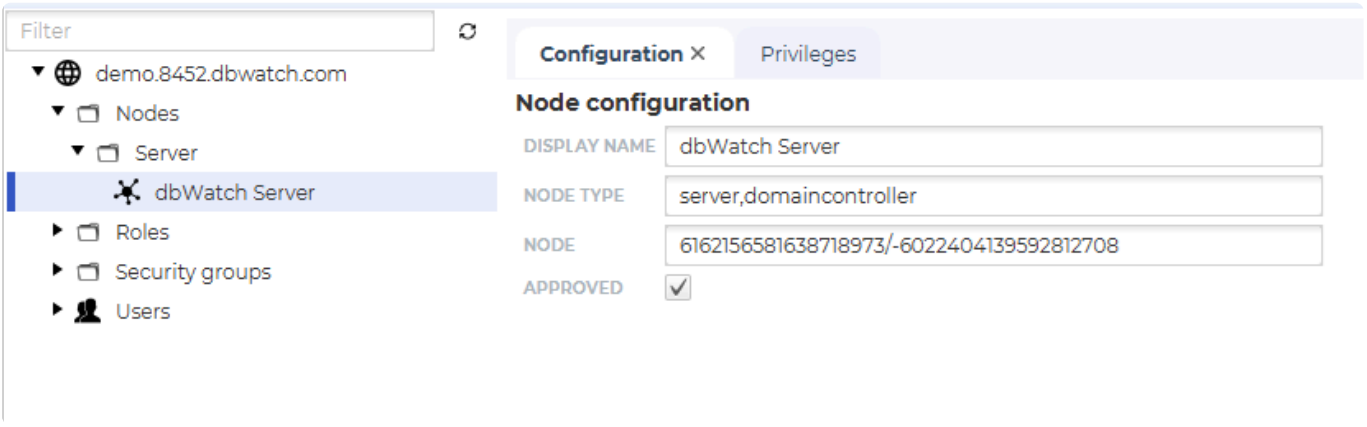


Domain

In the Domain configuration section you can set configuration settings for the domain as a whole.

Monitor Credential Storage If the Monitors are allowed to remember credentials between restarts.
Password Policy Rules that the user passwords must adhere to.
Cloud Router Access Access tokens that users connecting through a Cloud Router must provide.
License info Summary of the current license.
Upload license Set a new license.

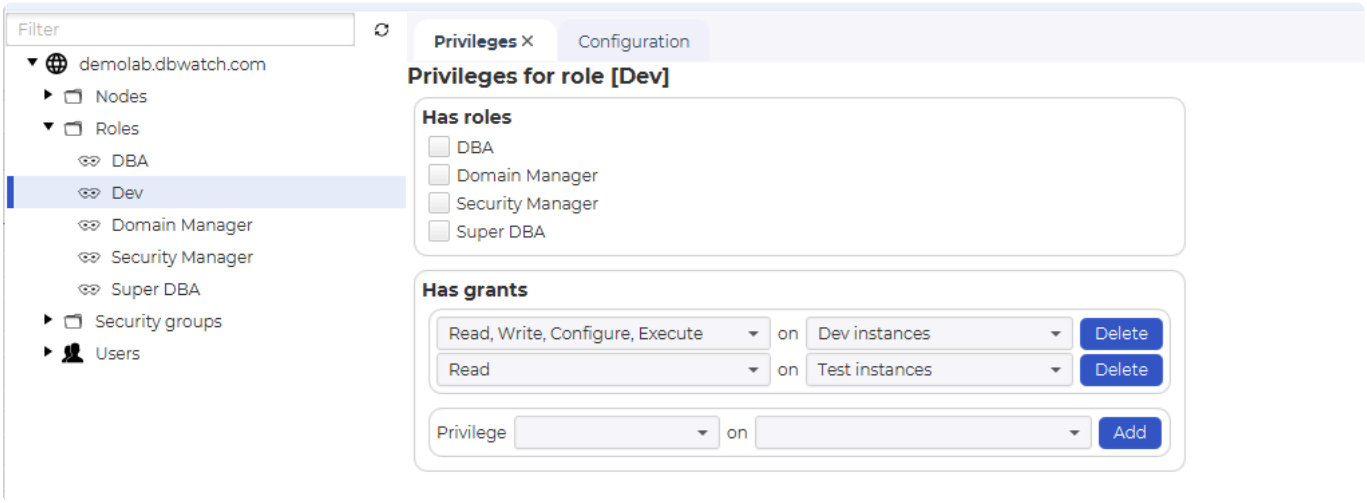
Nodes



In the node configuration there are 2 editable fileds. The displayname and the approved checkbox. This is were you approve new Servers that want to join the domain.
In addition you can see the type (currently server and/or domaincontroller) and id for the node (used in the internal network routing).

Look at the chapter for managing nodes for more detail.

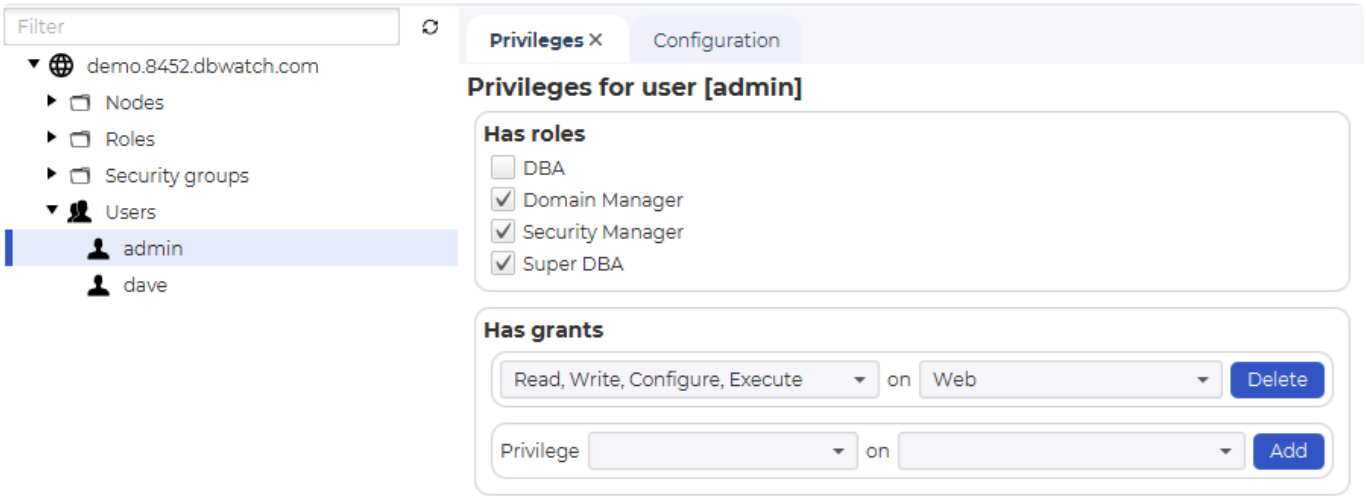
Roles



In the roles configuration you can add, delete and configure roles.

Look at the chapter for managing roles for more detail.

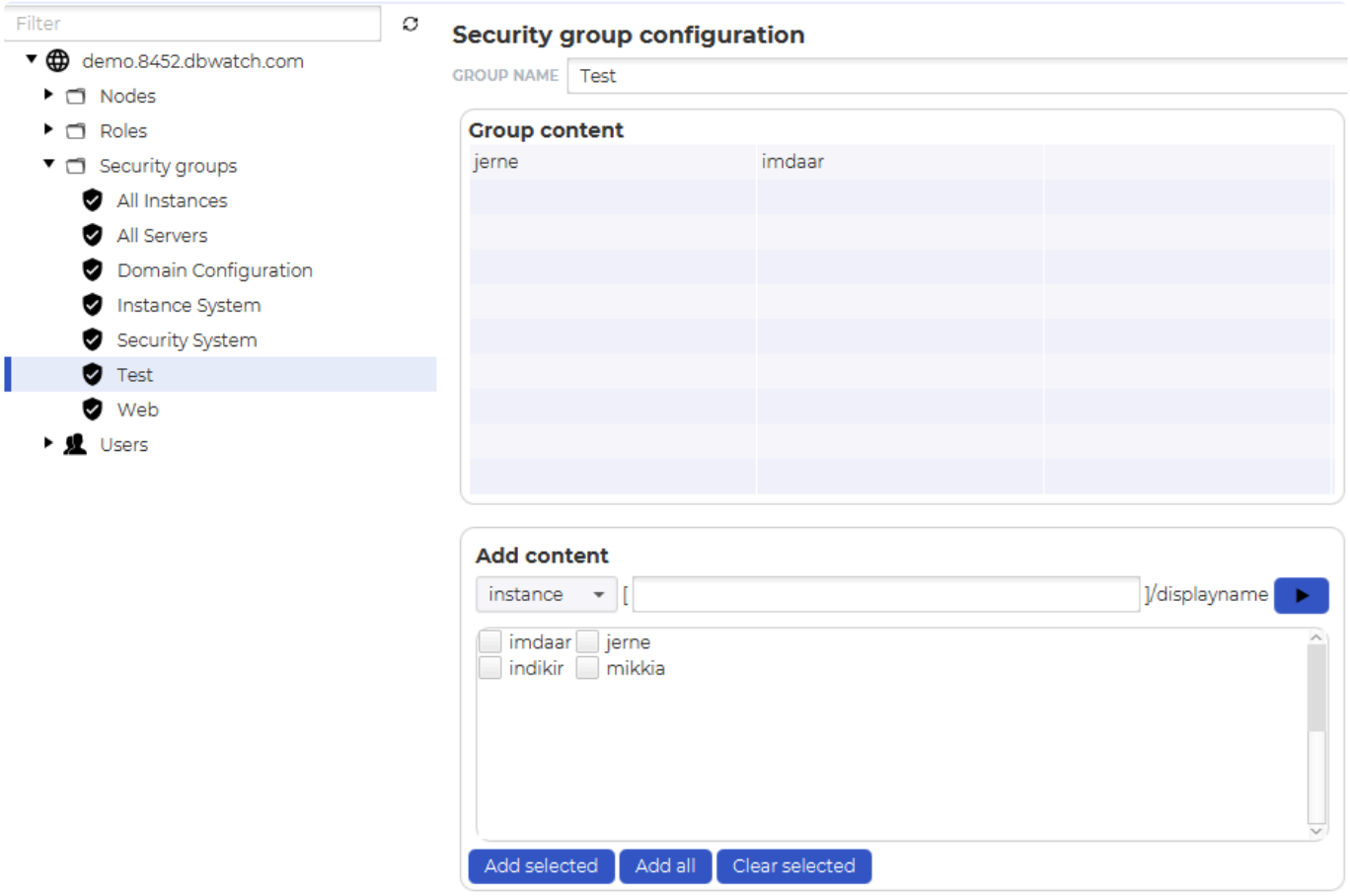
Users



This is were you add, delete and configure users in dbWatch.

Security groups

This is were you define the groups that users and roles can have privileges on.



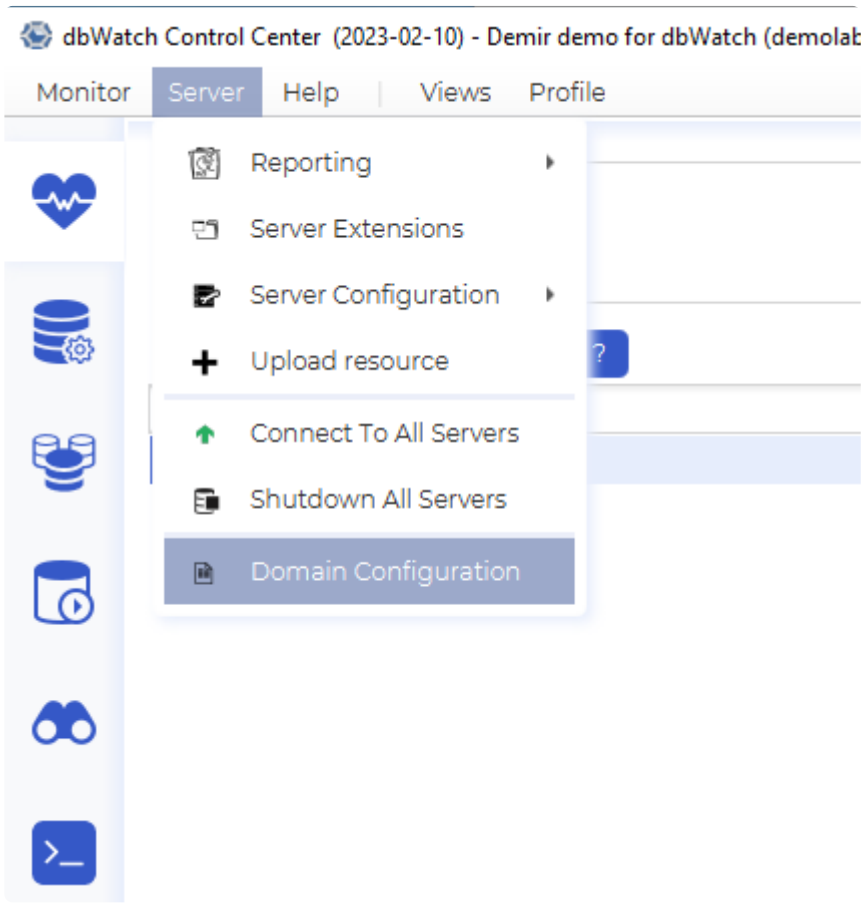
Look at the chapter for managing security groups for more detail.

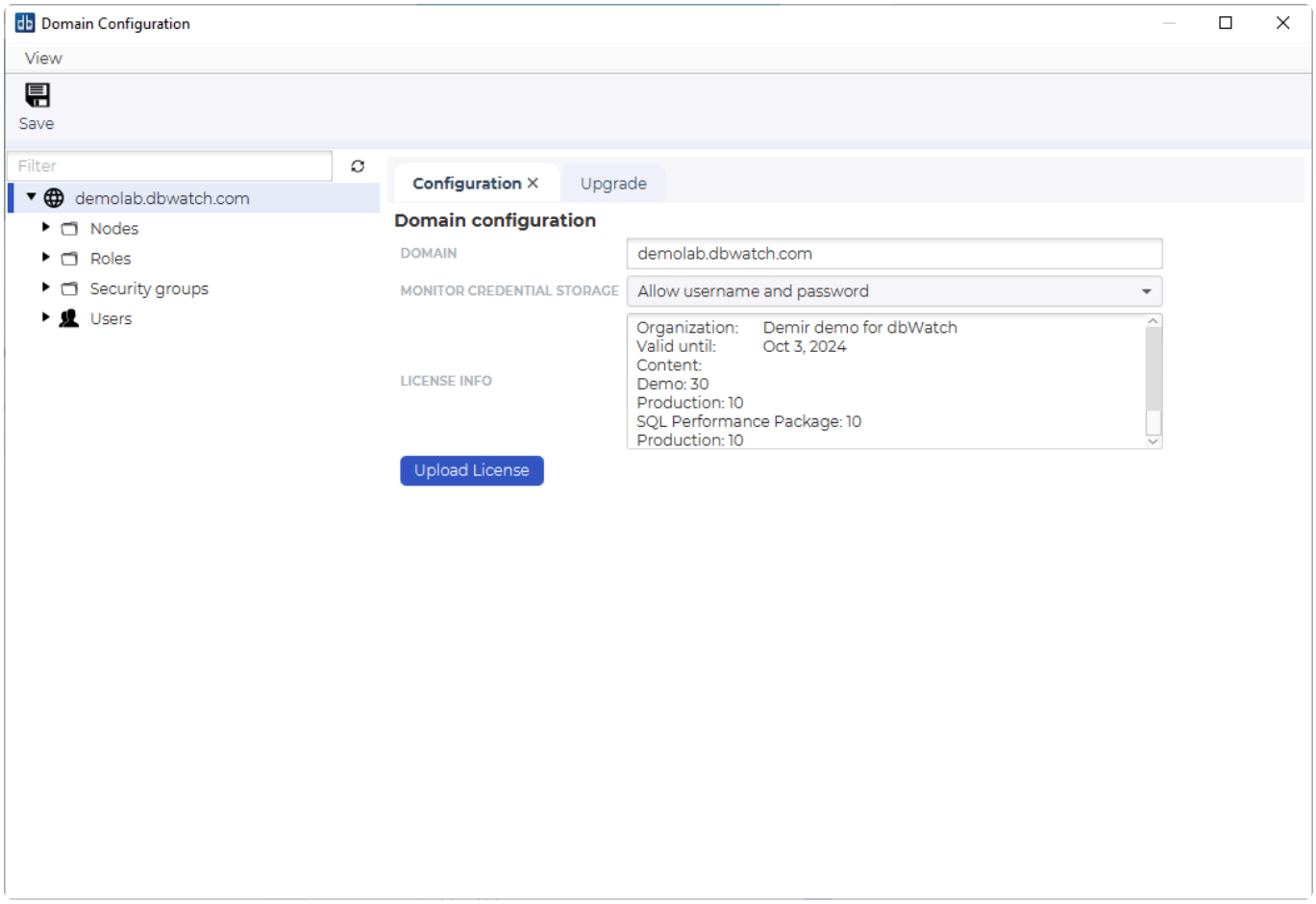
<< Service groups / Adding or changing license >>

Adding or changing license

Adding or changing license

To add or change the license, you need to start the domain configuration window. (Server->Domain Configuration)



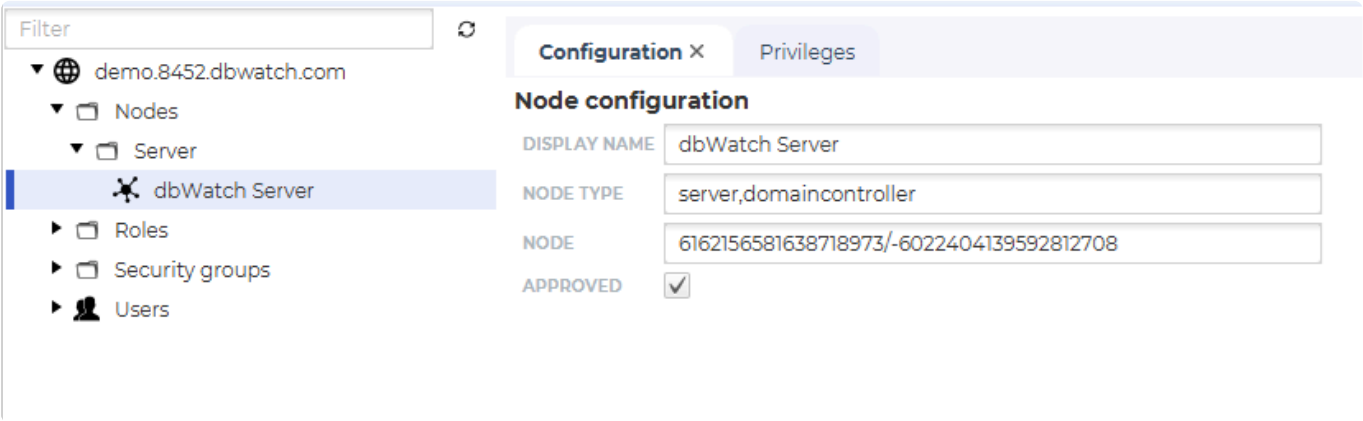


In this window, click on “Upload license” and point it to the license .json file you received from dbWatch.

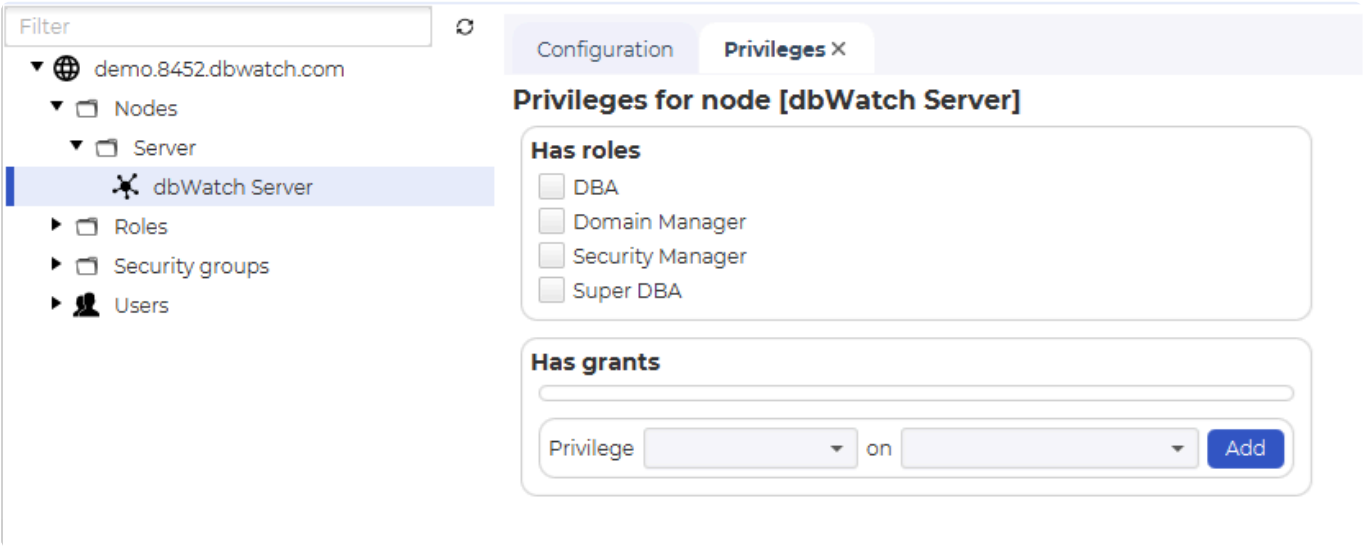
If the domain information in this file differ from the configured domain, as would be the case when you are moving from a demo license to a permanent one, you will be prompted to migrate to a new domain configuration. Accept this to move instances configured to the new domain.

Managing nodes

Nodes



In the node configuration there are 2 editable fileds. The displayname and the approved checkbox. This is were you approve new Servers that want to join the domain.
In addition you can see the type (currently server and/or domaincontroller) and id for the node (used in the internal network routing).

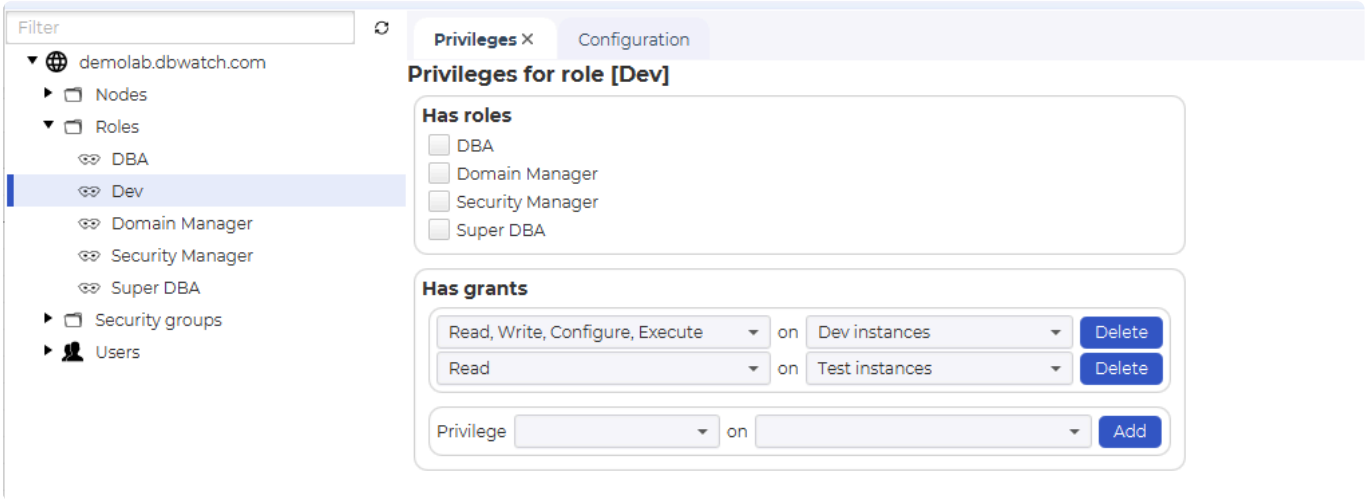


A node can also have privileges in the same way that a user has privileges. Currently this is only relevant for CCC nodes.

<< Adding or changing license / Managing roles >>

Managing roles

Roles



In the roles configuration you can add, delete and configure roles. In this example we have created a “Dev” role that has Read, Write and Configure on the security group “Test instances”, aswell as “Read” on “Dev instances”

There are 4 predefined roles:

Rolename	Privilege	Can be altered by user
DBA	Read,Write,Configure on All instances	Yes
Domain Manager	Special role enabling editing of domain configuration	No
Security Manager	Special role enabling changes to the security system	No
Super DBA	Special role with all privileges on all instances	No

For specific privileges look at the Privilege and actions map

<< Managing nodes / Managing users >>

Managing users

A user can have roles and grants. A grants is set of privileges on a “Security group”, while roles are a set of grants or other roles or security groups.

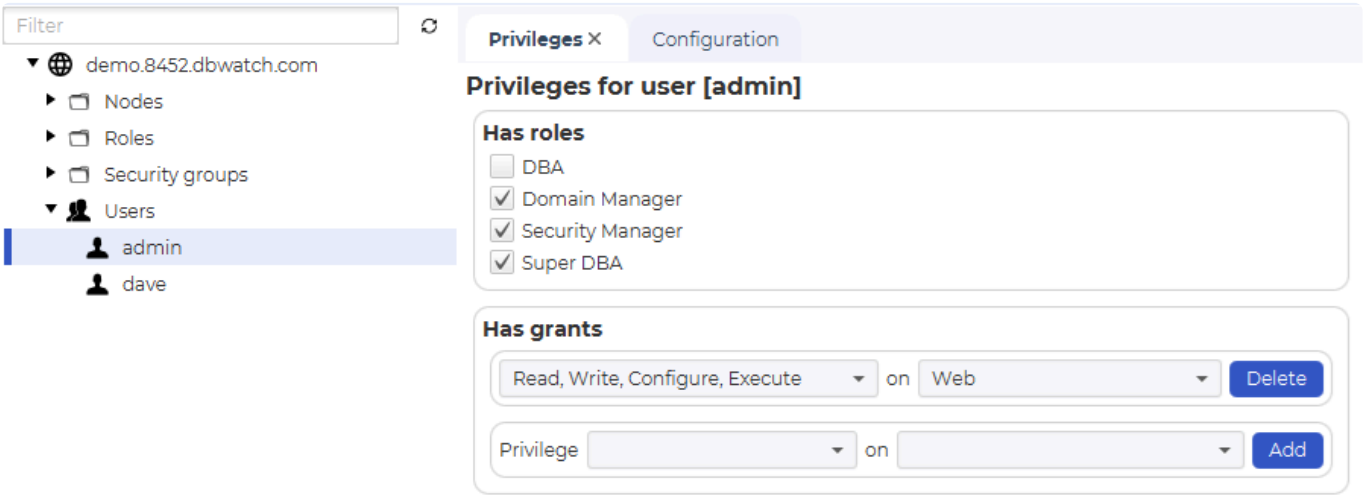
A “Security group” is a set of database instances and/or servers.

Four roles are predefined:
Domain Manager, a special role, that gives privileges to edit the domain configuration. This is to change anything related to the setup of the dbWatch Control Center installation. This role cant be altered, removes, modified or have extra privileges added to it.
Security Manager, a special role that gives privileges to edit the security system in dbWatch.
Super DBA, a special role, that gives all privileges on all the registered database instances. This role cant be altered, removes, modified or have extra privileges added to it.
DBA, by default, this gives all privileges on all the registered database instances. But different from the “Super User” role, this can be changed, and the list of instances can be changed.

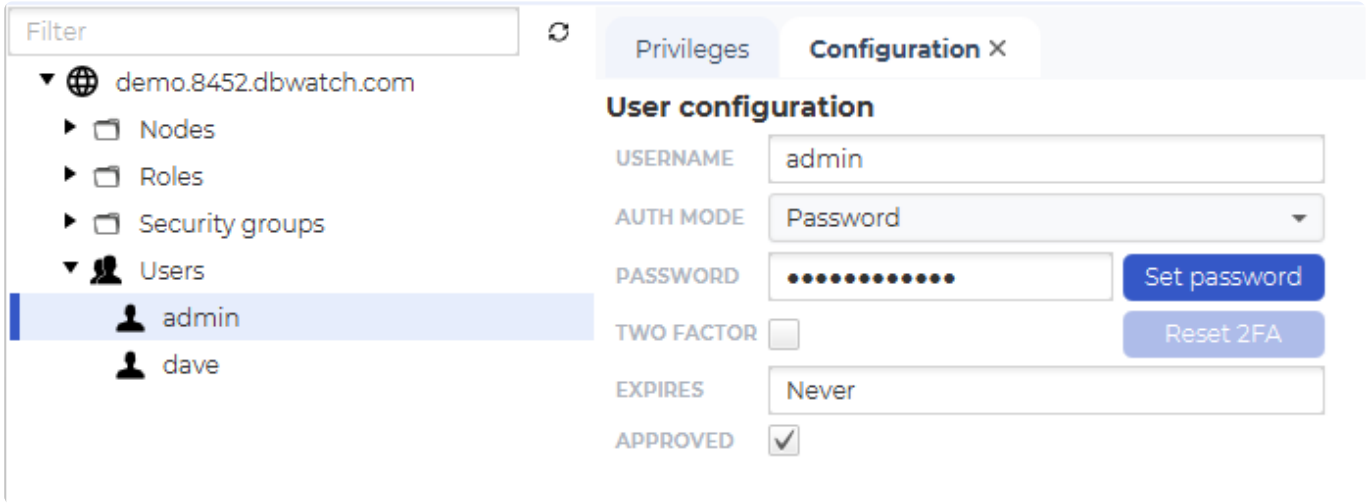
You can also create your own roles.

The normal setup is to give users either the “DBA” or “Super User” roles, and have the users that also configure security and setup of Control Center have the “Domain MAnager”/“Security Manager” role.

For specific privileges look at the Privilege and actions map



See details for 2 factor authentication.

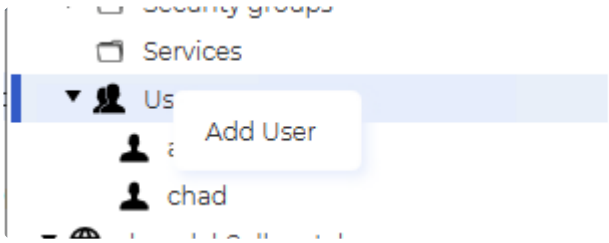


For specific privileges look at the Privilege and actions map

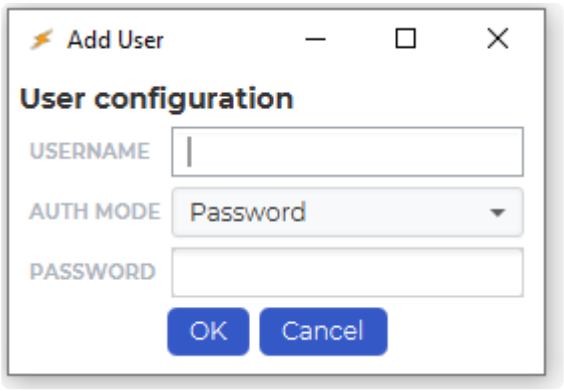
<< Managing roles / Creating a user >>

Creating a user

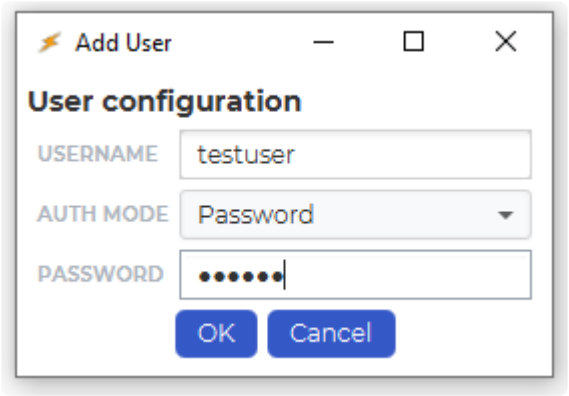
To create a user, right click on the “Users” element in the tree in Domain Configuration



This will open the add user dialog:

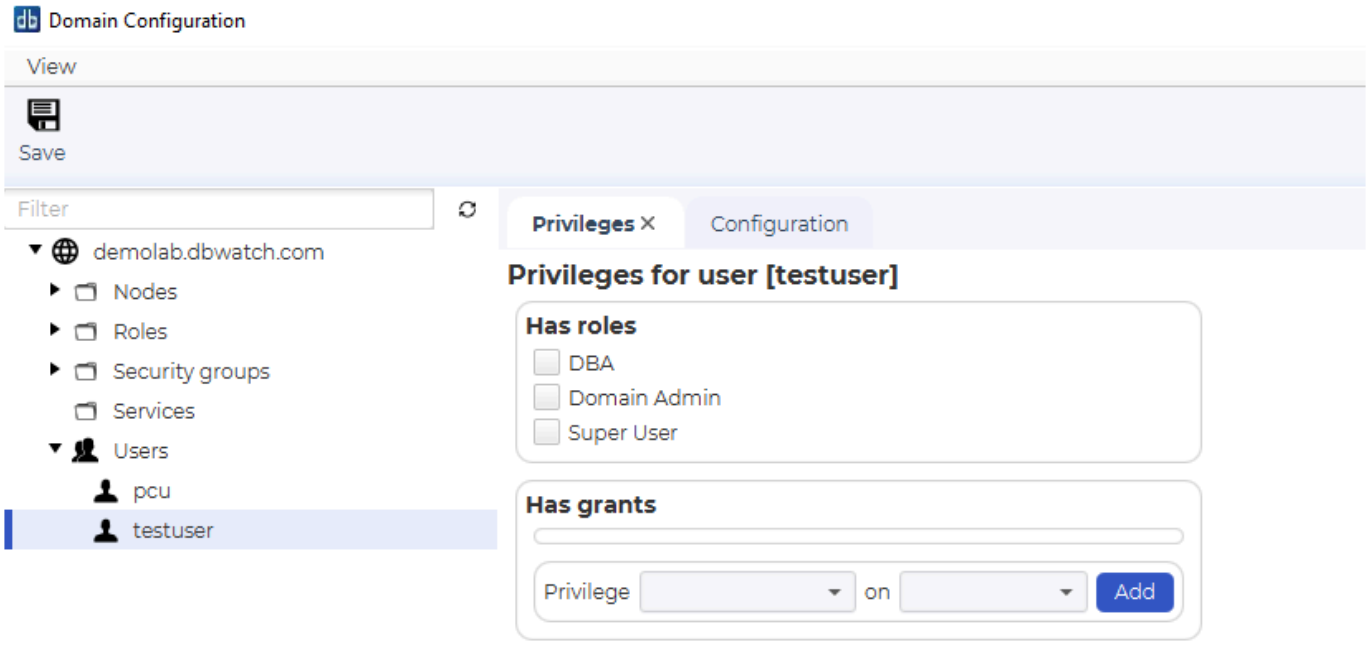


Where you add a username and password:



If you want to add a user that uses Active Directory for password verification, look into the section User integration with Active Directory

Once the user is created, it will exist in the tree, but have no privileges:



For details on setting privileges, see Managing Users.

<< Managing users / User integration with Active Directory >>

User integration with Active Directory

The user integration with Active Directory offloads the password verification and storage from the dbWatch Control Center domain controller to Active Directory.

Port openings

To allow the correct communications between the dbWatch Server and the Active Directory server, some firewall openings must be in place. Typically 2 port openings are used when communicating with AD.
Port 389/TCP for unencrypted and TLS connections
Port 636/TCP for SSL connections
Depending on the setup, one or two of these ports must be open.

Mapping the domains

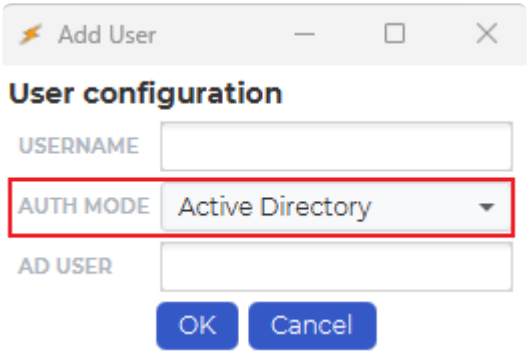
In the configuration directory of Control Center, by default “C:\ProgramData\dbWatchControlCenter\config\”, there is a directory domain and under that one directory for your specific Control Center domain.
Example:
bc.. C:\ProgramData\dbWatchControlCenter\config\domain\test.com\
p. In this directory a file, ldap.json must be created to provide a mapping between the Active Directory domain and the Active Directory server. dbWatch Control Center is able to authenticate users for multiple domains.
Example mapping file:

```
{
"DEMOAD1": {
    "server":"ldap://10.200.10.90"
  },
"DEMOAD2": {
    "server":"ldap://10.200.10.100"
  }
}
```

A restart of the Control Center service is necessary to trigger a detection and read of this file.

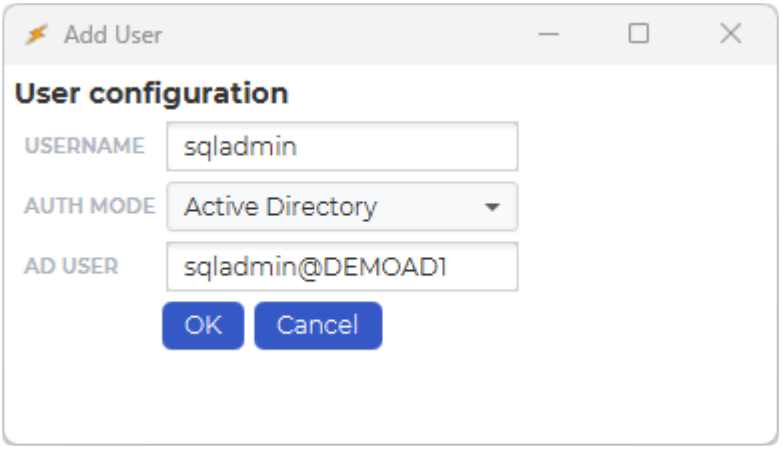
Creating users

When a new user is then created, the “AUTH MODE” must be set to “Active Directory”:



Mapping the usernames

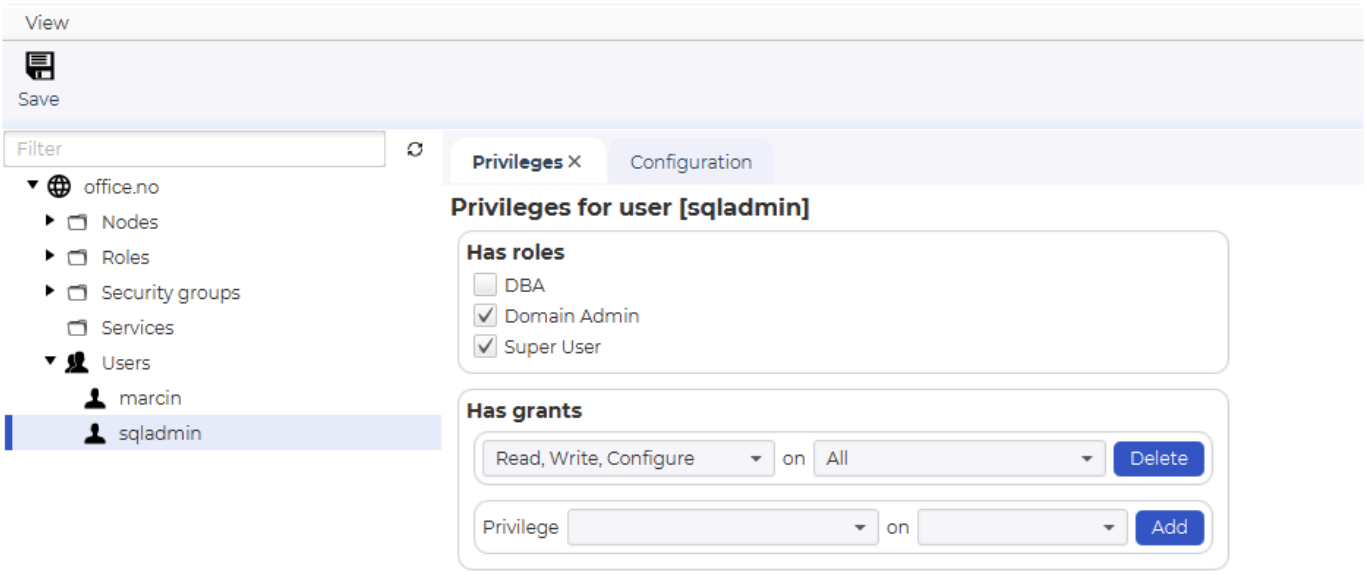
You must then map the Control Center username, in “USERNAME” to an Active Directory user, “AD USER”. It can be a similar username, but the “AD USER” must provide a domain after the “@”, that corresponds with the domain in the ldap.json mapping file.



Depending on your domain setup it might be necessary to prepend the domain name to the username. This is on the form domainname\username@ldap.json reference
So for the user test in the domain DEMOAD1 this would be DEMOAD1\test@demoload1

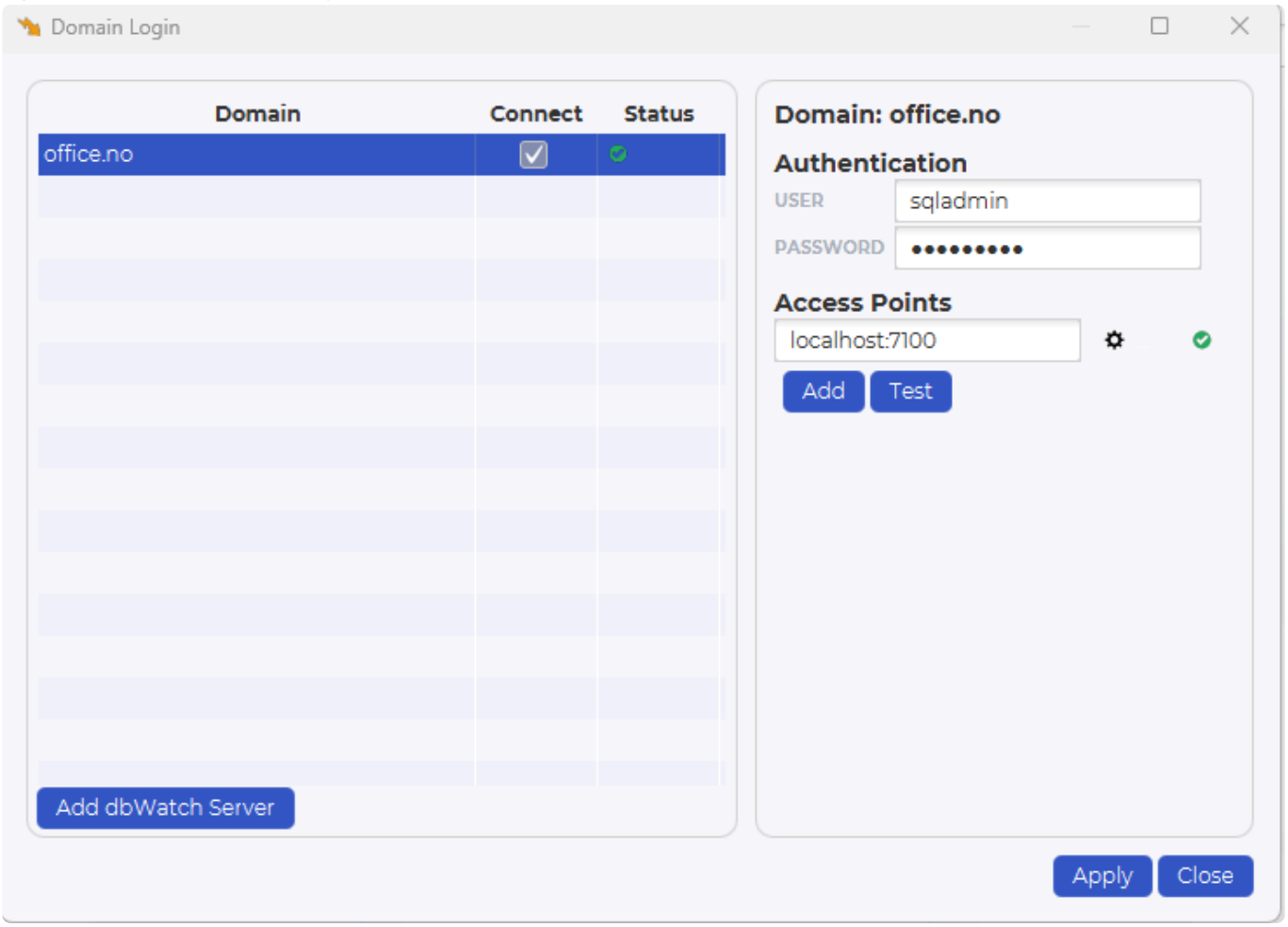
Add the credentials

As a normal user, you can then add and modify privileges in Control Center.



Test connection

Once all this is in place you should be able to test the connection. The password will be authenticated against the Active Directory server.



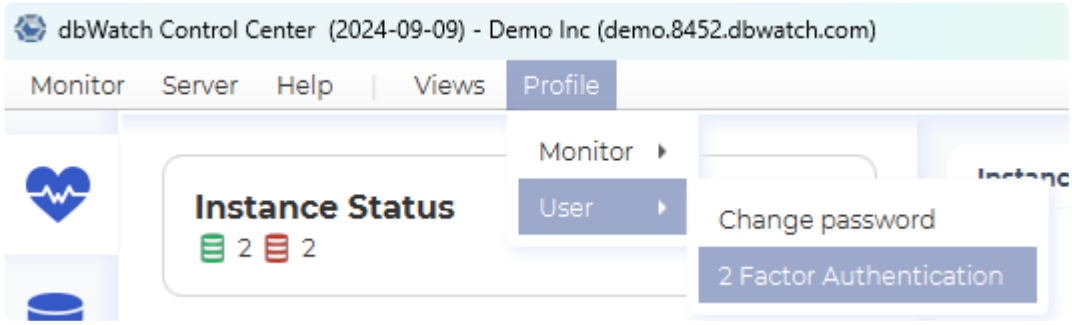
<< Creating a user / 2 factor authentication >>

2 factor authentication

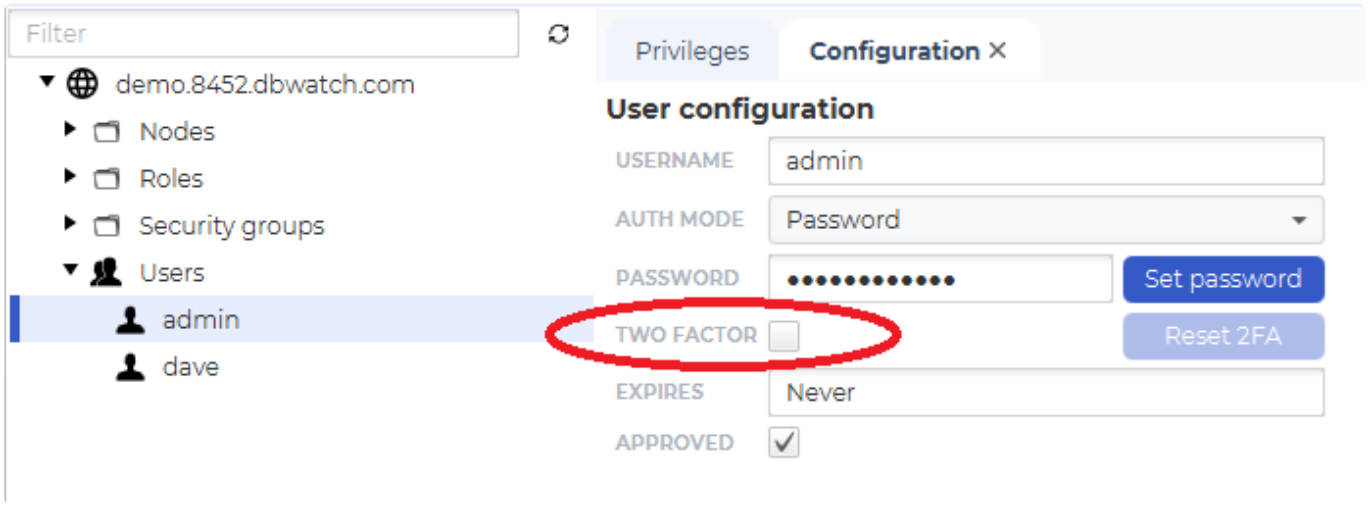
dbWatch offers 2 factor authentication through a TOTP (Time-Based One Time Password) mechanism.

Tou can use your favourite authenticator for this. For example Google Authenticator or Microsoft Authenticator.

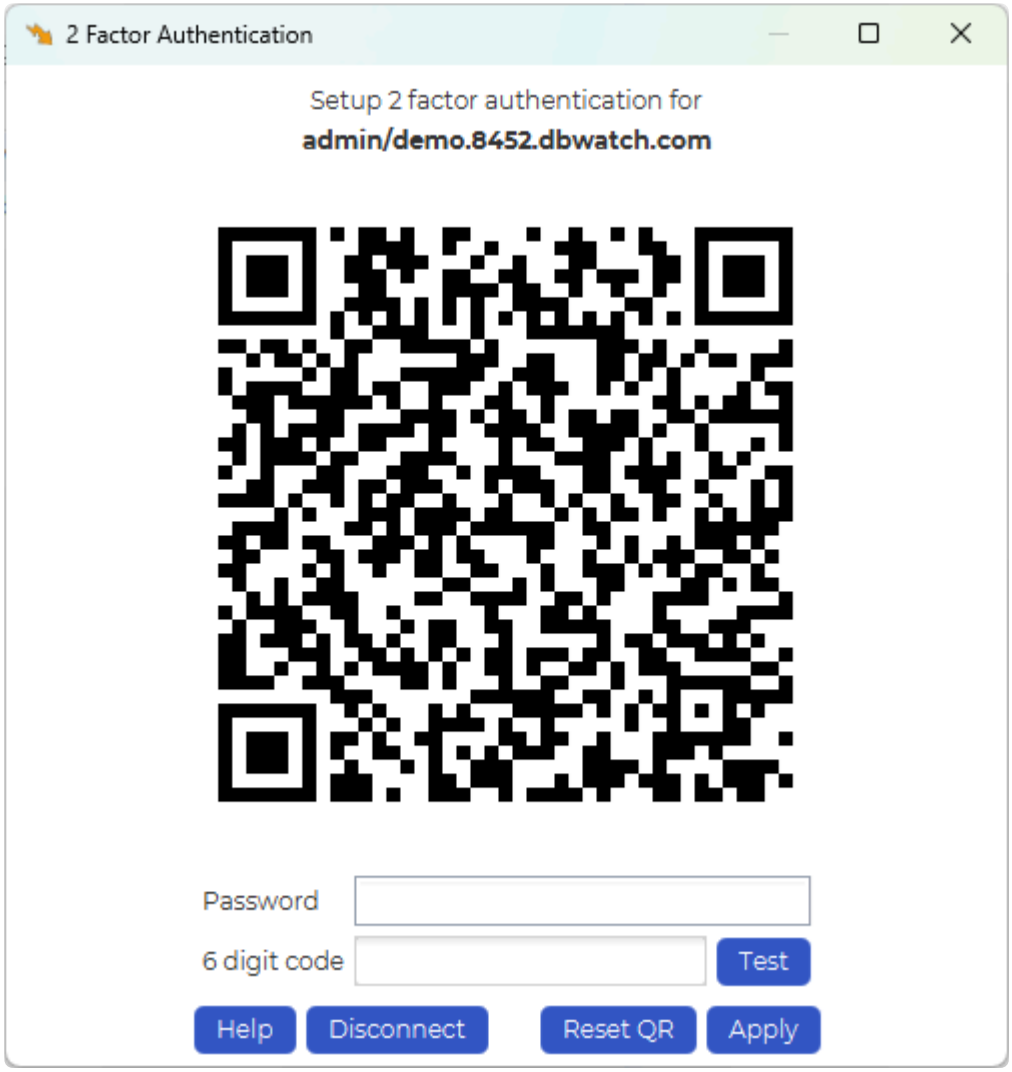
To enable 2 factor authentication for your own user, you can click the “Profile” -> “User” -> “2 Factor Authentication” menu.



To enable 2 factor authentication for other users, click the “2 Factor” checkbox in the user configuration.



The next time the user logs on,the 2 factor authentication initialization dialog will appear.



Scan the QR code in your Authenticator App. Enter the password and the 6 digit authenticator code to verify the setup.

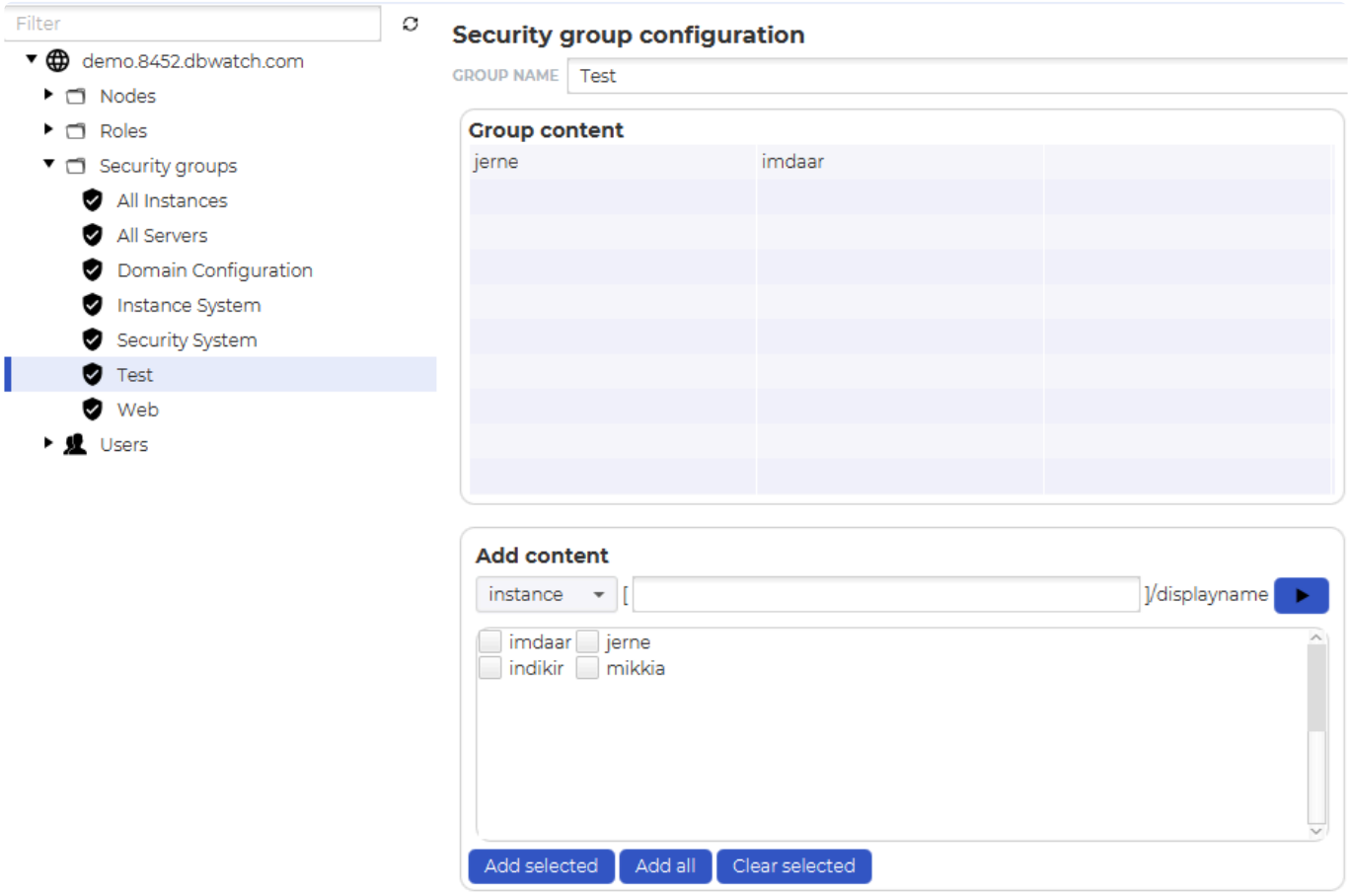
The user will be prompted for a new 6 digit code each time he/she logs on.

<< User integration with Active Directory / Managing security groups >>

Managing security groups

Security groups

This is were you define the groups that users and roles can have privileges on.



There are some predefined security groups:

Security group name	Description
All Instances	A group containing all instances in this dbWatch Domain
All Server	A group containing all dbWatch servers in this dbWatch Domain
Domain Configuration	A group that restricts access to the domain configuration
Instance System	A group that restricts access to adding and removing instances
Security System	A group that controls privilges related to the security system (adding/removing users, changing privileges and so on.
Web	A group that restricts access to the Web Server

User defined security groups can currently contain instances and servers. In the example, the security

group “Test” contains the two instances “jerne” and imdaar”.

For specific privileges look at the Privilege and actions map

<< 2 factor authentication / Privileges and actions map >>

Privileges and actions map

Overview of actions and required privilege

Instance is the specific database instance.
Instance system is a predefined security group controlling the overall instance monitoring and management.
Domain configuration is a predefined security group controlling the dbWatch Control Center configuration.
Security system is a predefined security group controlling access control.

Object type	Action	Required privilege	On what security object
Instance	View	Read	Instance
Instance	Edit configuration	Configure	Instance
Instance	Add new instance	Write	Instance System
Instance	Rename instance	Write	Instance
Instance	Uninstall instance	Write	Instance System
Instance	Remove instance	Write	Instance System
Instance	Connect	Configure	Instance
Instance	Disconnect	Configure	Instance
Server report configuration	View	Read	Domain Configuration
Server report configuration	Edit	Write	Domain Configuration
Job	Install new job	Write	Instance
Job	Uninstall job	Write	Instance
Job	Acknowledge status	Configure	Instance
Job	View configuration	Read	Instance
Job	Edit configuration	Configure	Instance
Job	Set schedule	Configure	Instance
Job	Enable job	Configure	Instance
Job	Disable job	Configure	Instance
Job	Run job	Execute	Instance
Job	View report	Read	Instance

Job script	Edit	Write	Domain Configuration
Job script	View	Read	Domain Configuration
Job templates	Trigger sync	Write	Domain Configuration
Server	Rename	Write	Domain Configuration
Server	Shutdown	Write	Domain Configuration
Server	Edit firewall	Write	Security system
Server	View configuration	Read	Domain Configuration
Server	Edit configuration	Write	Domain Configuration
Server	Edit job templates	Write	Domain Configuration
Server	Delete job template	Write	Domain Configuration
Resource	Upload	Write	Domain Configuration
Group	Delete	Write	Domain Configuration
Group	Add	Write	Domain Configuration
Group	Rename	Configure	Domain Configuration
Group	Edit	Configure	Domain Configuration
Group	View configuration	Read	Domain Configuration
Extensions	Disable	Configure	Domain Configuration
Extensions	Enable	Configure	Domain Configuration
Extensions	View configuration	Read	Domain Configuration
Extensions	Edit configuration	Configure	Domain Configuration
Global reports	Edit definition	Write	Domain Configuration
Global reports	View definition	Read	Domain Configuration
Global reports	Run report	Execute	Instance
Global reports	Schedule	Execute	Instance
Scheduled report	Delete scheduled report	Write	Domain Configuration
Scheduled report	Configure scheduled report	Configure	Domain Configuration
Scheduled report	Run report	Execute	Domain Configuration
Domain	Add/Edit/Delete User	Write	Security System
Domain	Add/Edit/Delete Role	Write	Security System
Domain	Add/Edit/Delete Security group	Write	Security System
Domain	Add/Edit/Delete Node	Write	Security System

Domain	Edit domain properties	Write	Domain Configuration
Domain	Change license	Write	Domain Configuration
Autodiscover	Start scan	Write	Instance system
Autodiscover	Stop scan	Write	Instance system
Autodiscover	Edit scan	Write	Instance system
SQL Worksheet	Execute query	Write	Instance
Management	View data	Read	Instance
Management	Change data/parameters/ configuration	Write	Instance

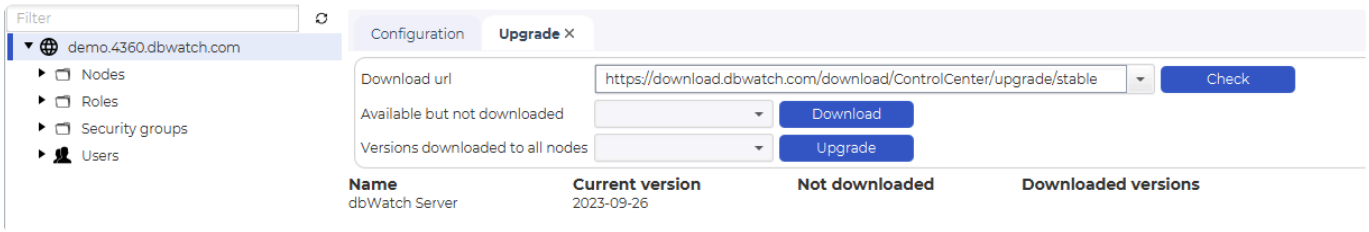
<< Managing security groups / Upgrade dbWatch >>

Upgrade dbWatch

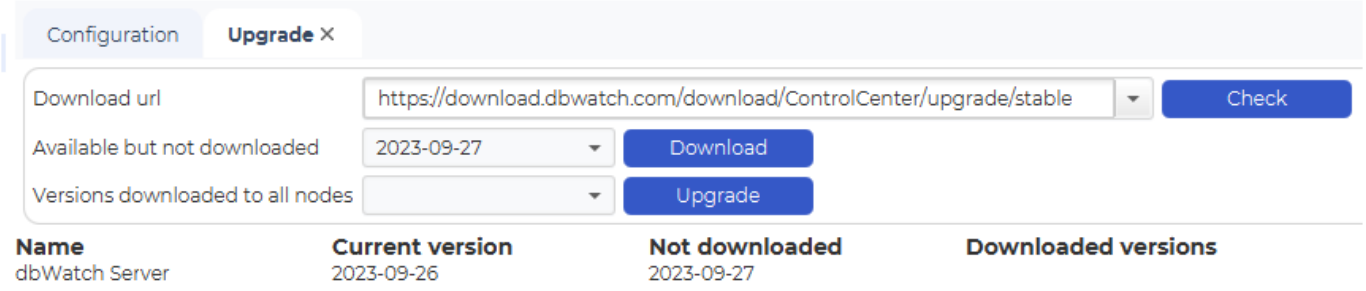
If you have dbWatch **version 2023-06-27** or later, it is possible to semi automatically upgrade the dbWatch Servers.

> Only available for Windows Servers. For linux, look here.

In the **Domain Configuration** dialog, select the **Upgrade** tab.

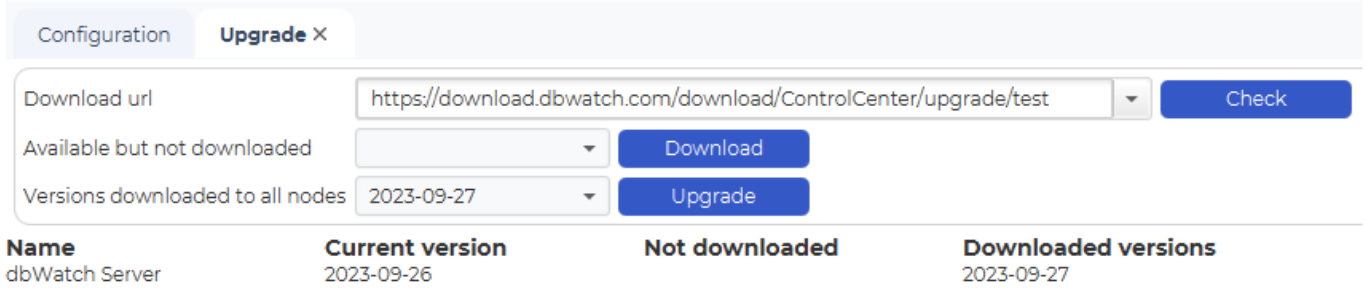


Click on the **Check** button to look for new versions of dbWatch.
The default url to look for new versions is **https://download.dbwatch.com/download/ControlCenter/upgrade/stable**.



If a newer version is found, it will appear in the “Available but not downloaded* box.

To download this version, click the **Download** button.



> Be aware that the download process will take a few moments. The spinning wheel on the download button may stop befrore the download is finished. If so, there is a graphical refresh issue. Leave the Upgrade tab and come back to it to trigger a refresh.

When the download is complete it will appear in the *Versions downloaded to all nodes” box.

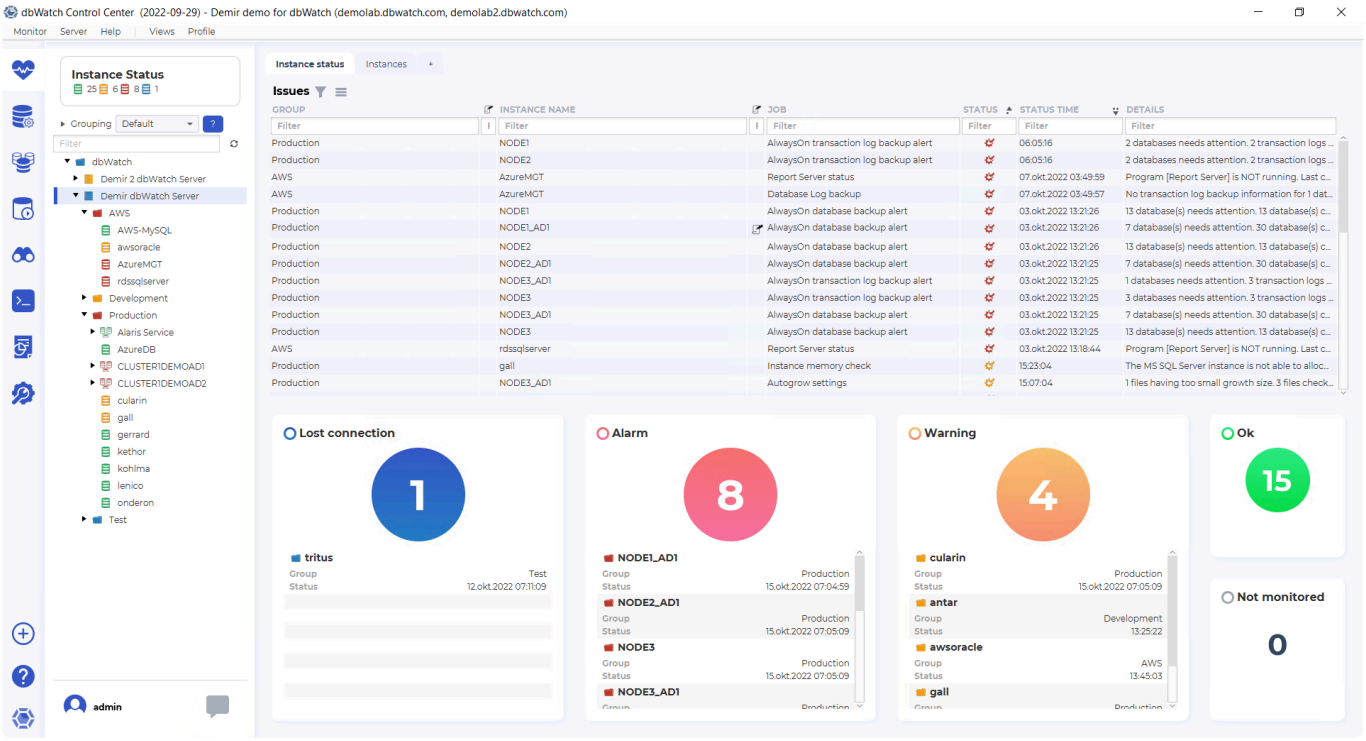
Click on **Upgrade**. The Servers will apply the update, and trigger a restart.

If you have dbWatch Monitoris installed on client machines, they still have to be upgraded manually.

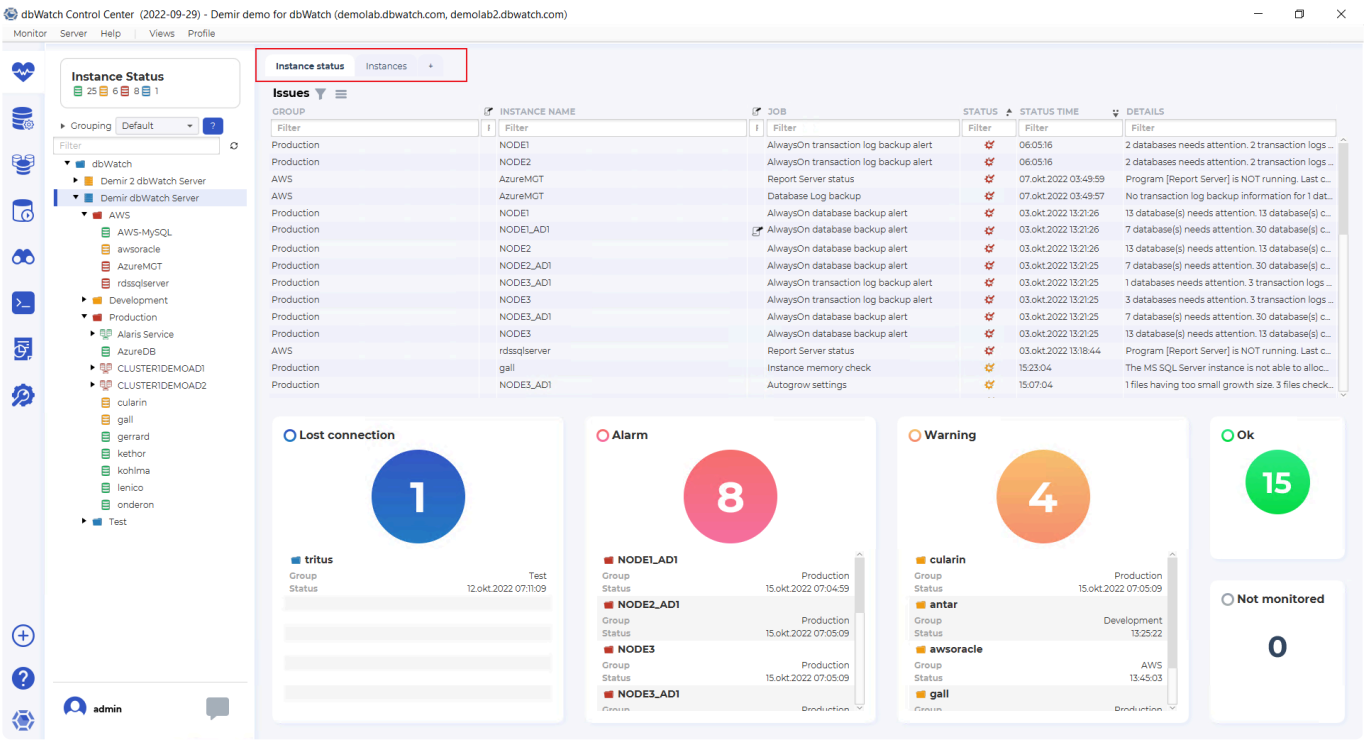
Monitoring

In this section, we'll be familiarized with how the Monitoring Module works.

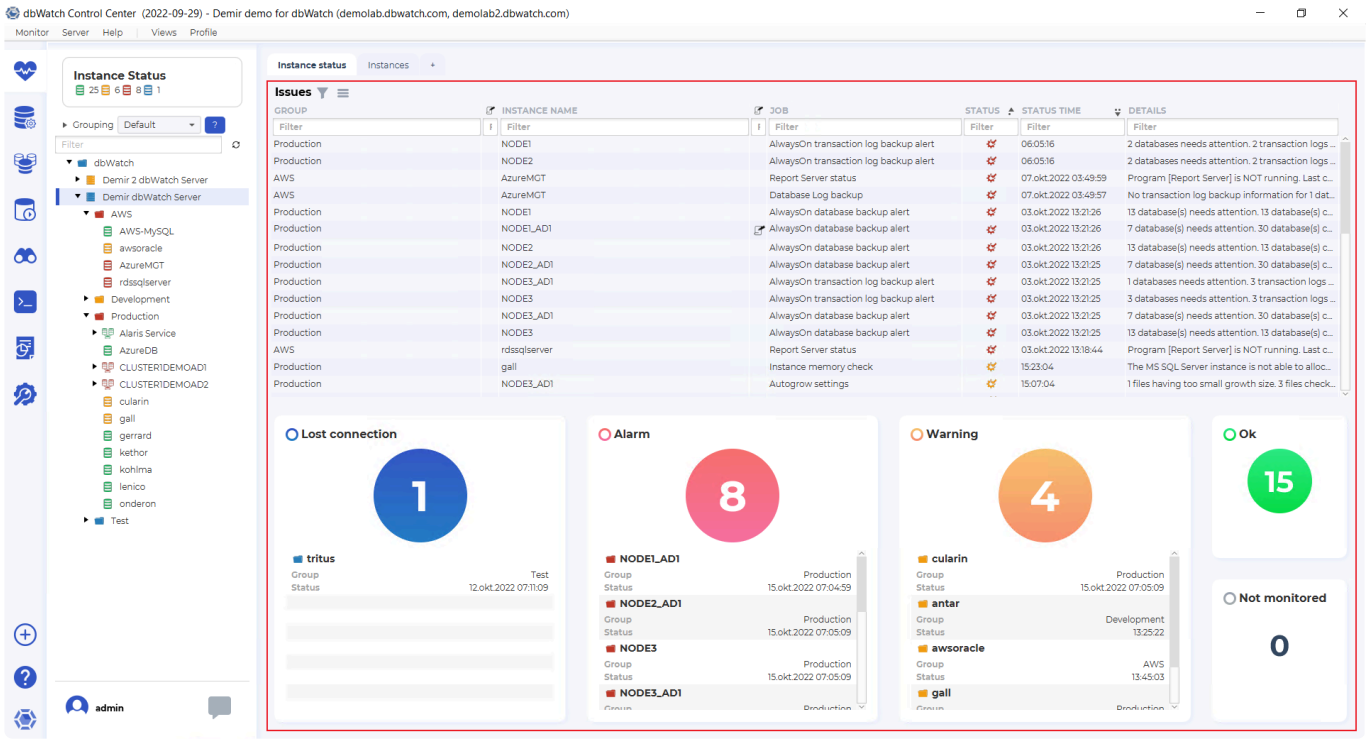
When opening the monitoring module on its initial state, you get the status dashboard:



The status dashboard is intended to provide a quick overview of the current state of your database instance environment.



The monitoring dashboard has tabs on the top, to switch between the dashboards.



The issue list dashboard has issues from the instances under the currently selected point the tree structure.
The issue list has the fields “Group”, “Instance note indicator”, “Instance name”, “Job note indicator”, “Job”, “Status”, “Status time” and “Details.”

The Group is the group you have put the instance into. By default we ship with groups “Development”, “Test”, and “Production”, this is, however, user-configurable.
The “Instance note indicator”, will show an icon when someone has created a note on this instance. Notes are used to store permanent and semi-permanent information about a database instance.
The “Instance name”, is the name you have given to the database instance when adding it to dbWatch. This can be changed and adjusted later.
The “Job note indicator”, will show an icon when someone has created a not on this job on this instance. Notes on jobs are used to store permanent and semi-permanent information about a job on a database instance, such as known issues and problems.
The “Job” is the name of the monitoring job that has found an issue. The jobs are named descriptively so it should be easy to understand what metric they are monitoring.
The “Static” icon indicates severity, yellow for warnings, and red for alarms. Only warnings and alarms will show up in this overview.
The “Status time” field will tell you at what time did this status change occur.
The “Details” field is used to show an error message regarding the problem detected. More in-depth information is available if you right-click on the issue line, and choose details.

Underneath the issue list, there are balls indicating the number of instances in each status, with a list of instance names, groups, and status change times. Blue ball is for “Lost connection”, red is for “Alarm”, yellow is for “Warning”, green is for “Ok” and if you have added instances without monitoring, they are in the “Not monitored” section.

STATUS	NAME	PLATFORM	EDITION	HOST	STARTED	SESSIONS	DISK USAGE (GB)	MEMORY USAGE (MB)	DB COUNT	ACTIVITY
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
	tritus	PostgreSQL 12.5	Community Edition	10.0.0.151	02.aug.2022 12:22:43	8	0	228	1	
	AzureMGT	SQL Azure 2014 (RTM)	SQL Azure	dbwatchmgt.pu...	20.sep.2022 19:47:52	17	1	2 143	4	
	NODE1	SQL Server 2016 (SP3)	Enterprise Edition	10.200.10.101	28.aug.2022 18:06:16	8	18	12 985	8	
	NODE1_AD1	SQL Server 2016 (SP3)	Enterprise Edition	10.200.10.91	28.aug.2022 18:07:05	26	16	12 439	14	
	NODE2	SQL Server 2016 (SP3)	Enterprise Edition	10.200.10.102	28.aug.2022 18:06:01	13	5	4 025	8	
	NODE2_AD1	SQL Server 2016 (SP3)	Enterprise Edition	10.200.10.92	28.aug.2022 18:07:05	21	9	5 859	14	
	NODE3	SQL Server 2016 (SP3)	Enterprise Edition	10.200.10.103	28.aug.2022 18:06:25	13	5	2 216	8	
	NODE3_AD1	SQL Server 2016 (SP3)	Enterprise Edition	10.200.10.93	28.aug.2022 18:06:43	20	11	4 572	13	
	rdssqlserver	SQL Server 2019 (CU15)	Express Edition	rdssqlserver19ex...	12.sep.2022 14:12:44	23	0	484	6	
	antar	Oracle 19.0.0.0.0	Enterprise Edition	10.0.0.109	02.aug.2022 11:13:34	6	5	3 557	1	
	awsoracle	Oracle 19.0.0.0.0	Standard Edition	dbworacle.cuhx...	20.sep.2022 16:16:20	11	2	1 725	1	
	culerin	SQL Server 2014 (RTM)	Enterprise Edition	10.0.0.111	04.aug.2022 08:29:38	7	0	584	4	
	AWS-MySQL	MySQL 8.0.28	Community Edition	mysql8.cuhxqm...	11.aug.2022 10:48:08	5	0	4 407	4	
	AzureDB	Azure SQL Database	Basic	dbwatch.databa...	21.okt.2020 11:40:19	59	0	29	1	
	alaris1	Oracle 11.2.0.1.0	Enterprise Edition	10.0.0.178	02.aug.2022 16:30:41	6	5	1 752	1	
	alaris2	Oracle 11.2.0.1.0	Enterprise Edition	10.0.0.179	22.aug.2022 13:37:18	4	5	1 727	1	
	alaris3	Oracle 11.2.0.1.0	Enterprise Edition	10.0.0.180	22.aug.2022 13:37:53	3	5	1 733	1	
	alaris4	Oracle 11.2.0.1.0	Enterprise Edition	10.0.0.184	23.aug.2022 14:24:29	5	5	1 729	1	
	gall	SQL Server 2017 (RTM)	Enterprise Edition	10.0.1.82	02.aug.2022 19:02:04	9	0	491	4	
	gerrard	SQL Server 2017 (RTM)	Enterprise Edition	10.0.1.84	02.aug.2022 19:23:05	8	0	202	4	
	kethor	SQL Server 2017 (RTM)	Enterprise Edition	10.0.1.85	02.aug.2022 19:43:40	9	0	638	4	
	kohlma	SQL Server 2017 (RTM)	Enterprise Edition	10.0.1.86	02.aug.2022 20:04:22	11	0	420	4	
	korriban	Oracle 12.1.0.1.0	Standard Edition	10.0.0.110	02.aug.2022 11:31:57	5	4	491	1	
	lenico	SQL Server 2017 (RTM)	Enterprise Edition	10.0.1.93	02.aug.2022 20:26:41	8	0	387	4	
	loronara	Oracle 11.2.0.3.0	Enterprise Edition	10.0.0.186	02.aug.2022 16:18:23	4	1	466	1	
	loronarb	Oracle 11.2.0.3.0	Enterprise Edition	10.0.0.190	02.aug.2022 16:28:16	3	1	455	1	
	ohmadun	SQL Server 2017 (RTM)	Enterprise Edition	10.0.1.95	06.aug.2022 00:47:34	9	0	593	4	
	onderon	SQL Server 2017 (RTM)	Standard Edition	10.0.0.115	02.aug.2022 11:54:05	8	0	923	4	

The “Instances” list the instances connected in a table format.
The fields are “Instance note indicator”, “Status”, “Name”, “Platform”, “Edition”, “Host”, Started”, “Sessions”; “Disk usage (GB)”, “Memory usage(MB)”, “DB count” and “Activity” bar.

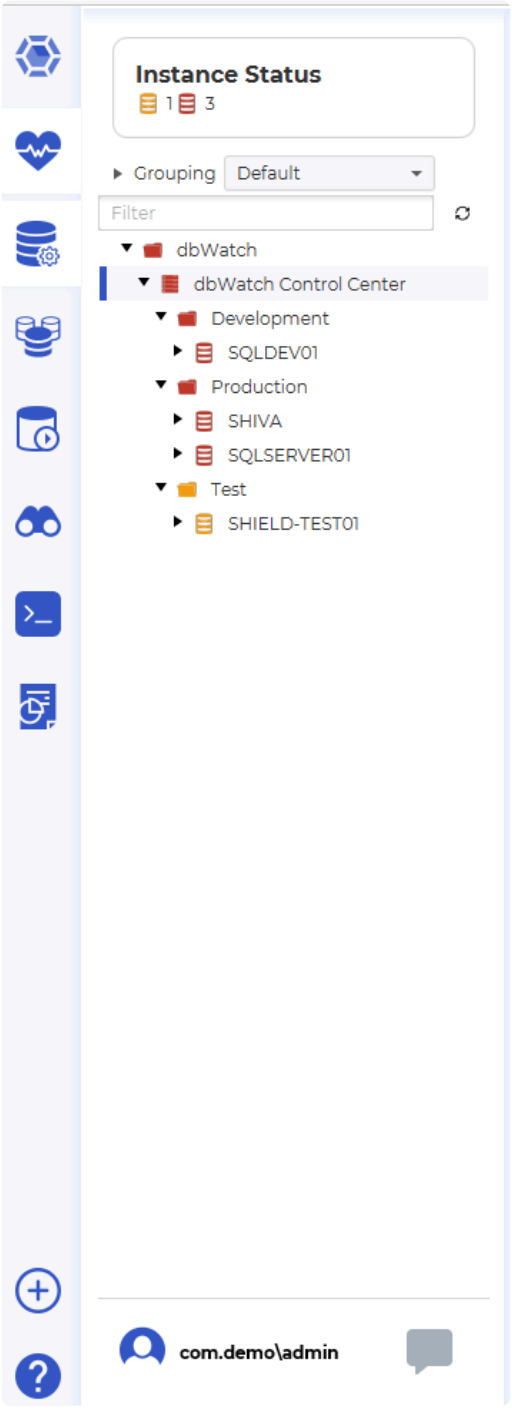
The “Instance note indicator”, will show an icon when someone has created a note on this instance. Notes are used to store permanent and semi-permanent information about a database instance.
The “Name”, is the name you have given to the database instance when adding it to dbWatch. This can be changed and adjusted later.
The “Platform” field, is the database type and version.
The “Edition” field, is the database edition type.
The “Host” field, is the hostname or IP address configured.
The “Started” field, is the last time this database instance has started up.
The “Sessions” field, is the current number of sessions connected to this database instance.
The “Disk usage (GB)” field, is the current disk usage by this database instance in GB.
The “Memory usage(MB)” field, is the current memory usage by this database instance in MB.
The “DB count” field, is the current database count in this database instance.
The “Activity” bar, is a calculated compound metric estimating the current activity load on this database instance.

You can use the tree structure to navigate to a database instance to get into the overview of what monitoring jobs are running on that instance.

<< Upgrade dbWatch / Tree navigation >>

Tree navigation

On the left side of the dbWatch monitor, you will see a tree structure. There are 2 component that lets your group and filter instances.



Top level of the tree, there is a “Grouping” panel. Below it is a quick text based filter.

Grouping

This is always set to “Default” but you can select a grouping appropriate for monitoring activities. To select a predefined grouping, click the drop down panel and click the grouping of your choosing. Aside

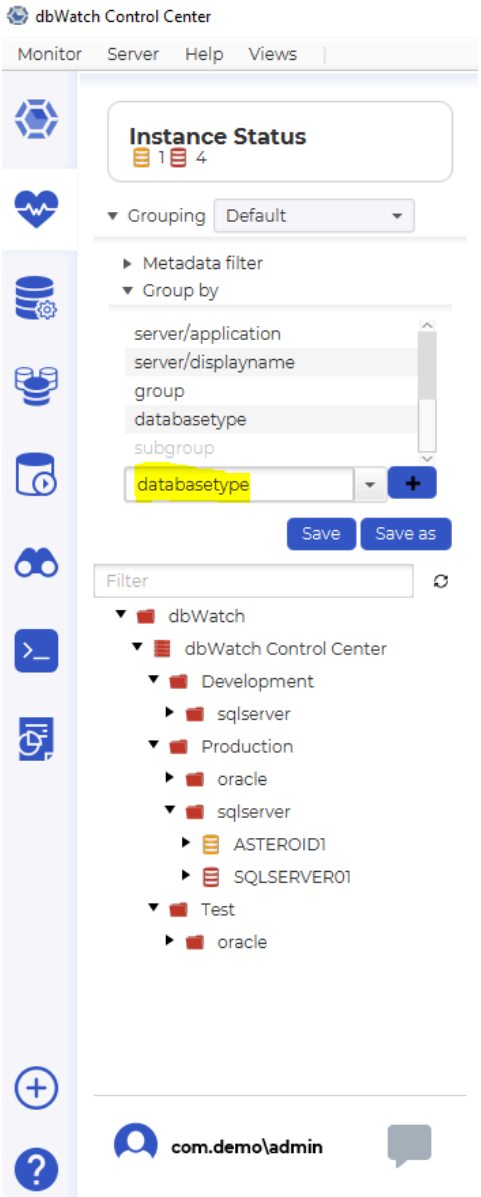
from selecting a predefined grouping, you can create new grouping definition as we shall see shortly.

Group by

The group by panel contains a list of properties that the instances in the tree are to be grouped by. In the example above, you see that we are grouping by two properties, server and group. This corresponds to the nodes in the tree. The top node is the Server (we only have one in this case) and on the next level we have the groups (Development, Production and Test).

You can add properties by typing them in the textfield and clicking the plus sign.

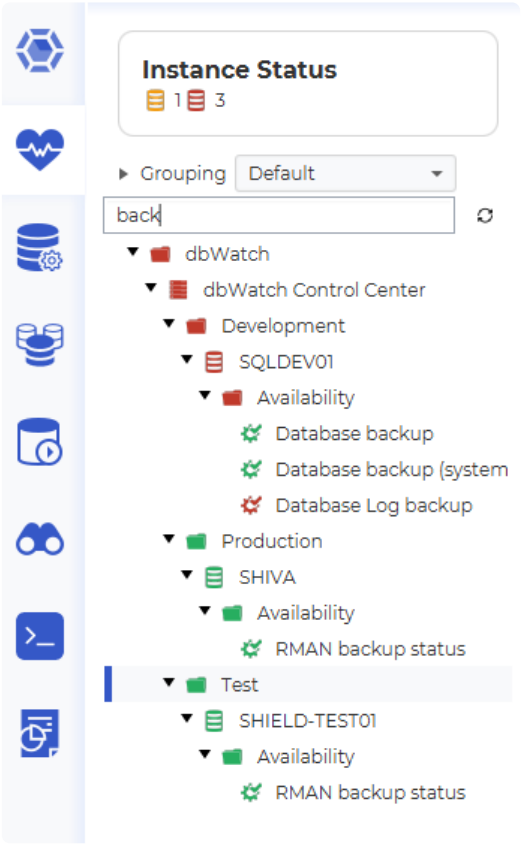
For example if we type in “databasetype”, we see that the database types appears as a new level in the tree.



Text filter

The text filter works by removing all branches of the tree that do not have a node matching the filter. In

the example below, we type in “Must” and all instances except the one called “Mustafar” are removed.



The text filter is not restricted to instance names, all nodes in the tree are considered. For example typing “Back” will remove all branches that do not contain that text. In our case that will leave only instances with an alert named something with “backup”, and also remove all other alerts.

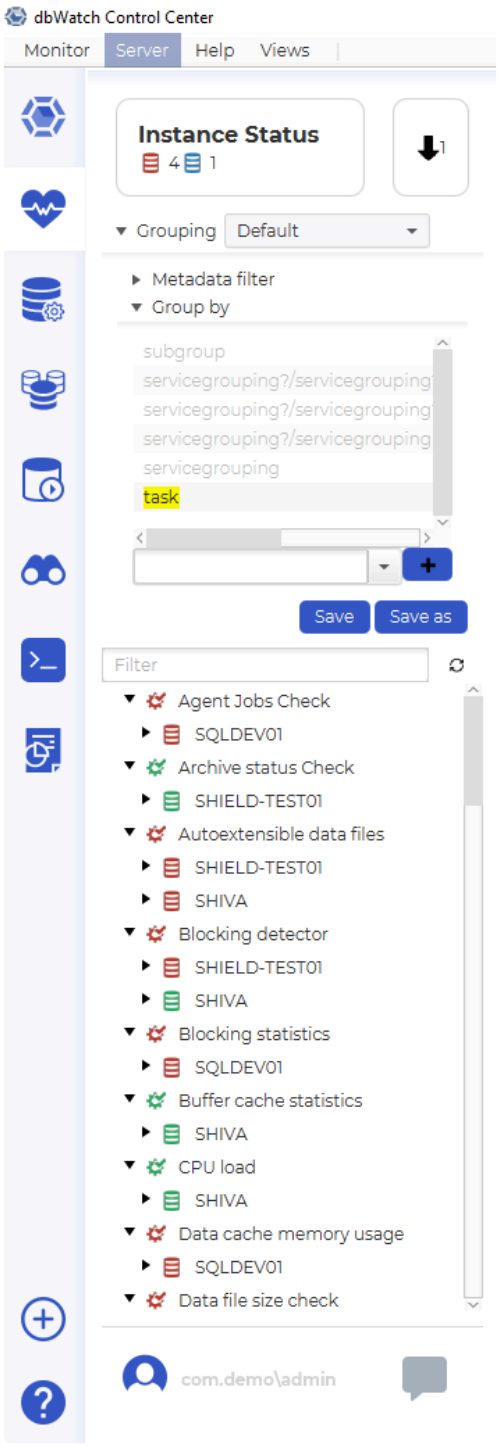
When the tree is a “status tree” (statuses propagate in the tree), the filtering will also result in the propagated statuses changing. For example if an instance has 2 alerts, a backup alert with status warning and a disk space alert with status alarm, the worst status will propagate to the instance (in this case alarm). The if you type “backup” in the filter box, the disk space alert will dissapear and the instance status will change to warning.

The order of the groupings can be rearranged by drag and drop in the list. You can remove them again by right clicking and selecting “Delete”.

You can save the new grouping definition by clicking “Save”, it will then replace the definition selected in the dropdown on the top.
You can save it as a new definition by clicking “Save as” and typing in a new name.

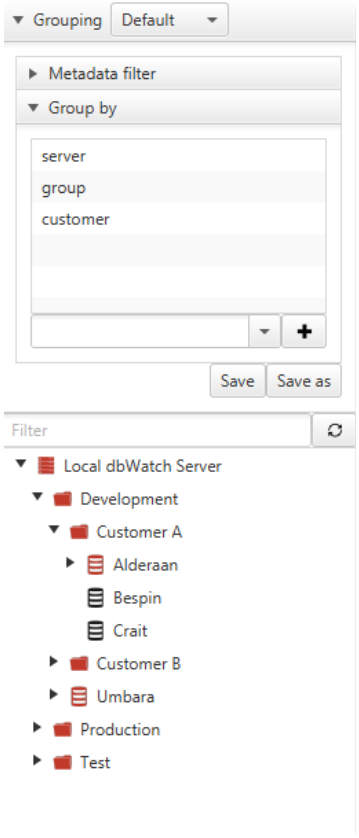
This dynamic grouping is an extremly powerfull feature that allows us to dynamically change your entire view of your environment.

For example, we can remove all the properties (right click on them and click delete) then add “task” (type task in the textbox and hit enter or click “+”).



Now we see that our instances are grouped by what tasks they have installed. For example we can see that only Alderaan has the “Job Scheduling check” installed.

In another example, we group by customer.



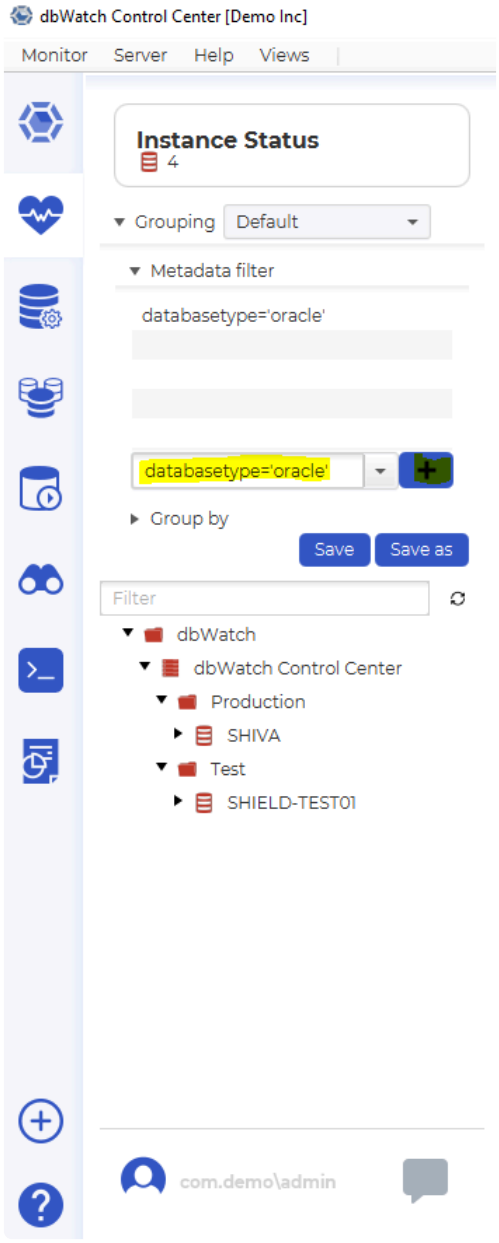
And we see that “Alderaan”, “Besbin” and “Crait” belongs to “Customer A” and another set of instances belongs to “Customer B”. Also notice that instances that do not have a defined customer are grouped directly under the group node.

All properties defined on instances can be used to group by. See the section on the Property System for details.

Metadata filter

The metadata filter is more powerfull than the simple text filter and allows you to filter instances based on all the properties available on the instance.

In the example below we state that only Oracle instances should be visible.



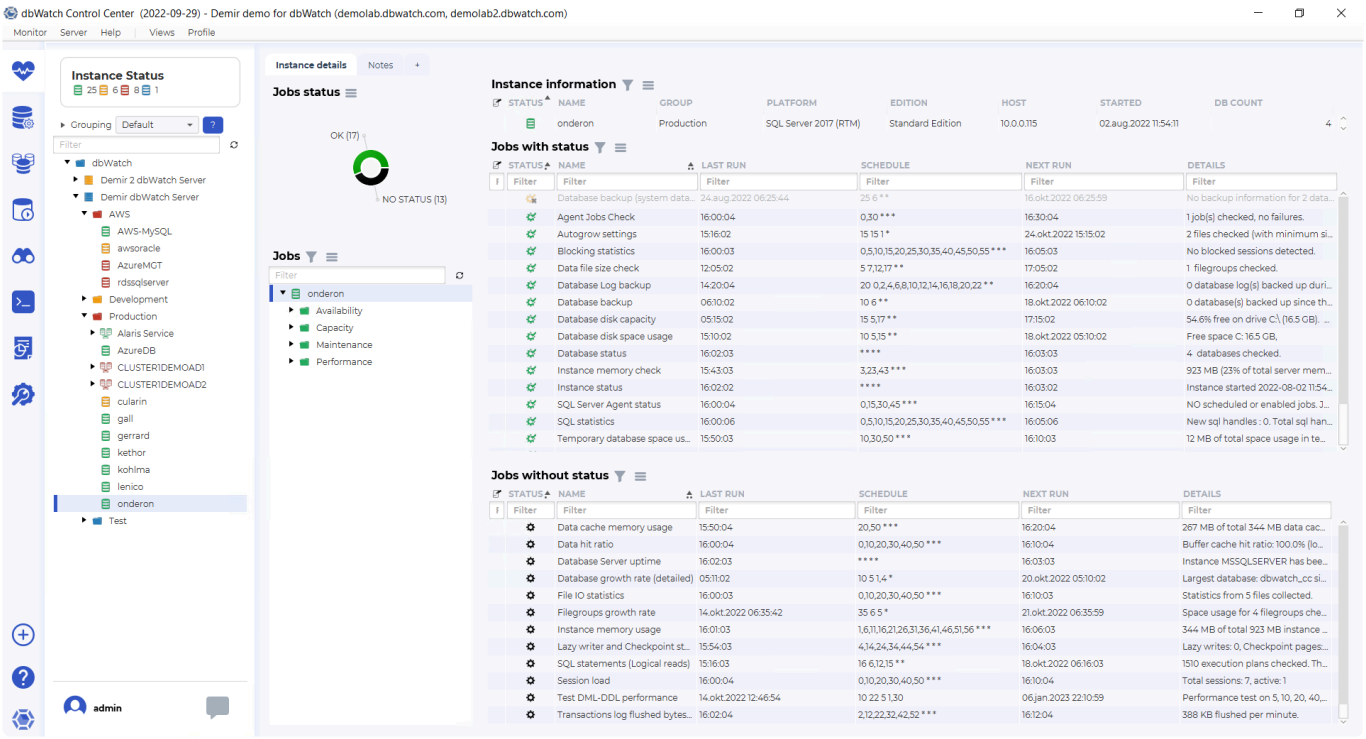
The lines in the metadata filter correspond to filters that can used in FDL queries for instances (The part in the square brackets). Read more about FDL

To see all the available properties that can be used in the metadata filter, open the FDL Console and type “instance/” and press the tab key.

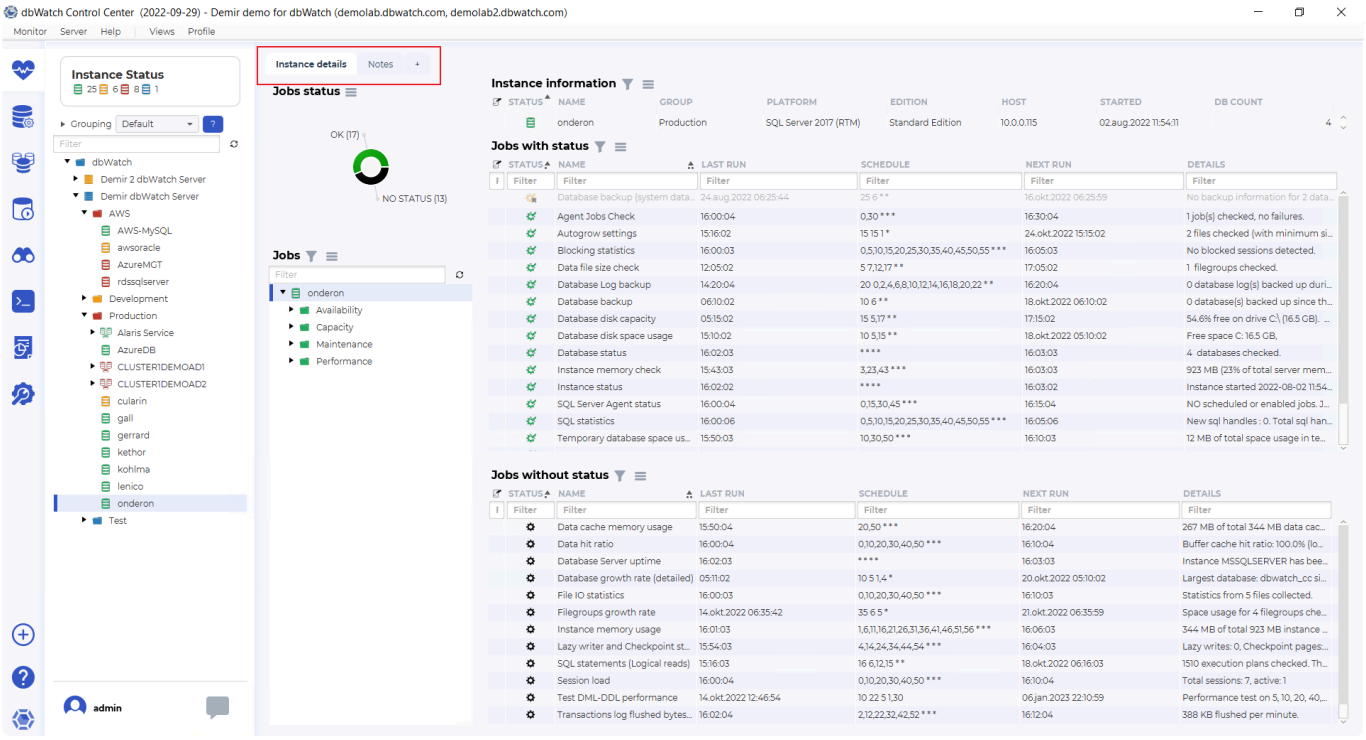
<< Monitoring / Instance details >>

Instance details

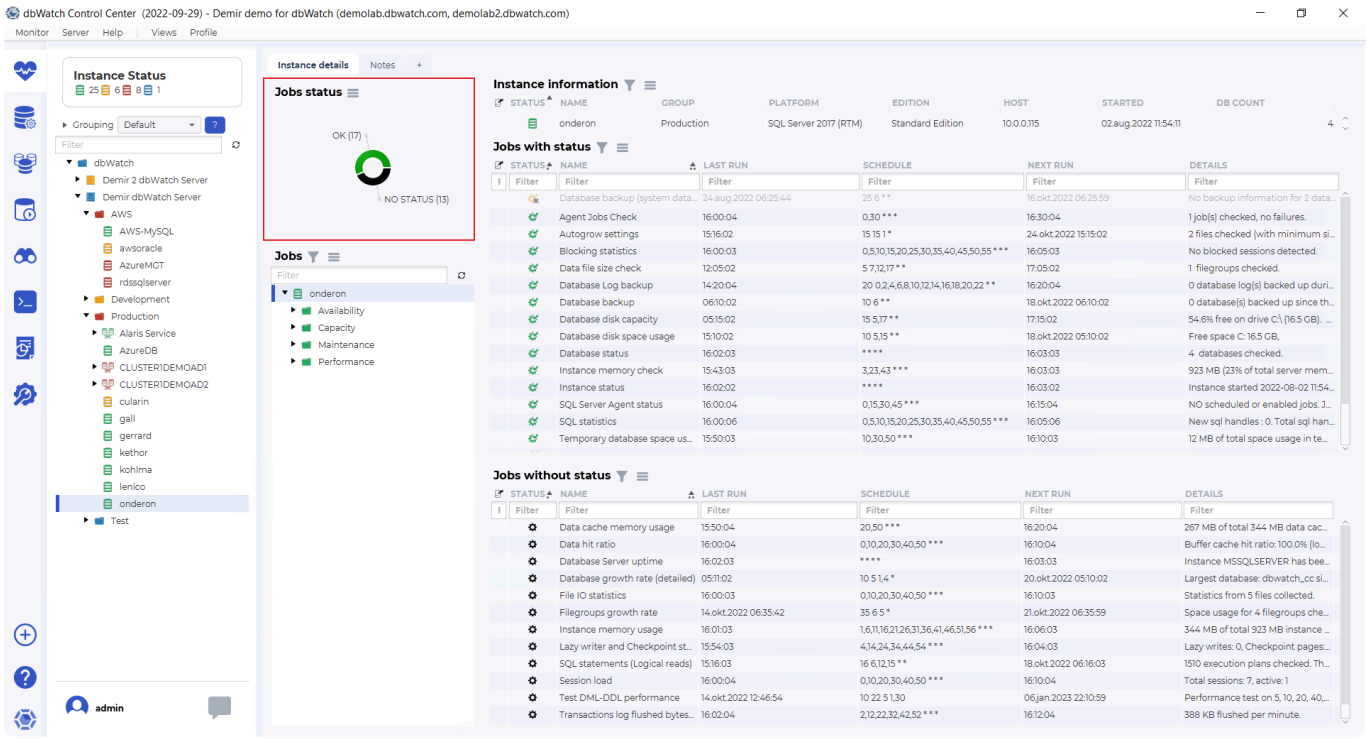
When navigating to an individual database instance, you get the instance details overview:



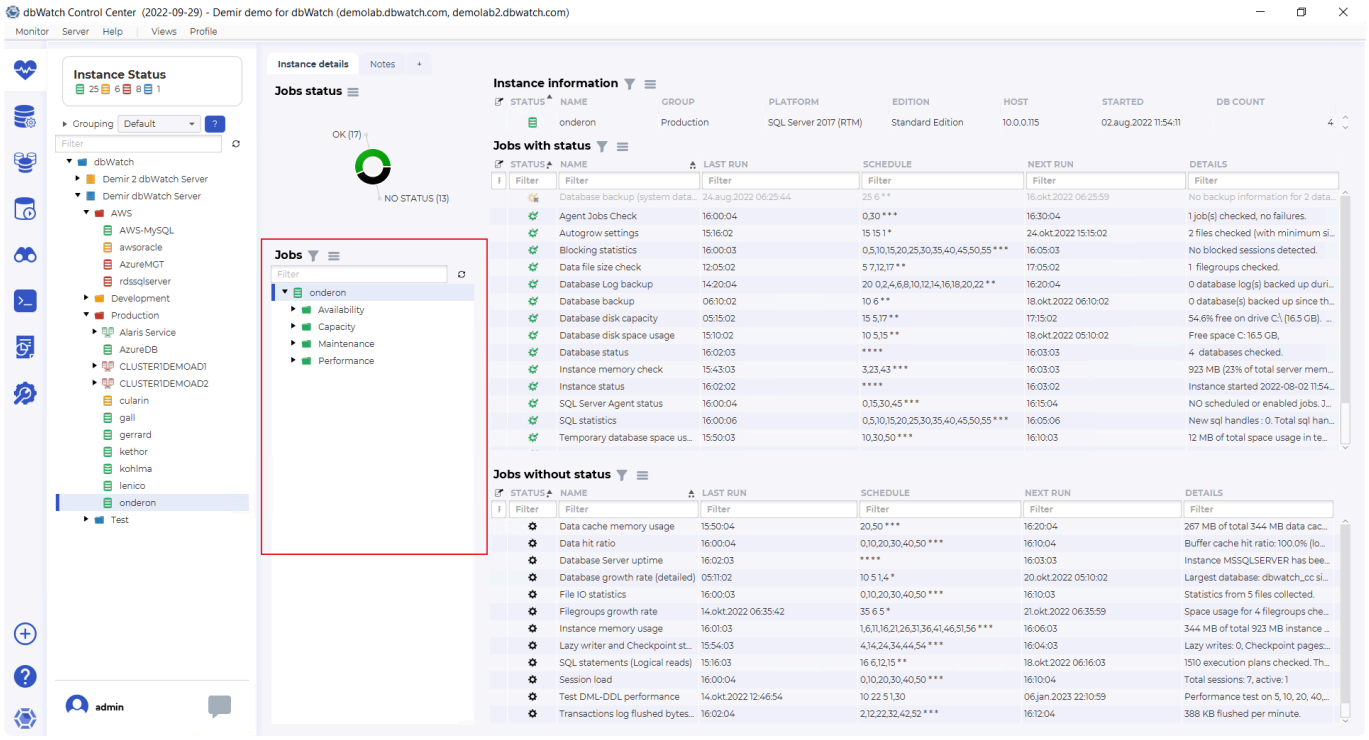
This dashboard is intended to give an overview of what monitoring jobs are installed and how they are performing.



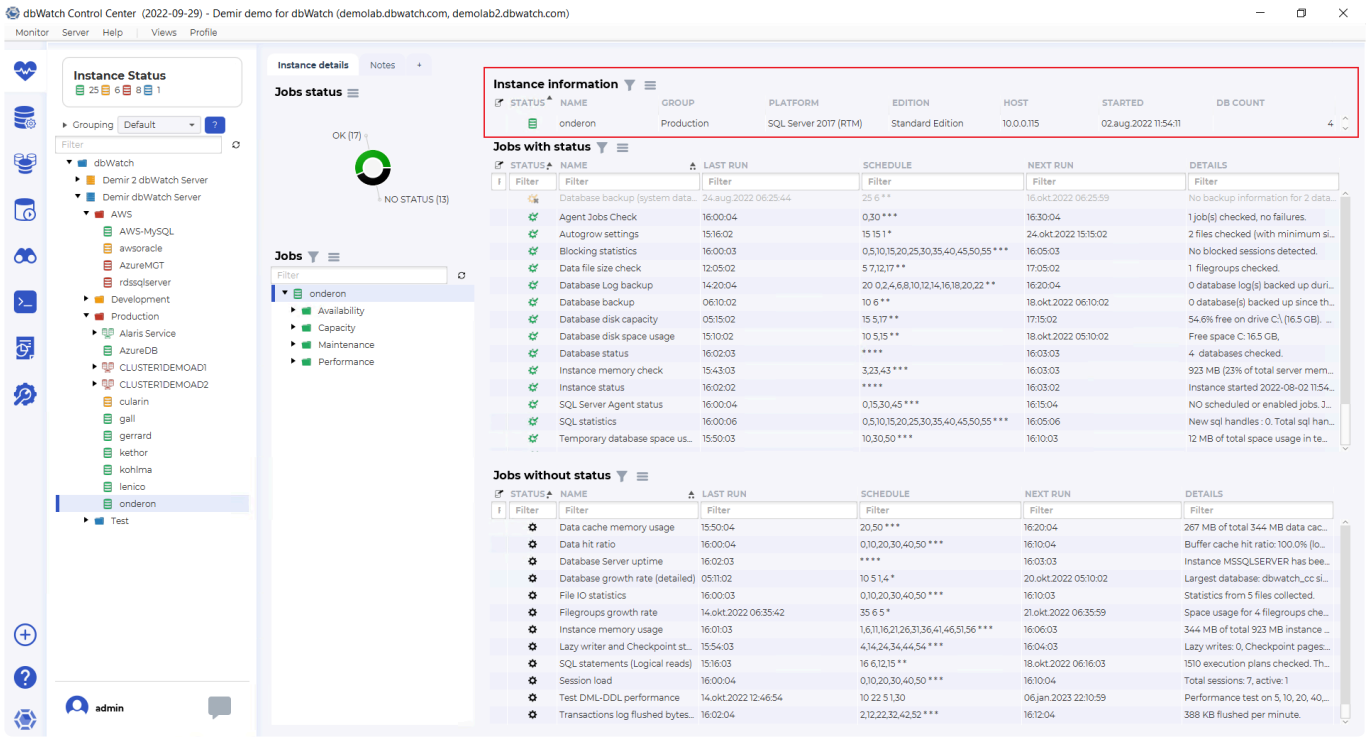
The top tabs let you switch between the instance details overview and the notes overview. The notes overview shows all notes on the instance and the jobs on that instance.



Underneath the top tabs is the jobs status piechart. This allows you to quickly see how many enabled jobs are in what status.

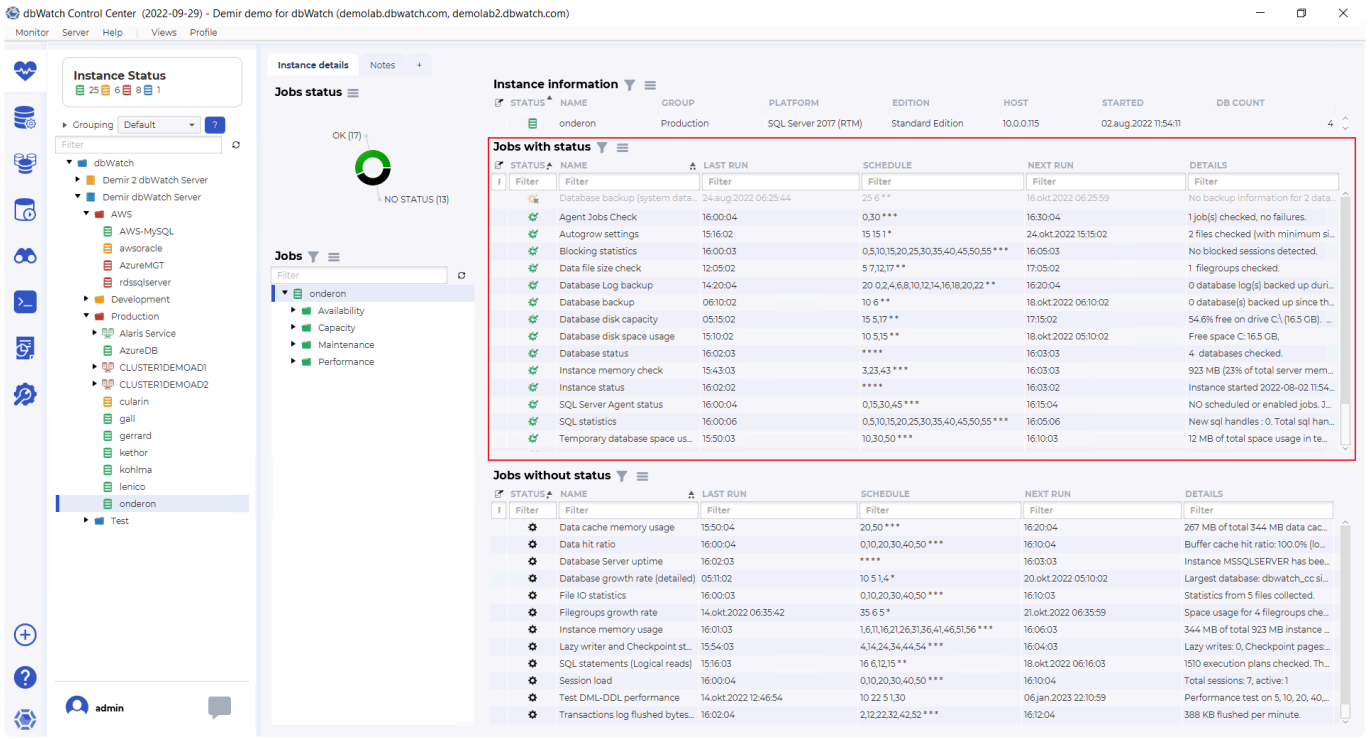


The jobs tree allows you to navigate to the individual jobs. The jobs are grouped together, in groups like Availability, Capacity, Maintenace, and Performance.



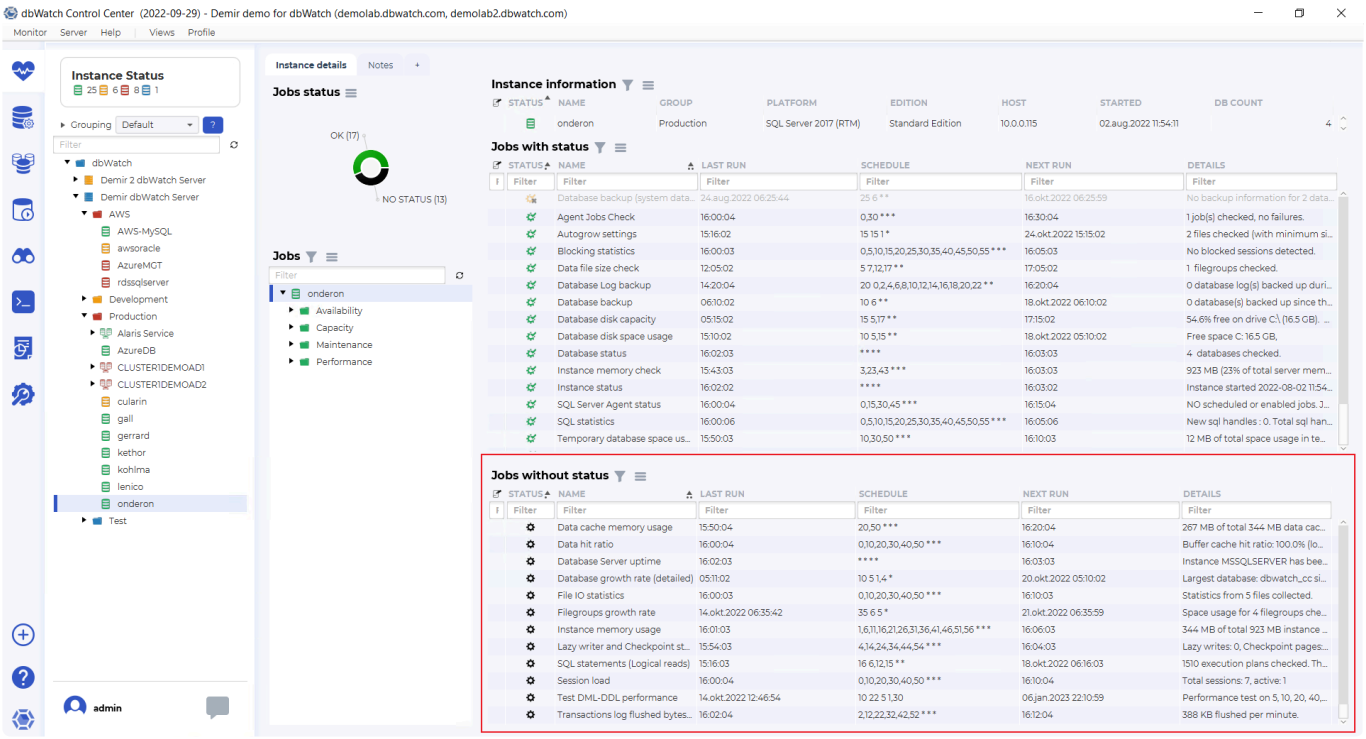
On top we have the instance information table, with the fields “Instance note indicator”, “Status”, “Name”, “Platform”, “Edition”, “Host”, Started” and “DB count”.

The “Instance note indicator”, will show an icon when someone has created a note on this instance. Notes are used to store permanent and semi-permanent information about a database instance.
The “Status” icon, indicates the current worst status of the jobs on this database instance.
The “Name” field, is the name you have given to the database instance when adding it to dbWatch. This can be changed and adjusted later.
The “Platform” field, is the database type and version.
The “Edition” field, is the database edition type.
The “Host” field, is the hostname or IP address configured.
The “Started” field, is the last time this database instance has started up.
The “DB count” field, is the current database count in this database instance.



Jobs in Control Center monitor a specific metric and can also store historical data, that is later utilized in reports and dashboards.
We distinguish between jobs with status and jobs without status. Jobs with status are jobs that monitor a metric that also creates warnings and alarms.
The jobs with status overview have the fields “Job note indicator”, “Status”, “Name”, “Last run”, “Schedule”, “Next run”, and “Details”.

The “Job note indicator”, will show an icon when someone has created a not on this job on this instance. Notes on jobs are used to store permanent and semi-permanent information about a job on a database instance, such as known issues and problems.
The “Status” icon indicates severity, yellow for warnings, and red for alarms. If there is a black x on the icon, this indicates unacknowledged historical alarms or warnings.
The “Name” is the name of the monitoring job that has found an issue. The jobs are named descriptively so it should be easy to understand what metric they are monitoring.
The “Last run” field is the time this job was run last.
The “Schedule” field is the current schedule for this job.
The “Next run” field indicates the next time this monitoring job is scheduled to run.
The “Details” field is used to show an error message regarding the problem detected. More in-depth information is available if you right-click on the issue line, and choose details.



The jobs without status are used to gather historical or trend data about a metric. They provide their own report and are also used to provide data for reports historical data for management.
The jobs without status overview have the fields “Job note indicator”, “Status”, “Name”, “Last run”, “Schedule”, “Next runtextileRef:385862163672220e80a31e:linkStartMarker:”, and “Details”.
The “Job note indicator”, will show an icon when someone has created a not on this job on this instance. Notes on jobs are used to store permanent and semi-permanent information about a job on a database instance, such as known issues and problems.
The “Status” icon in jobs without status is always black.
The “Name” is the name of the monitoring job that has found an issue. The jobs are named descriptively so it should be easy to understand what metric they are monitoring.
The “Last run” field is the time this job was run last.
The “Schedule” field is the current” schedule”:#jobs-schedule for this job.
The “Next run” field indicates the next time this monitoring job is scheduled to run.
The “Details” field is used to show an error message regarding the problem detected. More in-depth information is available if you right-click on the issue line, and choose details.

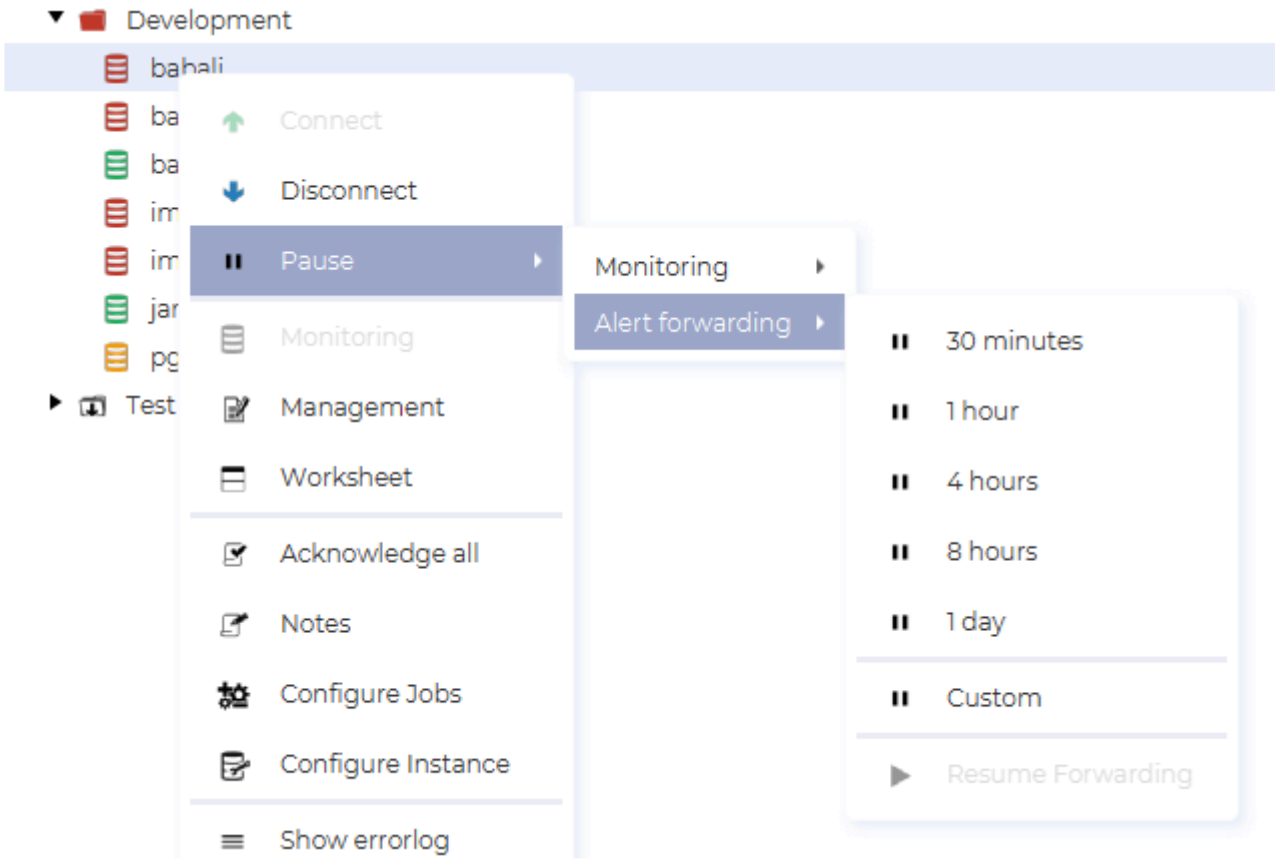
We can also go into” further details for each job. “:#job-details

<< Tree navigation / Pause >>

Pause

Monitoring in dbWatch have two “levels” of pause. You can pause the monitoring on an instance altogether, meaning that the monitoring jobs will not execute. Or you can pause the alert forwarding, meaning that the jobs will execute but dbWatch will not send any alarms through email or 3 party integrations.

Pause alert forwarding

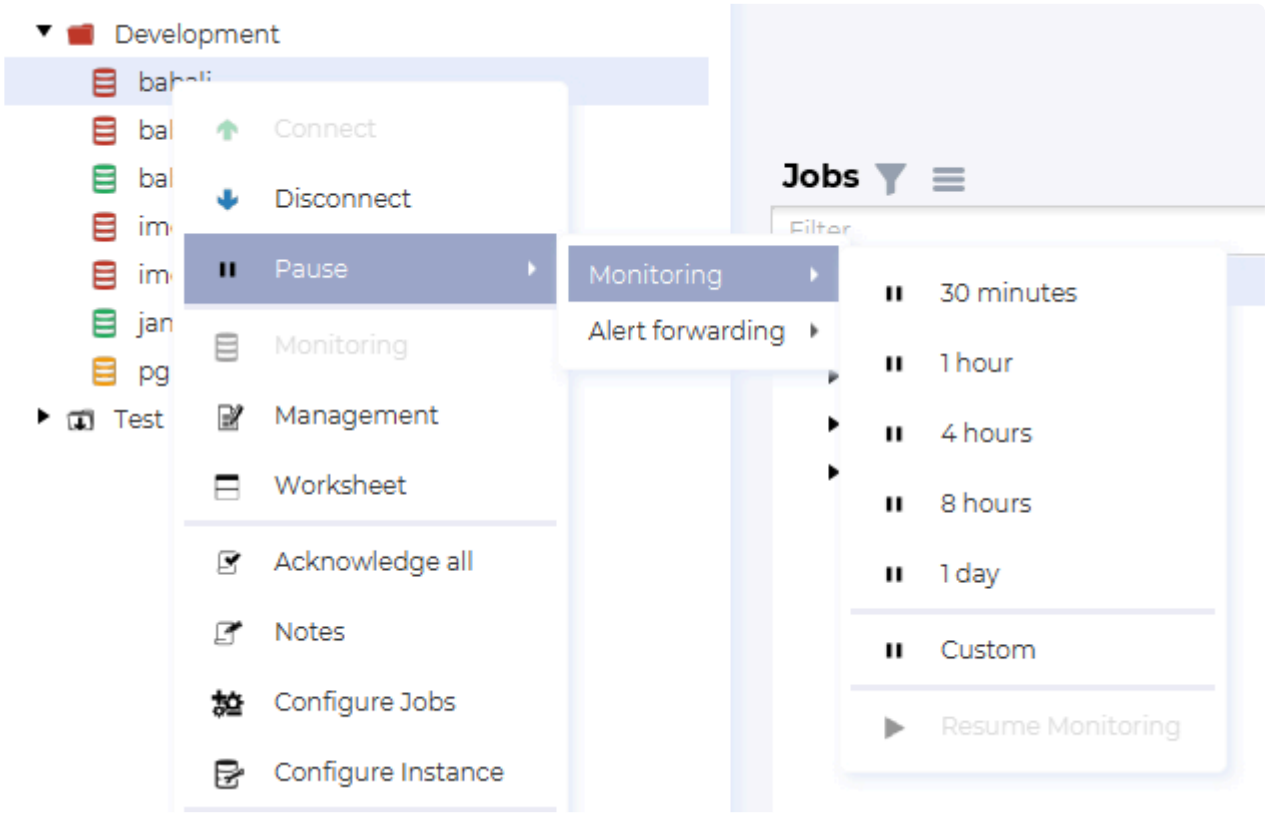


You can pause alert forwarding by right clicking on an instance and selecting the **Pause -> Alert forwarding** menu. Here you can start a pause from now until a predetermind period has passed, or you can create a custom period. If a pause is active, you can end it by selecting **Resume forwarding**.

Monitoring pause

There are two types of monitoring pauses. A “one time” pause can be started at any time and will last a specified time. A scheduled monitoring pause is a recurring pause scheduled with a crontab style definition.

One time monitoring pause (now)

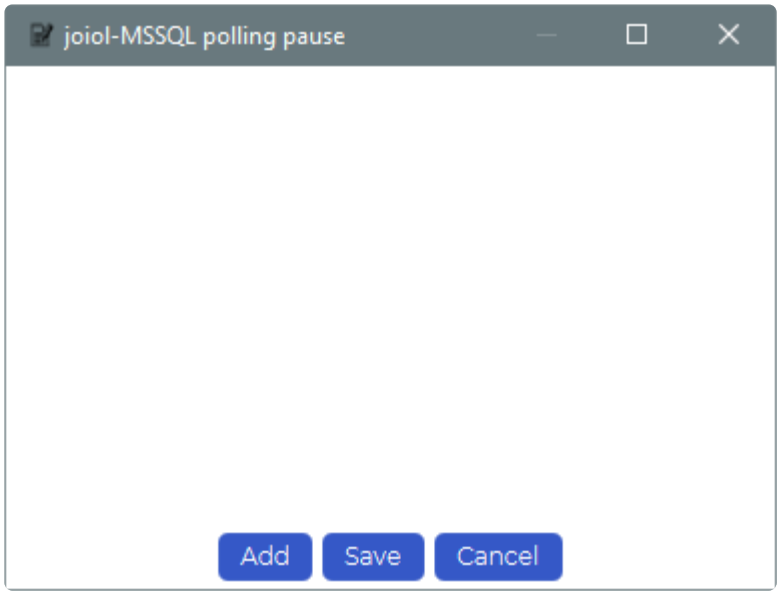


You can pause monitoring now by right clicking on an instance and selecting the **Pause -> Monitoring** menu. This will start a monitoring pause from now and until the selected time has passed. You can stop the pause by selecting Resume monitoring.

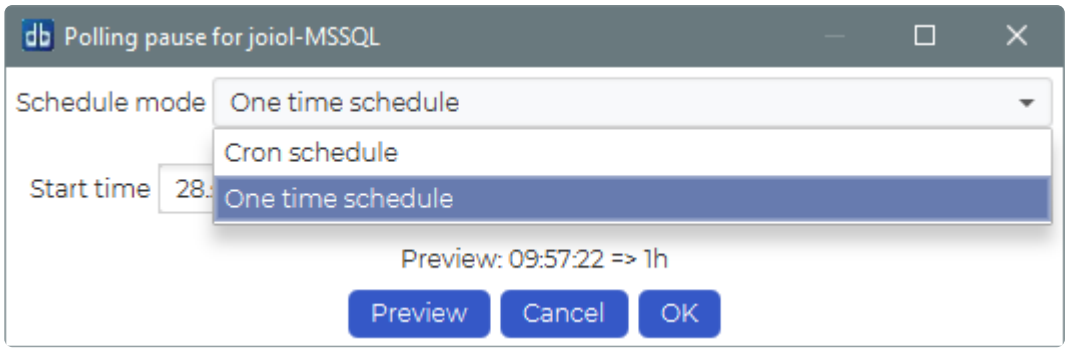
Scheduled monitoring pause

To schedule a monitoring pause, open the instance configuration view and find the **Monitoring** section under the **General** tab, then click on “Edit”

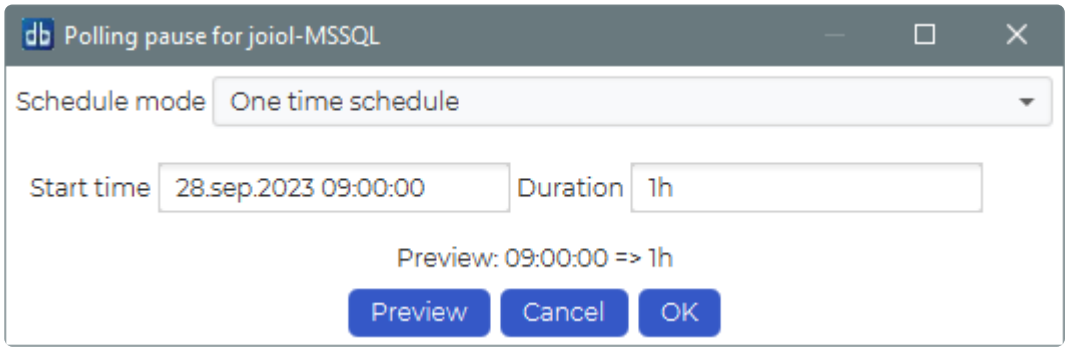
This will open the polling pause list view for the instance. Select “Add”.



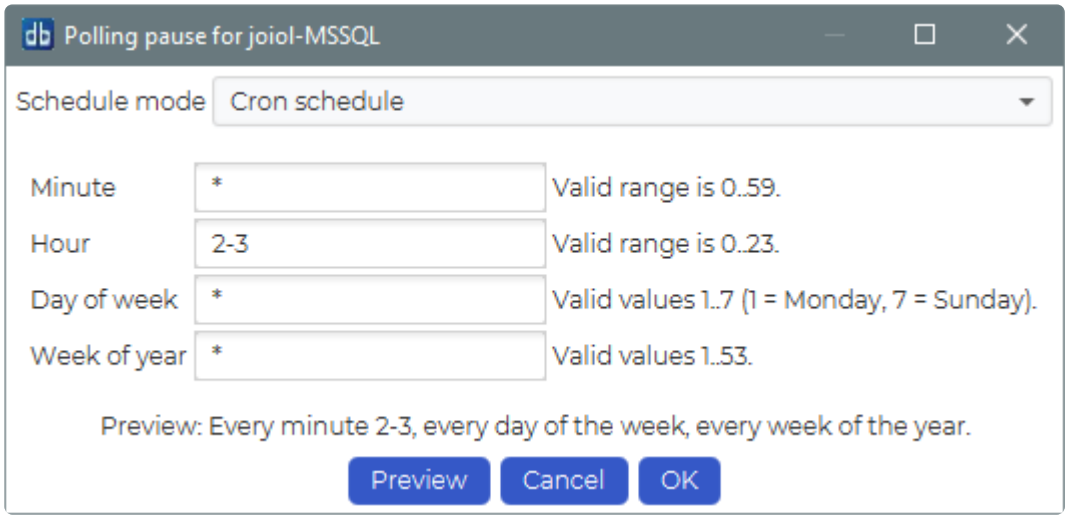
You can choose between scheduling a one time pause or a cron scheduled pause.



A one time pause is from a specified start time and for a specified duration.



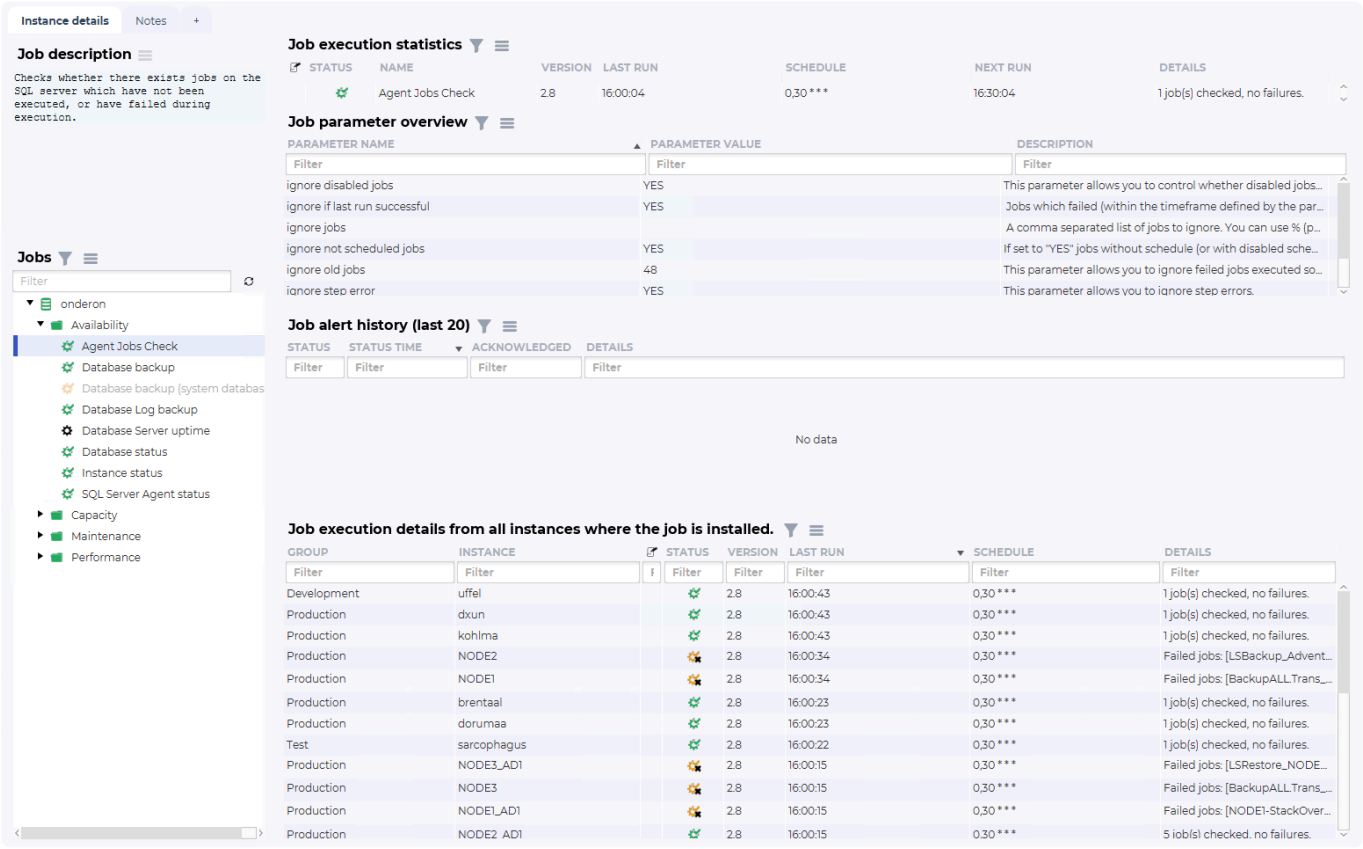
A cron scheduled pause is lets you sepcify a recurring pause. The example below specifies a pause every minute of the hours 2 and 3 (02:00 AM to 03:59 AM), every day of the week, every year.



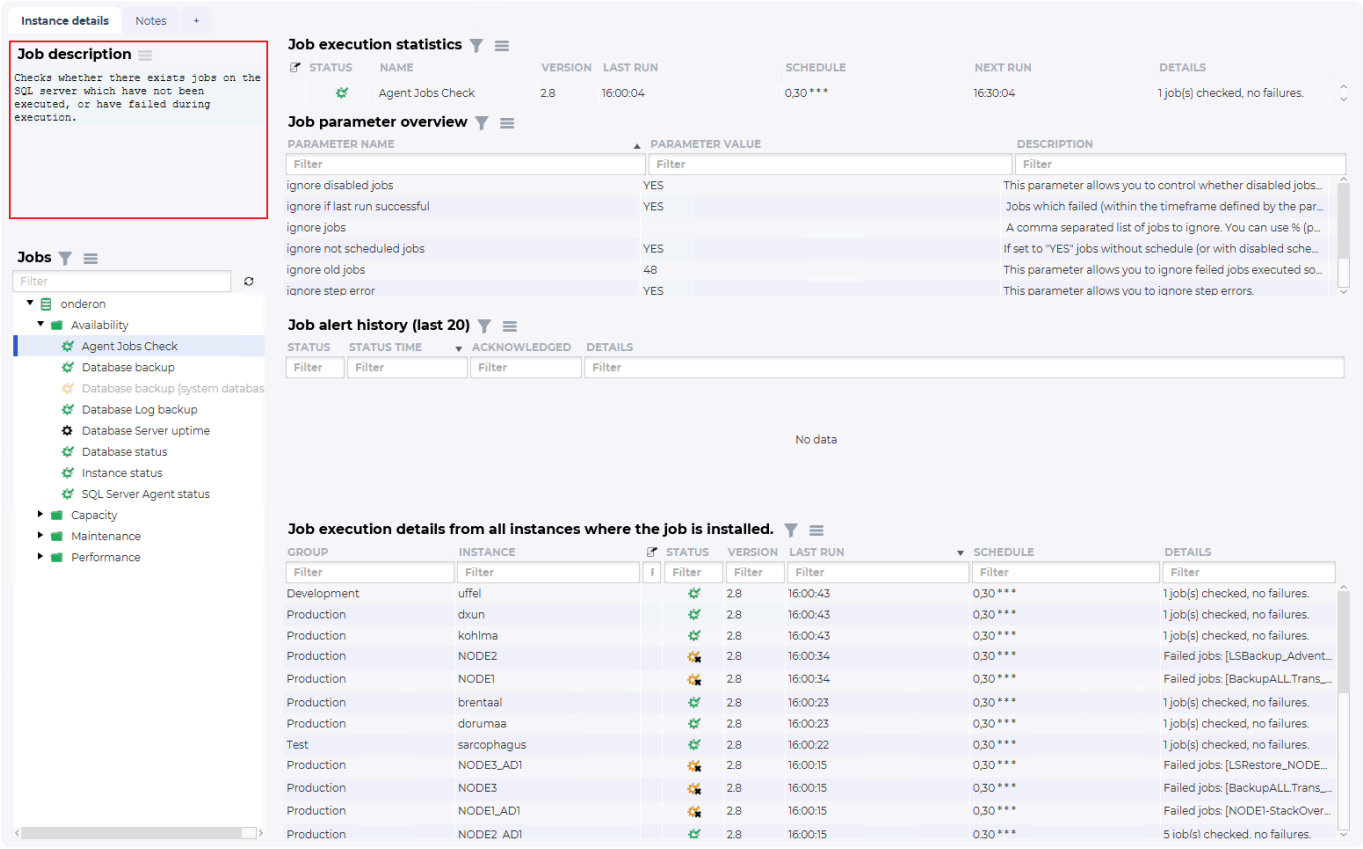
<< Instance details / Job details >>

Job details

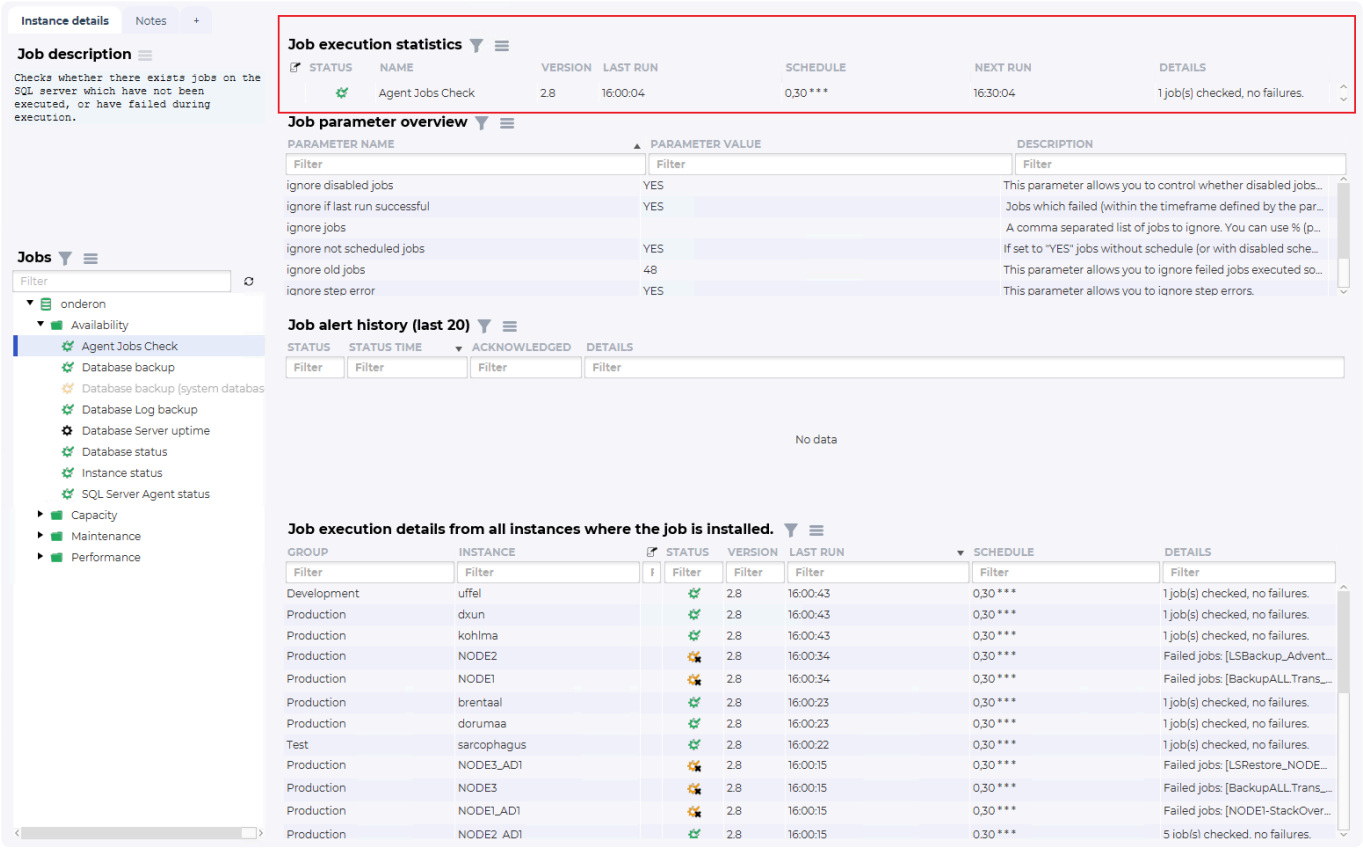
When you click on one of the jobs in the job tree, you can see a more detailed job overview.



The idea of this overview is to show current and historical state, configured parameters and how the job is behaving on other database instances.

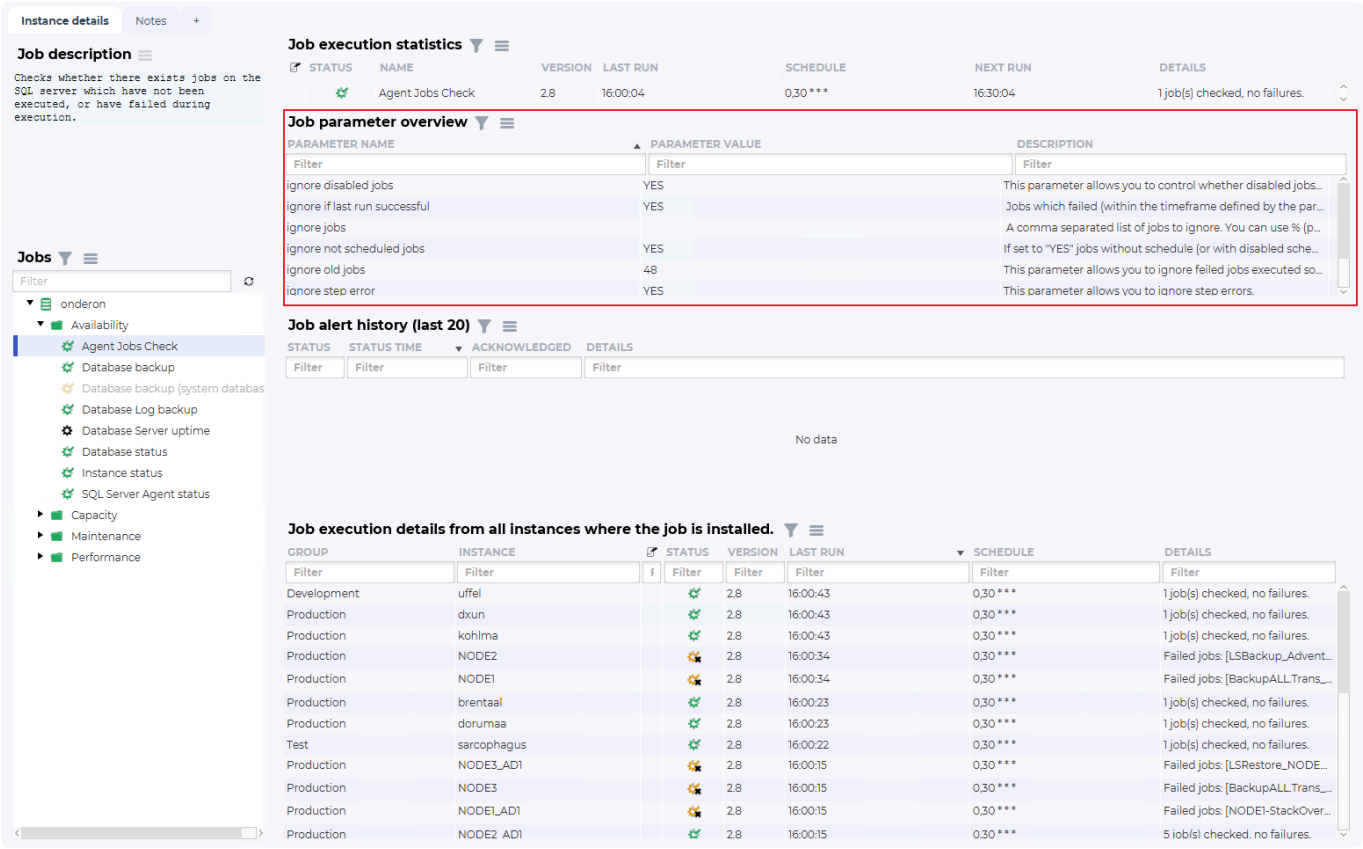


In the top left corner, there is a description of this job and what it does.



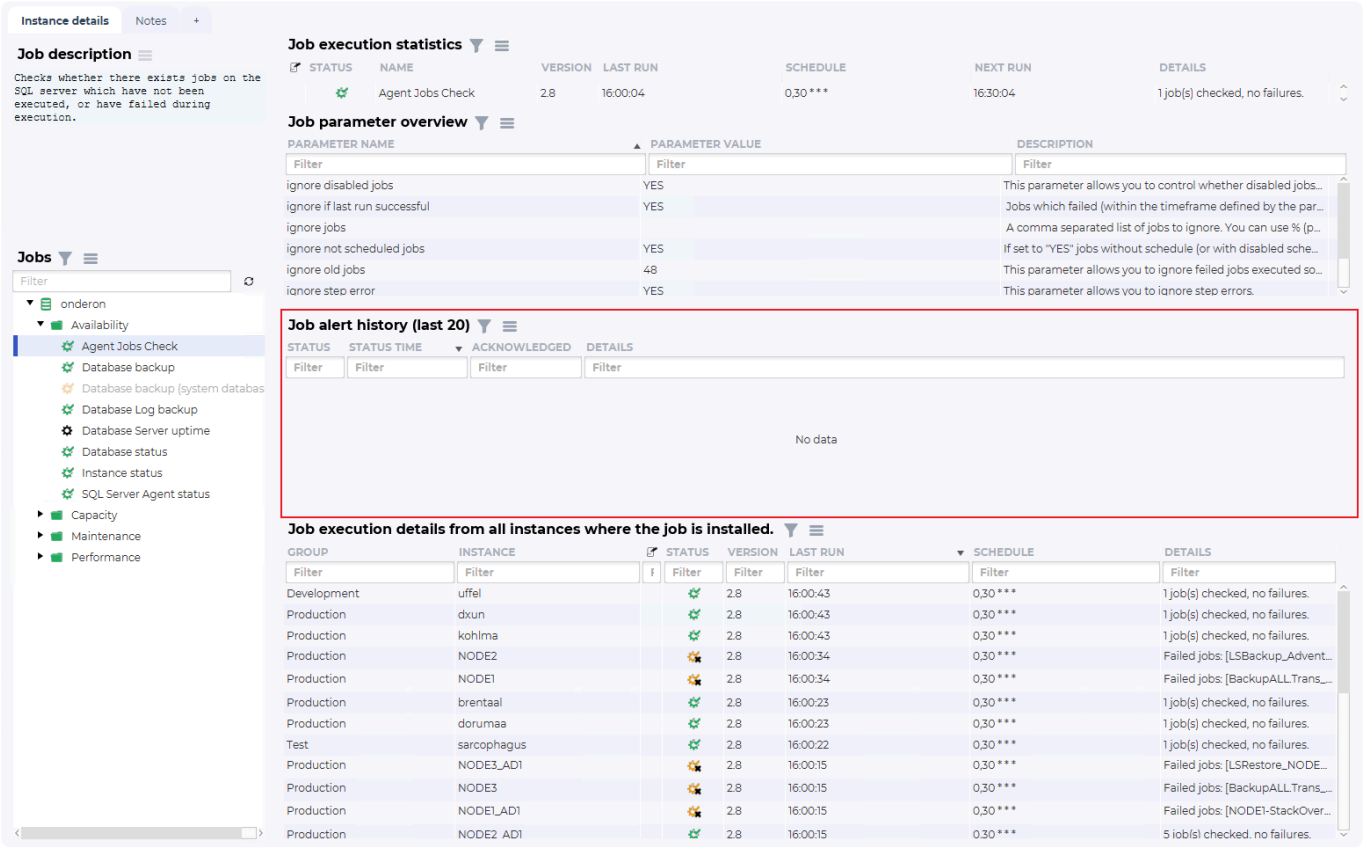
The job execution statistics provide the current status for this job. The fields are “Job note indicator”, “Status”, “Name”, “Version”, “Last run”, “Schedule”, “Next run”, and “Details”.

The “Job note indicator”, will show an icon when someone has created a not on this job on this instance. Notes on jobs are used to store permanent and semi-permanent information about a job on a database instance, such as known issues and problems.
The “Status” icon indicates severity, yellow for warnings, and red for alarms. If there is a black x on the icon, this indicates unacknowledged historical alarms or warnings.
The “Name” is the name of the monitoring job that has found an issue. The jobs are named descriptively so it should be easy to understand what metric they are monitoring.
The “Version” field is the currently running version of this job.
The “Last run” field is the time this job was run last.
The “Schedule” field is the current schedule for this job.
The “Next run” field indicates the next time this monitoring job is scheduled to run.
The “Details” field is used to show an error message regarding the problem detected. More in-depth information is available if you right-click on the issue line, and choose details.

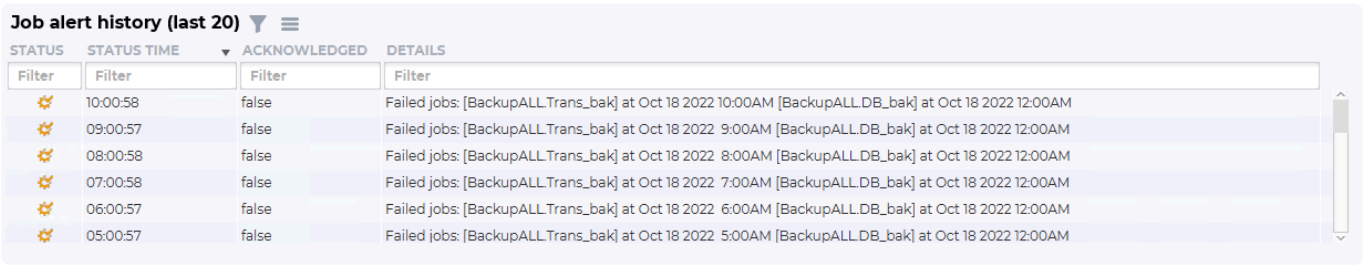


The job parameter overview provides insight into what parameters are configured for this job on this database instance. Each job can have parameters that allow users to control thresholds and behavior of a job to better align with the setup of the database instance and the preferences of the DBA. The fields are “Parameter name”, “Parameter value” and “Description”.

The “Parameter name” field is the descriptive name of the parameter and what it adjusts.
The “Paratemer value” field is the current setting for this parameter.
The “Description” field is the parameter description, that gives the user information about what values are correct for this parameter and what it adjusts.

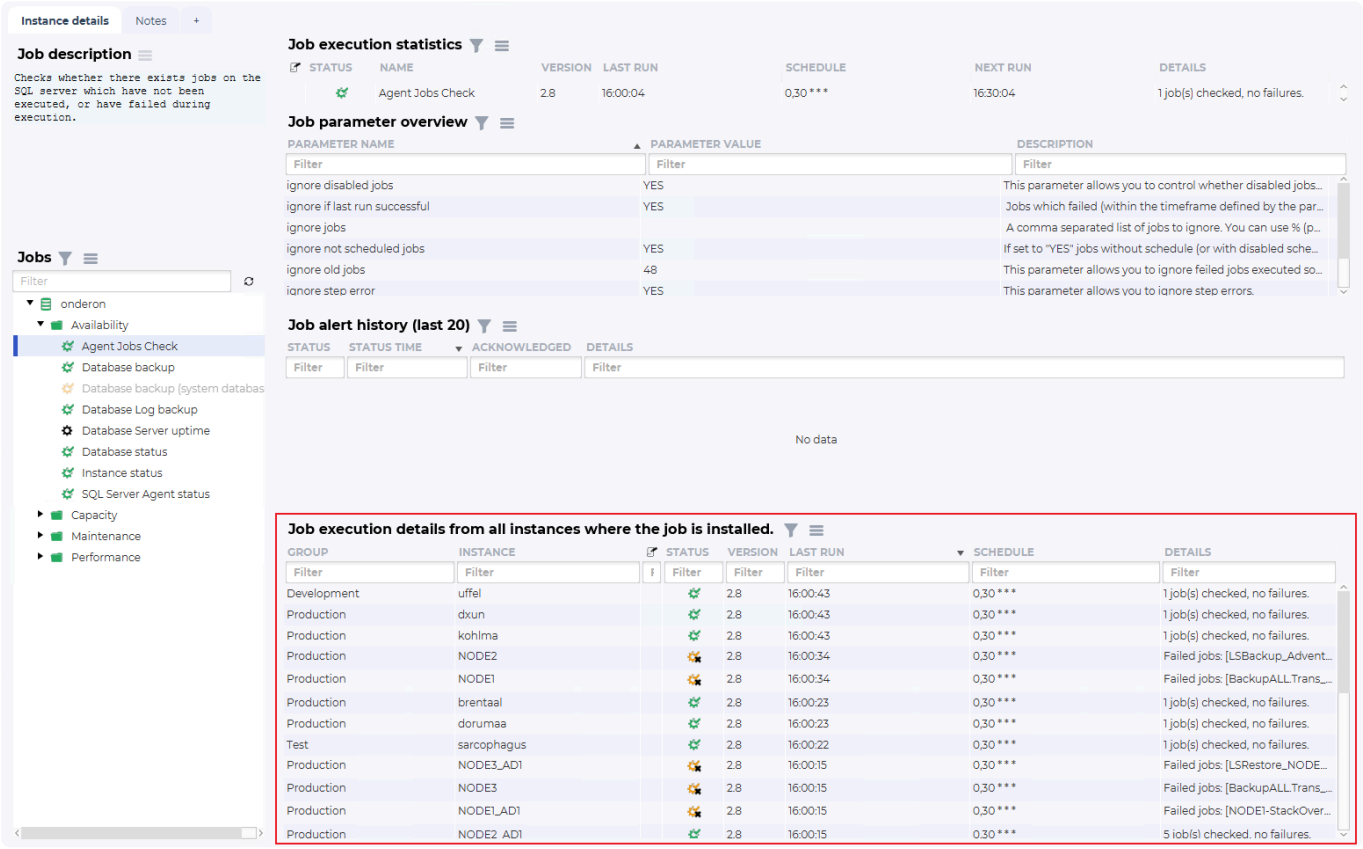


The job alert history (last 20) shows the historical errors this job has had on this instance. It can be empty, as in this example, and show up with “No data”



Or contain a long list of alerts generated. This overview has the fields “Status”, “Status time”; “Acknowledged” and “Details”.

The “Status” icon indicates severity, yellow for warnings, and red for alarms.
The “Status time” field shows the time this alert was created.
The “Acknowledged” field shows if this error has been acknowledged not.
The “Details” field is used to show an error message regarding the problem detected.



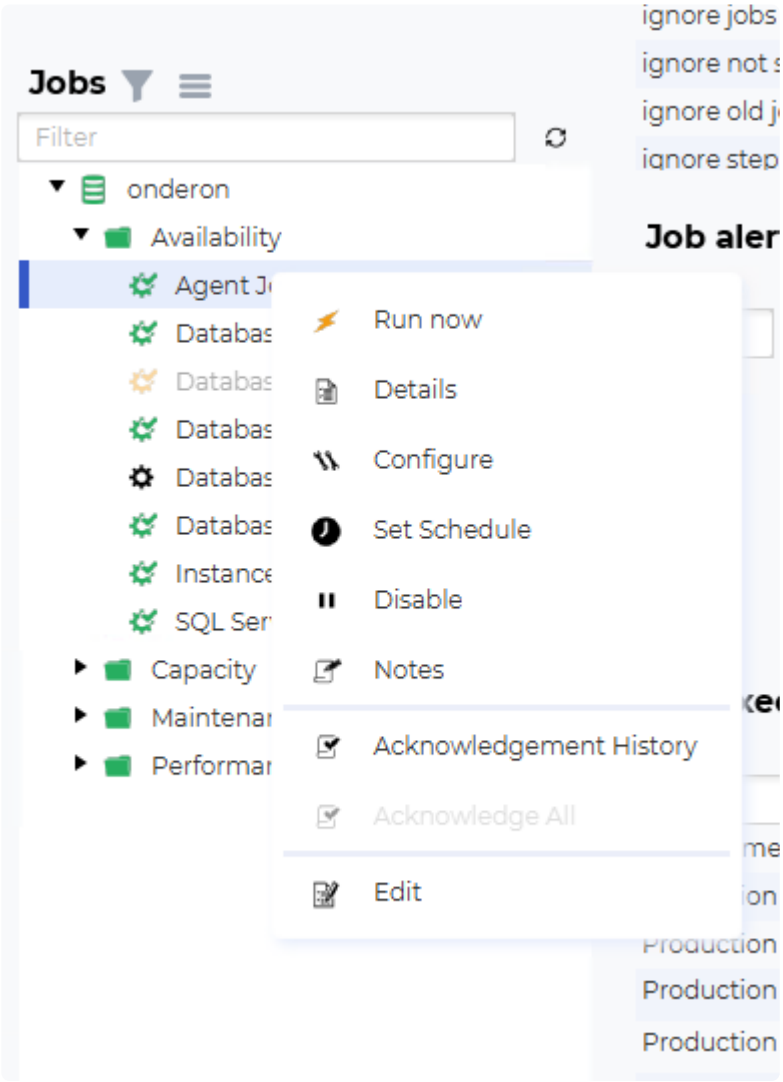
The Job execution details from all instances where the job is installed has a very descriptive name. Its used to see how the same monitoring job is deployed and its status on other database instances. There are several fields: “Group”, “Instance”, “Job note indicator”, “Status”, “Version”, “Last run”, “Schedule”, and “Details”.

The “Group” field is the instance group for the database instance in the next field.
The “Instance” is the name you have given to the database instance when adding it to dbWatch.
The “Job note indicator”, will show an icon when someone has created a not on this job on this instance. Notes on jobs are used to store permanent and semi-permanent information about a job on a database instance, such as known issues and problems.
The “Status” icon indicates severity, yellow for warnings, and red for alarms. If there is a black x on the icon, this indicates unacknowledged historical alarms or warnings.
The “Version” field is the currently running version of this job.
The “Last run” field is the time this job was run last.
The “Schedule” field is the current schedule for this job.
The “Details” field is used to show an error message regarding the problem detected. More in-depth information is available if you right-click on the issue line, and choose details.

<< Pause / Job menus >>

Job menus

Each job has a set of menus. They can be accessed if you right-click on a job.



This gives you several options.
“Run now” will trigger the job now, outside of the normal schedule.
“Details” will open the job report, providing details about this monitoring job and historical data about the metric it monitors.
“Configure” will open the parameter configuration.
“Set schedule” will allow you to change the schedule this database instance.
“Disable” (or “Enable” if it is already disabled) will turn off (or on) the scheduling and alert propagation. This means it will be greyed out, displaying last status, but the status will not be propagated to the job group, instance, instance group, or global status level.
“Notes” will open the notes editor
“Acknowledgement history” will open the acknowledgment history overview, showing previous warnings and alarms.
“Acknowledge All” will set all old warnings and alarms as acknowledged.
“Edit” will open the job editor. This will be greyed out if the job type does not have an editor.

<< Job details / Job concepts >>

Job concepts

This section will try to explain more in-depth some of the concepts used in monitoring jobs in Control Center.

There are currently two types of jobs, engine jobs, and no-engine jobs. The main difference between them is where processing occurs.
Engine jobs do the processing in the database instance engine, and no-engine jobs do the processing on the Control Center Instance hub.

They both share concepts such as status, version, schedule, and acknowledgment system.

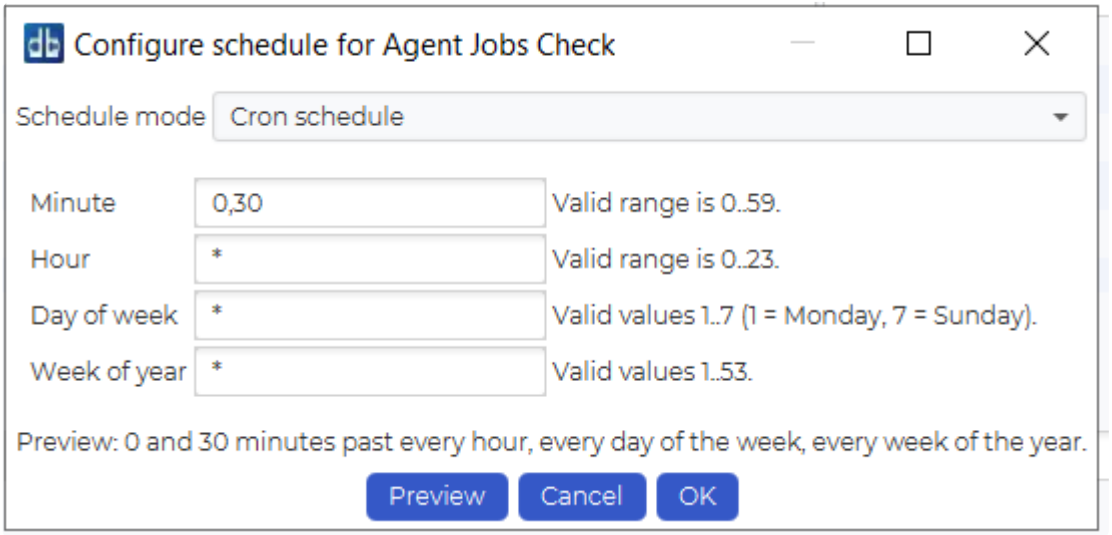
More on the job types here.

<< Job menus / Schedule >>

Schedule

The scheduling in Control Center has two modes, cron schedule, and interval schedule.

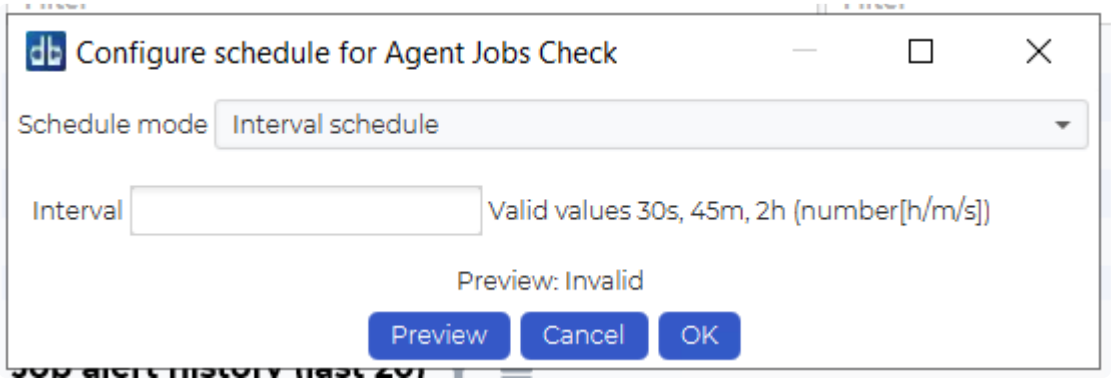
Cron schedule



The cron schedule is loosely based on the crontab format used on Unix systems. We have 4 fields that can be set, “Minute”, “Hour”, “Day of week”, and “Week of year”.
The “Minute” field can have a number between 0 and 59, indicating the minute past the hour this will be scheduled. It can also have a list of numbers with commas between them to run at several specific minute values, or a * to indicate every minute.
The “Hour” field can have a number between 0 and 23, indicating the hour this will be scheduled. It can also have a list of numbers with commas between them to run at several specific hour values, or a * to indicate every hour.
The “Day of week” field can have a number between 1 and 7, indicating the day of the week. Monday is 1, Tuesday is 2, Wednesday is 3, Thursday is 4, Friday is 5, Saturday is 6 and Sunday is 7. It can also have a list of numbers with commas between them to run at several specific day values, or a * to indicate every day of the week.
The “Week of year” field can have a number between 1 and 53, indicating the week of the year. It can also have a list of numbers with commas between them to run at several specific “week of year” values, or a * to indicate every “week of the year”.

Clicking on the preview button evaluates values and writes them out in a more human-readable form.

Interval schedule

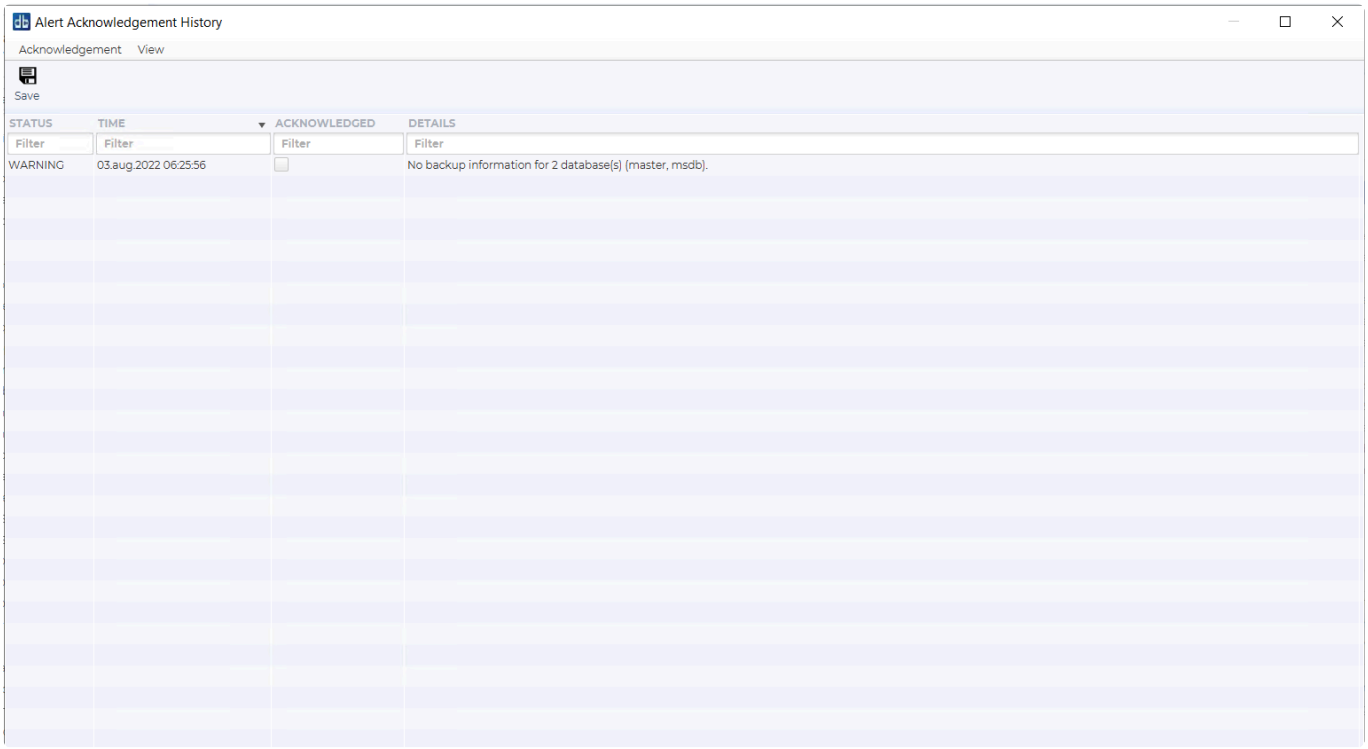


The interval schedule can schedule a job at repeating, but not exact times. Values are a number with a time denominator, such as 30m (every 30 minutes) or 3h (every 3 hours).
Values can range from 1 to the max value for int, but scheduling more frequently than around 15s will not be accurate due to limitations in the scheduling timing code.
The interval scheduling will also, by design, drift slightly to sparse out job scheduling across the database instance farm.

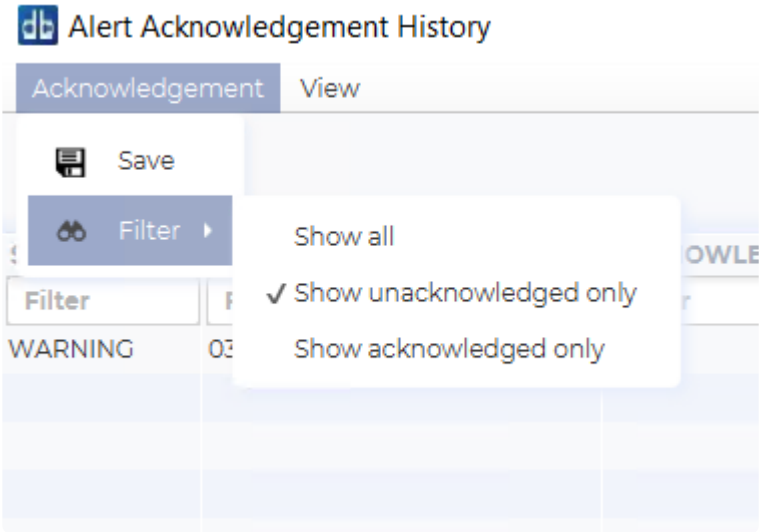
<< Job concepts / Acknowledgement system >>

Acknowledgement system

The acknowledgment system is designed to make sure that jobs that have had alarms in the past, but now have changed status, is known to the DBA.



By default we list all un-acknowledged warnings or alarms in the “Alert Acknowledgement History” view. If you tick the “Acknowledged” radio-button, and click save, the warning will disappear from the list. If you use the “Acknowledge all” button in the job menu, all old warnings and alarms will be acknowledged at once.



You can later change the filtering of this view, if you want to see acknowledged warnings again, or check the history of warning for this job.

<< Schedule / Versioning >>

Versioning

All jobs and most resource files in Control Center have a version to differentiate between new and old jobs.

It is used to upgrade jobs, and most jobs are auto-upgraded.

Engine jobs have upgrade scripts built in, to alter tables and update changes to PL/SQL or TSQL code. No-engine jobs are self-contained and when new versions are available the latest version will be run at the next schedule.

[<< Acknowledgement system] / [Status >>]

Status

Jobs can have four different statuses. Additionally, the instance can have the “Lost connection” state, blue, that takes precedence over job statuses.

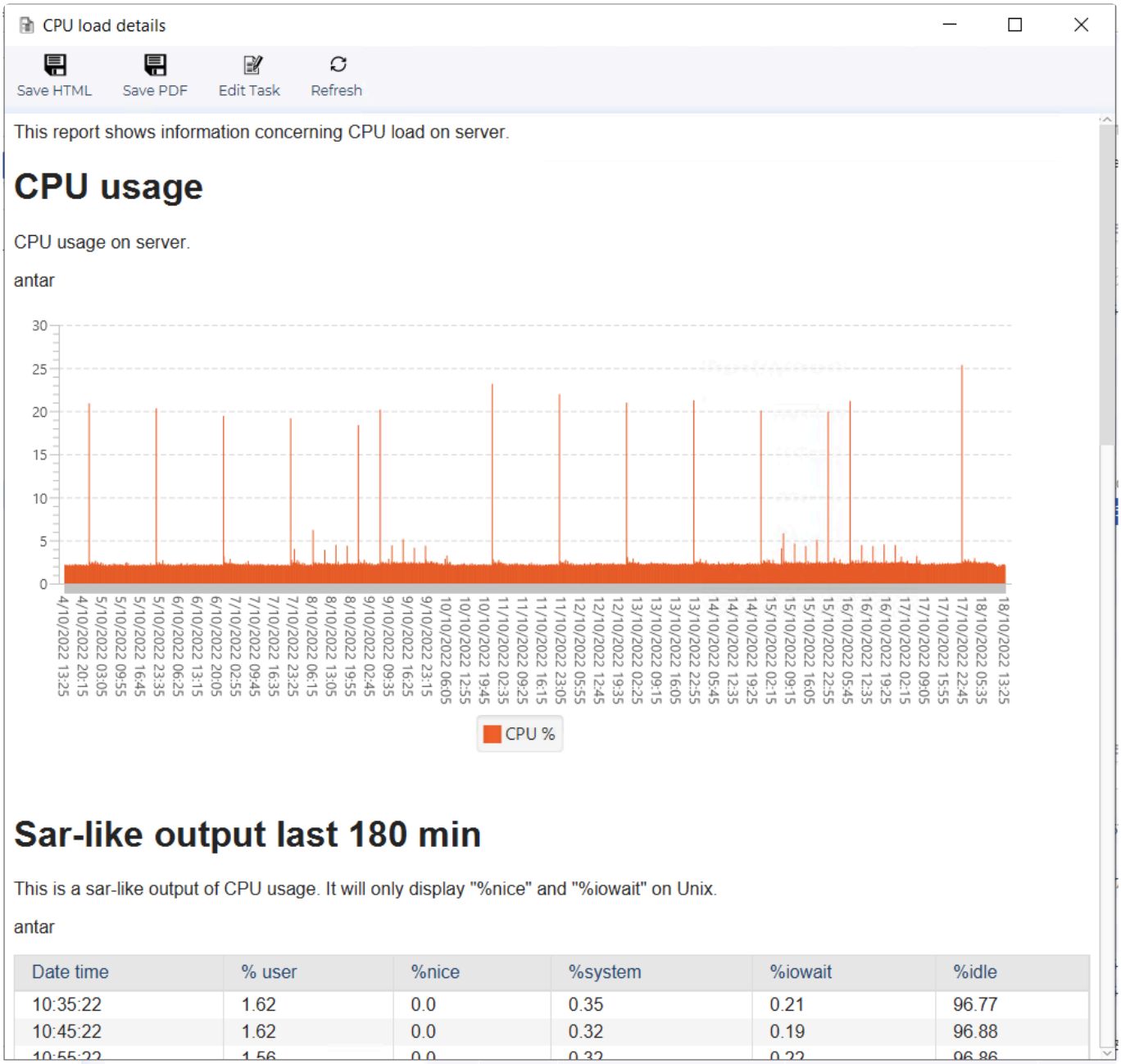
Green for OK, Yellow for Warning, and Red for Alarm. Tasks can be without status, and the no-status icon is black.

Statuses will propagate, so the worst alarm state, Alarm is worse than Warning, Warning is worse than alarm, will propagate from the job status to the job group. The job group propagates to the instance. The instance status propagates to the instance group. The instance group propagates to the instance hub or domain.

<< Versioning / Report >>

Report

Each job has a job report that is generated when you use the job menu and choose “Details”.



The job report, here an example from the CPU load details on Oracle, provides a set of graphs, pie charts, and tables.
It uses previously generated history stored in the tables of an engine job or a mix of SQL from the database or properties history for the no-engine jobs
For engine jobs, the report can be edited as part of the editor for engine jobs.
No-engine jobs not have an editor, but reports follow the same XML structure as an engine job.

The report contents can be saved as HTML and as PDF. You can also open the editor or rerun the report.

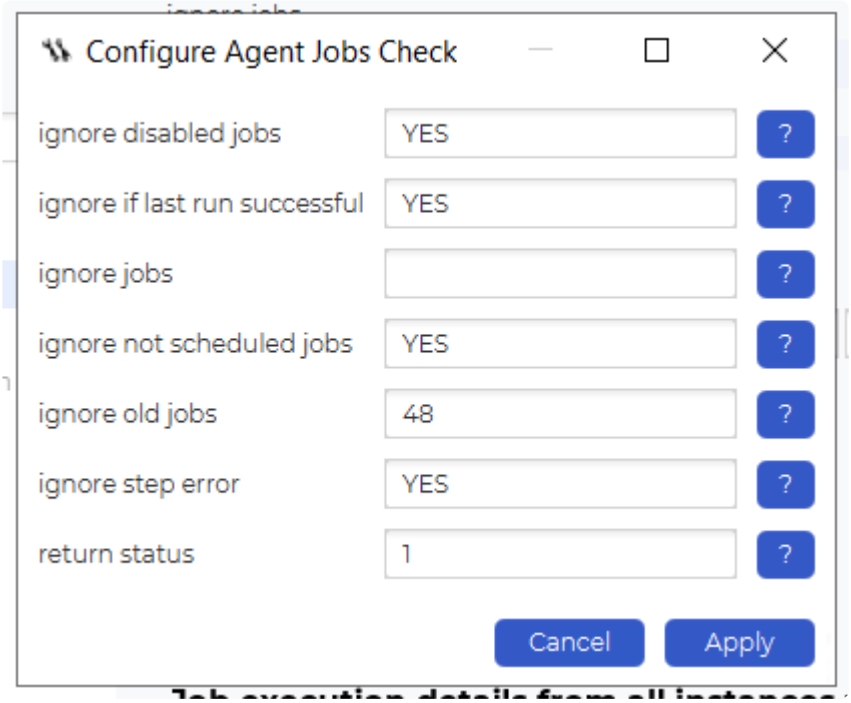
For generic reports, Control Center provides a reporting module that has more capabilities and a built-in

editor.

<< Status / Parameters >>

Parameters

Most of the monitoring jobs will have parameters. This is used controllable thresholds to adjust the monitoring job behavior and to better align it with the database instance and the preferences of the DBA.



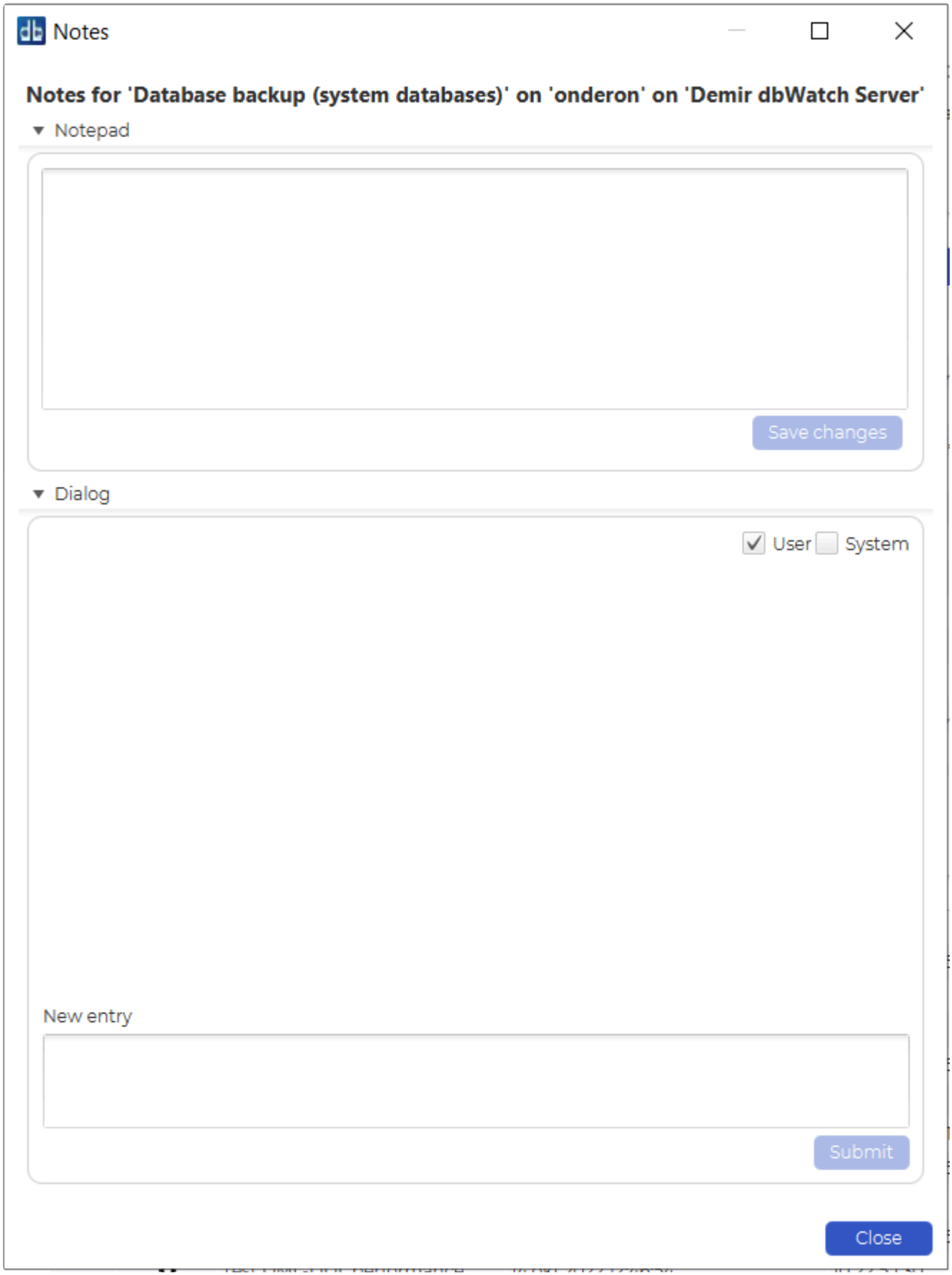
This is an example from the “Agent jobs check” that monitors agent jobs on MS SQL Server.
This dialog lets you adjust this monitoring job on only this database instance.
The layout is a descriptive parameter name, a user-configurable value, and a question mark icon.
Clicking on this icon will open a pop-up with additional information about how this parameter can be configured.

Each job comes with default values for the parameters, and changes to the parameters are written in the dbw_parameters table in the database instance for engine jobs and as metadata, if changed from defaults, if its no-engine jobs.

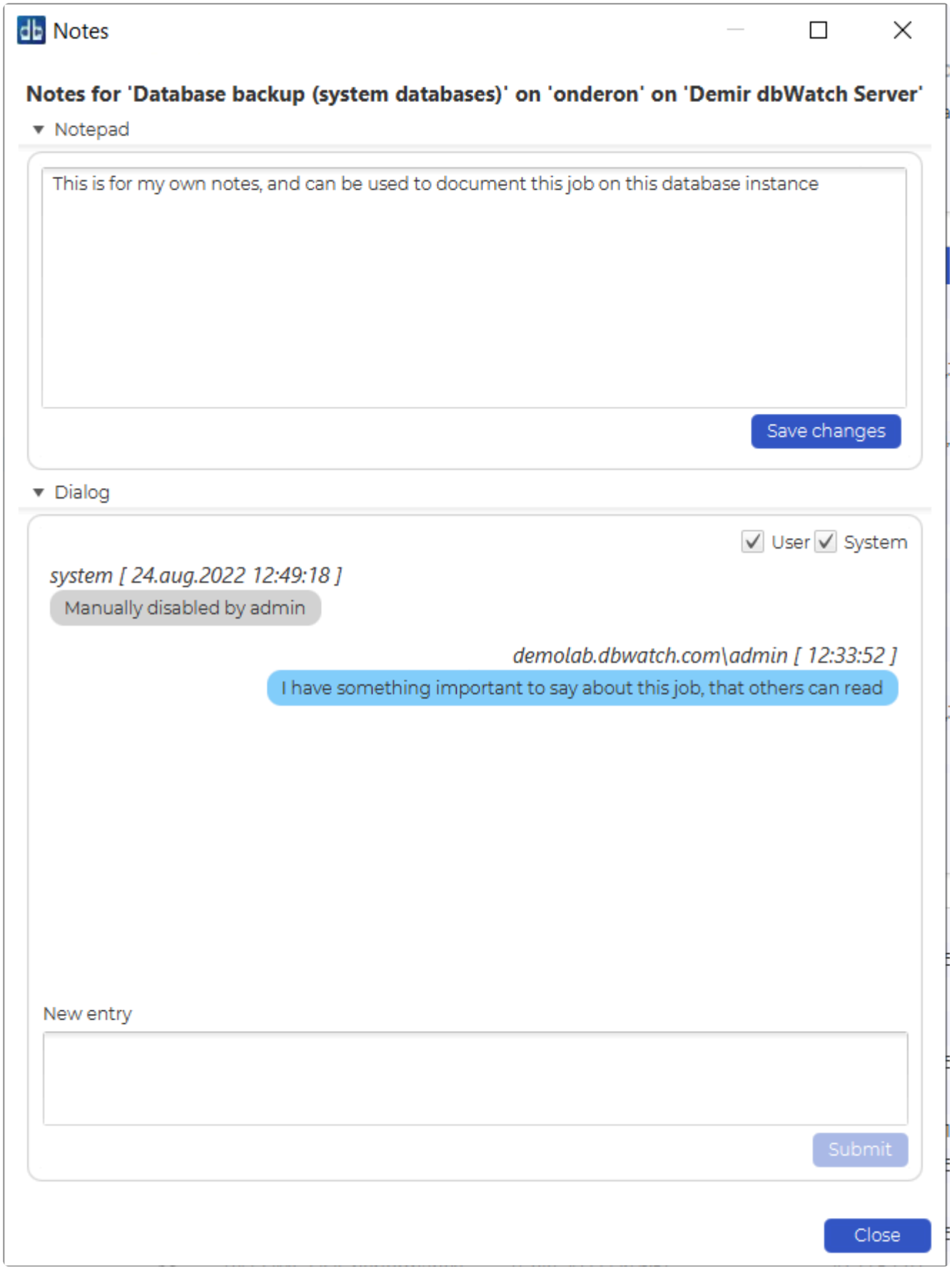
<< Report / Notes >>

Notes

Notes on jobs are information that DBA’s and the Control Center system save to better document a system setup and known behaviour.

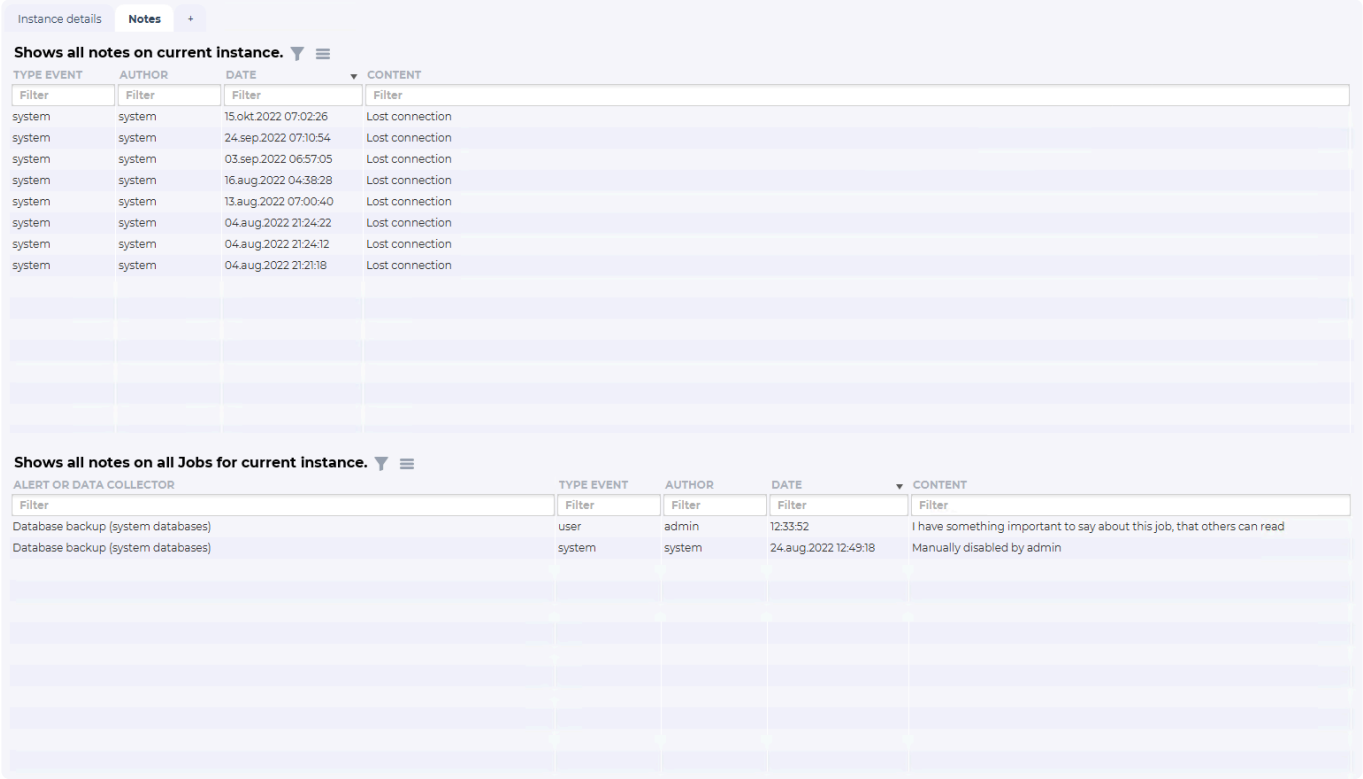


When you open notes on a job, it will start off as a blank note. The top field is for personal notes and the contents are stored in an encrypted format. The dialog is split into user and system dialogue.



You can add notes in the notepad window, it can be information you want to store on how issues this job detected in the past were fixed. In the case of an agent jobs check, steps taken to fix the issue, or how the job was re-run.
The user dialog is the communication from a DBA to other DBA's regarding this job. It is public notes,

visible to others. The system dialog is log entries that will highlight changes to this job and its schedule done by other users, such as disabling the job.



The dialog part of the job notes is also visible in the notes overview for the database instance, as part of instance details. Also the instance notes are visible in this view. And Control Center creates notes when database instances loose connection.

<< Parameters / Job types >>

Job types

There are currently two types of jobs, engine jobs, and no-engine jobs.
The main difference between them is where processing occurs.
Engine jobs do the processing in the database instance engine, and no-engine jobs do the processing on the Control Center Instance hub.

They both share concepts such as status, version, schedule, and acknowledgment system.

<< Notes / Engine jobs >>

Engine jobs

Engine jobs

Each engine Job consists of a set of objects (procedures, tables, etc) that are installed on the monitored database instance. The main procedure will be triggered by the dbWatch Server, This procedure will then perform a check or collect statistics regarding the instance and typically put data in a table that can be reported on later.

All Jobs provided by dbWatch are provided as open source. This is part of the dbWatch philosophy of mixed source. Customers are able to (and indeed we recommend) that they change the standard supplied jobs. By using our codes as the basis for their own customized ‘developments’. They can then customize their jobs to suit their specific needs. dbWatch Software Support is available to assist with customized developments.

<< Job types / No-engine jobs >>

No-engine jobs

No-engine jobs

No-engine jobs consists of a series of steps (defined in xml files), and each step retreives data and/or performs some computation on the data.
The last step will also set a status and a text describing the current state of the job.

The steps can be a mixture of different languages, currently sql, fdl, javascript and ssh are supported.

Each step has the results from the previous steps available as input.

The basic jobs do not require any framework or objects installed on the databases, all state is stored on the dbWatch Server.

[<< Engine jobs] / [Control Center Jobs >>]

Control Center Jobs

Jobs (older dbWatch versions Tasks and alerts) are the cornerstone of the dbWatch Monitoring module.

Jobs perform the actual monitoring of the database instance.

There are two types of Monitoring in dbWatch. The Basic and Standard monitoring. To see each monitoring and management jobs available, you can check the sub topics below:

- Jobs for Oracle
- Jobs for MS SQLServer
- Jobs for MySQL
- Jobs for MariaDB
- Jobs for PostgreSQL
- Jobs for Sybase

Basic Monitoring

Basic jobs consists of a series of steps (defined in xml files), and each step retreives data and/or performs some computation on the data.
The last step will also set a status and a text describing the current state of the job.

The steps can be a mixture of different languages, currently sql, fdl, javascript and ssh are supported.

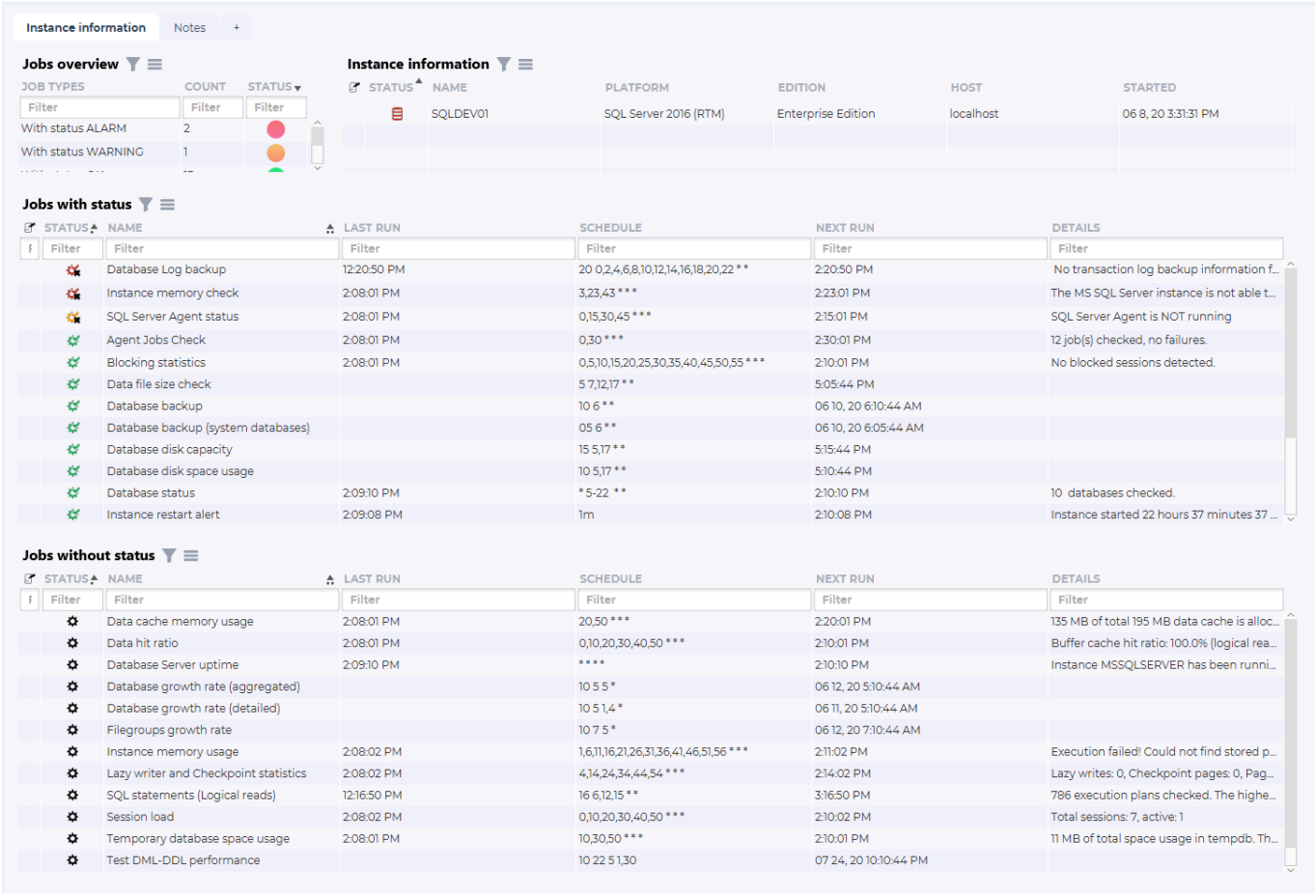
Each step has the results from the previous steps available as input.

The basic jobs do not require any framework or objects installed on the databases, all state is stored on the dbWatch Server.

Standard Monitoring

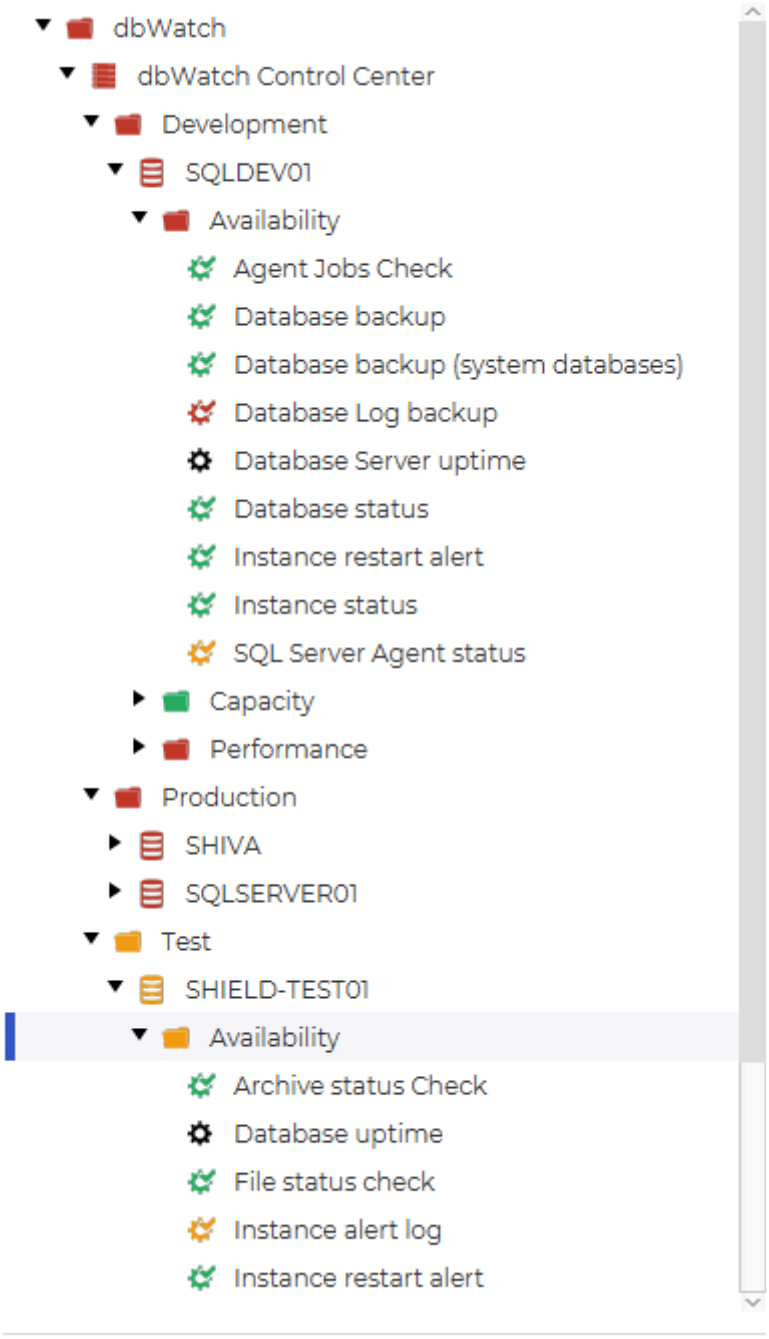
Each Standard Job consists of a set of objects (procedures, tables, etc) that are installed on the monitored database instance. The main procedure will be triggered by the dbWatch Server, This procedure will then perform a check or collect statistics regarding the instance and typically put data in a table that can be reported on later.

All Jobs provided by dbWatch are provided as open source. This is part of the dbWatch philosophy of mixed source. Customers are able to (and indeed we recommend) that they change the standard supplied jobs. By using our codes as the basis for their own customized ‘developments’. They can then customize their jobs to suit their specific needs. dbWatch Software Support is available to assist with customized developments.



In the main monitoring window you will see jobs along with various other information relating to their schedules. Such as when the last run was performed and so on.

The first column indicates whether it is a task (grey color) or an alert. With status (green = ok, yellow = warning, red = alarm).



On the left side of the monitoring window, you see tasks and alerts with their status order by package.

<< No-engine jobs / Jobs grouped by platform >>

Jobs grouped by platform

Go to the individual platforms to see the jobs for that database platform

- Jobs for Oracle
- Jobs for MS SQLServer
- Jobs for MySQL
- Jobs for MariaDB
- Jobs for PostgreSQL
- Jobs for Sybase

<< Control Center Jobs / Jobs for Oracle >>

Jobs for Oracle

Control Center Jobs	Description
Availability	
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Alert log check	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Archive status Check	Checks how many redolog files that are not archived.
Backup log Check 10g	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Backup log Check 8i	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Backup log Check 9i	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Data Guard Archive Status	Checks that the standby database is receiving archive files from master database.
Data Guard Archive Status Check	Checks that the standby database is receiving archive files from master database.
Database link check	Checks database links.
Instance alert log	Reads and checks errors in the X$DBGALERTEXT performance view contained in the XML decoded version of the XML version of Alert log file.
Database uptime	Collects database uptime statistics for all instances mounted on the database.
Dba Jobs check	Detecting failed scheduled jobs in old style dba_jobs.
DBMS uptime	Collects uptime statistics in database.
Export log Check	Checks for errors in a log file (i.e. export log).
File status check	Reacts on any changes in status of data files and/or temporary files.
Index status check	Detecting indexes in status UNUSABLE
Invalid objects check	Detecting invalid objects in the database.
Job scheduling check	Detecting failed scheduled jobs.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.

Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors in database listener log. Using embedded java to read.
Listener log check	This checks and reads errors in database listener log. Using embedded java to read.
Listener status check	Checks listener status to verify if up and running, requires java support in the database.
Max datafiles check	Checks the “soft limit” and the “hard limit” of maximum number of physical OS files, that can be mapped to an Oracle instance.
Database mount state	Checking if database is in correct mount status
Datafile status	Checking status for datafiles
Max datafiles	Checking if datafiles reaches max datafiles (db_files) parameter
RMAN backup status	Checking status of the last RMAN backup
Password expire	Checks for users whose password will soon expire.
PDB status	Alerts if PDB is not in correct state
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN archivelog backup status	Checks the status of RMAN archivelog backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN full backup from V$RMAN_BACKUP_JOB_DETAILS performance view for input_type ‘DB FULL’. The procedure can also check incremental (input_type ‘DB INCR’) level 0 backup (equivalent to full backup) by

	checking V$BACKUP_DATAFILE performance view (column INCREMENTAL_LEVEL=0).
RMAN incremental backup status	Checks the status of RMAN incremental backup (input_type 'DB INCR') from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN recovery area backup status	Checks the status of RMAN recovery area backup (input_type 'RECVR AREA') from V$RMAN_BACKUP_JOB_DETAILS performance view.
Tablespace in BEGIN BACKUP mode	Checks if any of the tablespaces are in 'BEGIN BACKUP' mode (caused by DDL 'alter tablespace
Test alert db	Test alert that alerts every 'X' minutes.
Test alert	Test alert that alerts every 'X' minutes.
TNSping check	Checks TNSping connectivity on listed targets. Task requires 'java' support. Supports Linux/Unix and Windows, will react with an alarm if host is unreachable.
Capacity	
ASM disk statistics	Checks ASM disks for status and statistics.
ASM diskgroup space	Checks ASM diskgroups space statistics.
Monitor SYS.AUD$ table size	Checks size of the AUD$ trail table.
Autoextensible data files	Checks all tablespaces for total disk space usage (autoextensible data files).
Disk space check	Disk space check requiring java support database. (Supports Linux, Solaris, AIX, HPUX and Windows).
Flash Recovery Area Usage	Checks space usage in the flash recovery area.
Flash Recovery Area Usage	Checks space usage in the flash recovery area.
Free extents	Checks for free extents in all tablespaces. A warning or an alarm is returned if segments storage value is higher than the largest free extent chunk.
Max processes	Checks the maximum number of processes.
ASM diskgroup space	Checking available space on the ASM diskgroups
Extent fragmentation	Checking for extent fragmentation in dba_free_space
Flash recovery space usage	Checking available space on the Flash recovery area
Max processes	Checking if database reaches max processes (processes) parameter
SYS.AUD size	Checking if SYS.AUD$ fills up with data
Tablespace free space	Checking tablespaces for free space

Segment size collector	Collects size and number totals on all segment types per segment-owner and tablespace_name.
Segment size collector (all segments — aggregate) (9i)	Collects size and number totals on all segment types per segment-owner and tablespace_name.
Segment size collector	Collects size and extent number info for larger segments in the database.
Segment size status for old style tablespaces	Checks segment storage definitions in tablespaces where extent management is not set to local.
Temporary tablespace free space check	Checks level of free space for temporary tablespaces.
Tablespace free space check	Checks level of free space for all tablespaces.
Cluster and replication	
Applied archive log gap status	Reacts on applied archived log gaps and errors.
Blocking detector for RAC	Checks if a session is waiting on a TX (transaction) lock. (RAC only)
Memory session load for RAC	Records the load memory of RAC active sessions over time.
MV Refresh Group	Checks refresh date of scheduled jobs in refresh group(s).
Dataguard apply time	Checking apply time in dataguard standby nodes
Dataguard archive sequence apply lag	Checking lag of applied archive logs on dataguard standby node
Dataguard standby archivelog gap	Checking for archivelog gap between threads on primary and standby instances in a replication/service group. Metadata “replicationgroup” must be set on all instances to the same as parameter replicationgroup. For upto 10 redo threads.
Dataguard startup time	Checking startup time on dataguard standby nodes
Dataguard transport time	Checking transport time in dataguard standby nodes
RAC instances status	Reacts if one or more RAC instances are not available.
Session load Rac	Records the number of active sessions over time (RAC only).
Snapshot Log(s) rows count	Checks row count on all snapshot logs for given schema.
Snapshot Log(s) size	Checks size of all snapshot logs.

Data Guard role switch	This job checks if Primary/Standby switch occurs.
Internal (optional)	
Restore uptodate status	Checking if database restore is uptodate
Maintenance	
Framework	Engine framework is used when upgrading earlier versions of dbWatch. Ensures the dbWatch Engine and dbWatch Server run compatible codes.
Maintenance (extra cost package)	
Analyze tables	Analyse tables automatically.
Delete SYS.AUD$ data	Deletes old rows from the SYS.AUD$ trail table.
Rebuild indexes	Rebuild indexes automatically.
Migration (optional for migration between dbWatch EM 12 and dbWatch Control Center)	
Statistics migration	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
Performance	
Blocking detector	Checks if a session is waiting on a TX (transaction) lock.
Buffer cache statistics	Gets data cache statistics for buffer cache in SGA.
CPU load	Checks CPU load using Oracle performance view v$osstat.
Database network statistics	Checks database network SQL*NET statistics.
Test DML performance	Runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on schema test tables.
File IO statistics	Collects I/O statistics for data files.
Instance memory statistics	Collects statistics on instance memory.
Latch statistics	Collects latch status statistics.
Long running queries	Checks any long running queries and then alerts.
Open cursors	Checks and collects the amount of open cursors per session.

Disk read statistics	Collects disk read statistics from v$system_event performance view for db file sequential and dbfile scattered read event.
Redo statistics	Gets redolog files statistics. Based on Oracle dictionary and performance views.
Redo statistics	Gets redolog files statistics. Based on Oracle dictionary and performance views.
Session load	Records the number of active and inactive sessions over time.
Session load	Records the number of active sessions over time.
SQL waits	Collects SQL wait statistics.
Table statistics check	Reacts on old (or missing) statistisc in DBA_TABLES dictionary table (column LAST_ANALYZED).
Top user memory usage	Check can be used to trace memory usage per server-process (session).
Undo statistics	Collects rollback segment statistics.
User memory statistics	Collects user memory statistics.
Wait statistics	Collects waits statistics encountered by threads that executed. This task is based on the v$system_event dynamic performance view.
Performance (extra cost package)	
SQL statistics	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views..
SQL statistics	Collects SQL statements statistics (for container database) from V$SQLAREA and V$SQL_PLAN dynamic performance views.
SQL statistics	Collects SQL statements statistics (for pluggable database) from V$SQLAREA and V$SQL_PLAN dynamic performance views.
SQL statistics	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views..

Alert log check

Name:	Alert log check
Platform:	Oracle
Category:	Availability
Description:	This check reads and looks for errors in the database alert log.
Long description:	This check reads and looks for errors in the database alert log. Uses Oracle internal routines to read files.
Version:	2.6
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above = '1' & _priv_read_v_parameter = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
alert log name	alert_ORACLE_SID.log	The name of the Oracle alert log.
alert log directory name	DBW_DUMP_DEST	The name of the directory object created with the DDL – "create directory".
alert log error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,).
alert log line HWM	1	Each time the Check read the alert log, it registers how many rows have been checked. The next time
alert log max lines	200000	If the alert log has more lines than the value of this parameter, then the Check terminates and
alert log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the execution time reach
alert log error text allowed	NULL	Specifies the error strings which are excluded.

<< Jobs for Oracle / Alert log check (10g) >>

Alert log check (10g)

Name:	Alert log check
Platform:	Oracle
Category:	Availability
Description:	This check reads and looks for errors in the database alert log.
Long description:	This check reads and looks for errors in the database alert log. Uses Oracle internal routines to read files. (Oracle 10g only).
Version:	2.8
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10%') & _priv_read_v_parameter = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
alert log name	alert_ORACLE_SID.log	The name of the Oracle alert log.
alert log directory name	DBW_DUMP_DEST	The name of the directory object created with the DDL – "create directory".
alert log error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,).
alert log line HWM	1	Each time the Check reads the alert log, it registers how many rows have been checked. The next time
alert log max lines	200000	If the alert log has more lines than the value of this parameter, then the Check terminates and
alert log max elapsed time	60	Defines the maximum time the Check can execute (values are in seconds). If the execution time reaches
alert log error text allowed	NULL	Specifies the error strings which are excluded.

loop time	180	The number of seconds the “Alert log check” will continue to try to open the alert log file if an exception occurs during opening of the file. This parameter was introduced as a workaround when the alert log file is locked by other process.
pause time	20	The number of seconds the check will wait before trying to reopen the alert log file when an exception occurred during opening of that file.
max exception time	40	The maximum number of minutes the check will try to reopen the alert log file. If the check is not able to open the file for this amount of time (in minutes), it will return a warning.
last exception	null	The timestamp (hh24:mi dd.mm.yyyy) when an exception occurred trying to open the alert log file. If null, the opening of the alert log file is successful.

<< Alert log check / Alert log check (9i) >>

Alert log check (9i)

Name:	Alert log check
Platform:	Oracle
Category:	Availability
Description:	This check reads and looks for errors in the database alert log.
Long description:	This check reads and looks for errors in the database alert log. Uses Oracle internal routines to read files. (Oracle 9i only).
Version:	2.6
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '9%') & _priv_read_v_parameter = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
alert log name	alert_ORACLE_SID.log	The name of the Oracle alert log.
alert log directory name	DBW_DUMP_DEST	The name of the directory object created with the DDL – "create directory".
alert log error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,).
alert log line HWM	1	Each time the Check read the alert log, it registers how many rows have been checked. The next time
alert log max lines	200000	If the alert log has more lines than the value of this parameter, then the Check terminates and
alert log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the execution time reach
alert log error text allowed	NULL	Specifies the error strings which are excluded.

loop time	180	The number of seconds the “Alert log check” will continue to try to open the alert log file if an exception occurs during opening of the file. This parameter was introduced as a workaround when the alert log file is locked by other process.
pause time	20	The number of seconds the check will wait before trying to reopen the alert log file when an exception occurred during opening of that file.
max exception time	40	The maximum number of minutes the check will try to reopen the alert log file. If the check is not able to open the file for this amount of time (in minutes), it will return a warning.
last exception	null	The timestamp (hh24:mi dd.mm.yyyy) when an exception occurred trying to open the alert log file. If null, the opening of the alert log file is successful.

<< Alert log check (10g) / Alert log check (8i) >>

Alert log check (8i)

Name:	Alert log check
Platform:	Oracle
Category:	Availability
Description:	This check reads and looks for errors in the database alert log.
Long description:	This check reads and looks for errors in the database alert log. Uses Oracle internal routines to read files. (Oracle 8i only).
Version:	2.5
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '8%') & _priv_read_v_parameter = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
alert log name	alert_ORACLE_SID.log	The name of the Oracle alert log.
alert log path	ORACLE_HOME\admin\ ORACLE_SID\bdump	The path name of the Oracle alert log.
alert log error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,).
alert log line HWM	1	Each time the Check read the alert log, it registers how many rows have been checked. The next time
alert log max lines	200000	If the alert log has more lines than the value of this parameter, then the Check terminates and
alert log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the execution time reach
alert log error text allowed	NULL	Specifies the error strings which are excluded.

<< Alert log check (9i) / Alert log check (java) >>

Alert log check (java)

Name:	Alert log check (java)
Platform:	Oracle
Category:	Availability
Description:	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Long description:	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Version:	2.7
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above = '1' & enable_java = '1'

Parameters		
Name	Default value	Description
alertlog name	AUTO	The name of the Oracle alert log. If set to AUTO, we will try to figure out the alert log file name ourself.
alertlog directory name	AUTO	The name of the directory we are looking for the alert log in. If set to AUTO, we will try to figure out where the alert log file is located.
alertlog error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,). Text is case and space sensitive, so " ORA-, Error" is looking for " ORA-" and " Error", different from "ORA-,Error" which looks for "ORA-" and "Error".
alertlog line HWM	0	Each time the Check read the alert log, it registers how many rows have been checked. The next time
alertlog read lines	4000	Number of lines to maximum read on each run.
alertlog error text allowed		Specifies the error strings which are excluded. Values must be separated with commas (,). Text is case and space sensitive, so " ORA-, Error" is looking for " ORA-" and " Error", different from "ORA-,Error" which looks for "ORA-" and "Error". This list is filtering away errors found by "alertlog error text", so if "ORA-" is searched for in "alertlog error text" parameter, and "ORA-600" is in "alertlog error text allowed" parameter, then "ORA-600" messages are not reported on.

[<< Alert log check (8i)] / [Alert log check (java for 10g) >>]

Alert log check (java for 10g)

Name:	Alert log check (java)
Platform:	Oracle
Category:	Availability
Description:	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Long description:	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Version:	2.7
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10.2%') & enable_java = '1'

Parameters		
Name	Default value	Description
alertlog name	AUTO	The name of the Oracle alert log. If set to AUTO, we will try to figure out the alert log file name ourself.
alertlog directory name	AUTO	The name of the directory we are looking for the alert log in. If set to AUTO, we will try to figure out where the alert log file is located.
alertlog error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,). Text is case and space sensitive, so " ORA-, Error" is looking for " ORA-" and " Error", different from "ORA-,Error" which looks for "ORA-" and "Error".
alertlog line HWM	0	Each time the Check read the alert log, it registers how many rows have been checked. The next time
alertlog read lines	4000	Number of lines to maximum read on each run.
alertlog error text allowed		Specifies the error strings which are excluded. Values must be separated with commas (,). Text is case and space sensitive, so " ORA-, Error" is looking for " ORA-" and " Error", different from "ORA-,Error" which looks for "ORA-" and "Error". This list is filtering away errors found by "alertlog error text", so if "ORA-" is searched for in "alertlog error text" parameter, and "ORA-600" is in "alertlog error text allowed" parameter, then "ORA-600" messages are not reported on.

<< Alert log check (java) / Archive status check >>

Archive status check

Name:	Archive status Check
Platform:	Oracle
Category:	Availability
Description:	Checks how many redolog files that are not archived.
Long description:	Checks how many redolog files that are not archived.
Version:	2.9
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_log = '1' & _priv_read_v_instance = '1' & _priv_read_v_parameter = '1'

Parameters		
Name	Default value	Description
max not archived	3	This is the maximum number of redolog groups that are allowed to have status "not archived" before a WARNING is triggered. If all redolog groups have status "not archived", an ALARM is triggered.

<< Alert log check (java for 10g) / Backup log check (old style 10g) >>

Backup log check (old style 10g)

Name:	Backup log Check 10g (‘old style’ backups)
Platform:	Oracle
Category:	Availability
Description:	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Long description:	Checks for errors in a log file (e.i. backup log). For backups not using RMAN, but using a script that uses ‘alter tablespace xx backup;’ and creates a log file. (Oracle 10g only).
Version:	2.5
Default schedule:	15 5 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & version like ’10%’ & _priv_read_dba_directories = ‘1’

Parameters		
Name	Default value	Description
backup log name	backup.log	The name of the backup log file.
backup log path	null	The path name of the backup log file which is used to create directory object.
backup log directory name	DBW_BACKUP_DIR	The name of the directory object created with the DDL – “create directory”.
backup log error text	ORA-	Defines which strings (errors) the Check must look for. Values for this parameter must be
backup log line HWM	0	Each time the Check read the backup log, it registers how many rows have been checked.
backup log max lines	200000	If the backup log has more lines than the value of this parameter, then the Check terminates
backup log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the execution ‘
backup log error text allowed	null	Specifies the error strings which are excluded.
backup log	successful	The Check loop throughout the log file, searching for a

confirm text		“confirm” text. If the Check do not find at least
backup log date format	dd-mm-yyyy hh24:mi	If a value is assigned to the parameter ~~backup log date position~~, you have to specify the date-format
backup log date position	backup finished	If the log-file has a date/time stamp in it, you can have the Check to check how
backup log nls lang	English	Defines handling of language-specific date-format used by the dbwatch parameter “backup log date format”.
backup log expire time	24	You can specify how old the log-file is allowed to be before the Check returns the

<< Archive status check / Backup log check (old style 9i) >>

Backup log check (old style 9i)

Name:	Backup log Check 9i (‘old style’ backups)
Platform:	Oracle
Category:	Availability
Description:	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Long description:	Checks for errors in a log file (e.i. backup log). For backups not using RMAN, but using a script that uses ‘alter tablespace xx backup;’ and creates a log file. (Oracle 9i only).
Version:	2.5
Default schedule:	15 5 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & version like ’9%’ & _priv_read_dba_directories = ‘1’

Parameters		
Name	Default value	Description
backup log name	backup.log	The name of the backup log file.
backup log path	null	The path name of the backup log file which is used to create directory object.
backup log directory name	DBW_BACKUP_DIR	The name of the directory object created with the DDL – “create directory”.
backup log error text	ORA-	Defines which strings (errors) the Check must look for. Values for this parameter must be
backup log line HWM	0	Each time the Check read the backup log, it registers how many rows have been checked.
backup log max lines	200000	If the backup log has more lines than the value of this parameter, then the Check terminates
backup log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the execution ‘
backup log error text allowed	null	Specifies the error strings which are excluded.
backup log	successful	The Check loop throughout the log file, searching for a

confirm text		“confirm” text. If the Check do not find at least
backup log date format	dd-mm-yyyy hh24:mi	If a value is assigned to the parameter ~~backup log date position~~, you have to specify the date-format
backup log date position	backup finished	If the log-file has a date/time stamp in it, you can have the Check to check how
backup log nls lang	English	Defines handling of language-specific date-format used by the dbwatch parameter “backup log date format”.
backup log expire time	24	You can specify how old the log-file is allowed to be before the Check returns the

[<< Backup log check (old style 10g)] / [Backup log check (old style 8i) >>]

Backup log check (old style 8i)

Name:	Backup log Check 8i(‘old style’ backups)
Platform:	Oracle
Category:	Availability
Description:	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Long description:	Checks for errors in a log file (e.i. backup log). For backups not using RMAN, but using a script that uses ‘alter tablespace xx backup;’ and creates a log file. (Oracle 8i only).
Version:	2.4
Default schedule:	55 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & version like ’8%’ & _priv_read_dba_directories = ‘1’

Parameters		
Name	Default value	Description
backup log name	backup log	The name of the backup file.
backup log path	backup path	The path name of the backup file.
backup log error text	warning	Defines which strings (errors) the Check must look for. Values must be separated with commas (,).
backup log confirm text	successful	The Check loop throughout the log file, searching for a “confirm” text. If the Check do not find at least
backup log date format	dd-mm-yyyy hh24:mi	If a value is assigned to the parameter ~~backup log date position~~, you have to specify the date-format
backup log date position	backup finished	If the log-file has a date/time stamp in it, you can have the Check to check how
backup log max lines	20000	If the log has more lines than the value of this parameter, then the Check terminates and
backup log line HWM	0	If the log-file is an append-type log (as alert and listener logs), the parameter must not be 0 (zero).
backup log nls lang	English	Defines handling of language-specific date-format used by the dbwatch parameter “backup log date format”.
backup log expire	24	You can specify how old the log-file is allowed to be before the Check

time		returns the
backup log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the
backup log error text allowed	NULL	Specifies the error strings which are excluded.

<< Backup log check (old style 9i) / Dataguard Archive Status >>

Dataguard Archive Status

Name:	Data Guard Archive Status
Platform:	Oracle
Category:	Availability
Description:	Checks that the standby database is receiving archive files from master database.
Long description:	Task checks that the standby database is receiving archive files from the master database.
Version:	3.0
Default schedule:	10,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_instance = '1' & _priv_read_v_database = '1' & _priv_read_v_archived_log = '1'

Parameters		
Name	Default value	Description
warning threshold	5	This parameter is used to control the gap between ARCHIVED_SEQ# values for Master and Standby databases (V$ARCHIVE_DEST_STATUS dictionary performance view).
alarm threshold	20	This parameter is used to control the gap between ARCHIVED_SEQ# values for Master and Standby databases (V$ARCHIVE_DEST_STATUS dictionary performance view).
local dest ID	1	This parameter refers to DEST_ID value (V$ARCHIVE_DEST_STATUS dictionary performance view) for local archive destination.
remote dest ID	2	This parameter refers to DEST_ID value (V$ARCHIVE_DEST_STATUS dictionary performance view) for remote archive destination.

<< Backup log check (old style 8i) / Data Guard Archive Status Check (10g & 9i) >>

Data Guard Archive Status Check (10g & 9i)

Name:	Data Guard Archive Status Check
Platform:	Oracle
Category:	Availability
Description:	Checks that the standby database is receiving archive files from master database.
Long description:	Task checks that the standby database is receiving archive files from the master database.
Version:	2.3
Default schedule:	10,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[version like '9%' | version like '10%'

Parameters		
Name	Default value	Description
warning threshold	5	This parameter is used to control the gap between ARCHIVED_SEQ# values for Master and Standby databases (V$ARCHIVE_DEST_STATUS dictionary performance view).
alarm threshold	20	This parameter is used to control the gap between ARCHIVED_SEQ# values for Master and Standby databases (V$ARCHIVE_DEST_STATUS dictionary performance view).
local dest ID	1	This parameter refers to DEST_ID value (V$ARCHIVE_DEST_STATUS dictionary performance view) for local archive destination.
remote dest ID	2	This parameter refers to DEST_ID value (V$ARCHIVE_DEST_STATUS dictionary performance view) for remote archive destination.

<< Dataguard Archive Status / Database link check >>

Database link check

Name:	Database link check
Platform:	Oracle
Category:	Availability
Description:	Checks database links.
Long description:	In order to check private database links you have to create a following view ‘create view dbw_link_check as select username from user_users@DB_LINK’ in the database link owner schema, and then grant select privilege to dbwatch user on that view (‘grant select on dbw_link_check to dbwatch’).
Version:	2.5
Default schedule:	15,45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & _priv_read_dba_db_links = ‘1’

Parameters		
Name	Default value	Description
ignore database links		Database links names excluded from being checked (comma separated).
return status when database link is down	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when database link is down.
check private db links	NO	

<< Data Guard Archive Status Check (10g & 9i) / Instance alert log >>

Instance alert log

Name:	Instance alert log
Platform:	Oracle
Category:	Availability
Description:	Reads and checks errors in the X$DBGALERTEXT performance view contained in the XML decoded version of the XML version of Alert log file.
Long description:	Task reads and checks errors in the X$DBGALERTEXT performance view which contains the XML decoded version of the XML version of the Alert log file.
Version:	5.2
Default schedule:	0,5,10,15,20,25,30,35,40,45,50,55 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1' & has_x_dbgalertext='1'

Parameters		
Name	Default value	Description
error text	ORA-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,).
HWM	1	Each time the Check read the X$DBGALERTEXT view, it registers how many rows have been checked. The next time the Check executes, it browses pass the rows already checked. The value of how many rows of the X$DBGALERTEXT view have already been checked are registered by this parameter.
error text allowed	ORA-0	Specifies the error strings which are excluded.
seperator	,	Specifies separator sign between text strings or lines
history threshold	1000	The maximum numbers of error messages history this table will keep (in the db_alert_err_tab_histr table).
max_lines	500000	The maximum numbers of lines we want to have in X$DBGALERTEXT view before we ask to delete files on disk

<< Database link check / Database uptime >>

Database uptime

Name:	Database uptime
Platform:	Oracle
Category:	Availability
Description:	Collects database uptime statistics for all instances mounted on the database.
Long description:	Collects database uptime statistics for all instances mounted on the database.
Version:	3.2
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above = '1' & _priv_read_sys_gv_instance = '1'

Parameters		
Name	Default value	Description

<< Instance alert log / DBA jobs check >>

DBA jobs check

Name:	Dba Jobs check
Platform:	Oracle
Category:	Availability
Description:	Detecting failed scheduled jobs in old style dba_jobs.
Long description:	The BROKEN column in the Oracle dictionary view DBA_JOBS is used to detect jobs that have failed.
Version:	2.3
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_dba_jobs = '1'

Parameters		
Name	Default value	Description
ignore job id		Job ID from job column in dba_jobs excluded from being checked (comma separated).
return status on failed job	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when a job have status FAILED (and are not in "ignore job names" list).
last run	01.03.2010 00:00	The last date the dbWatch Job scheduling check was run. Only jobs run after this date will be checked (column actual_start_date in the dba_scheduler_job_run_details view). Value format DD.MM.YYYY HH24:MI

<< Database uptime / DBMS uptime >>

DBMS uptime

Name:	DBMS uptime
Platform:	Oracle
Category:	Availability
Description:	Collects uptime statistics in database.
Long description:	Task collects uptime statistics in database.
Version:	3.0
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_session = '1'

Parameters		
Name	Default value	Description
dbw username	dbwatch	Database user name used by the dbWatch Server when connecting to the dbWatch Engine.
dbw user id	0	Database user id used by the dbWatch Server when connected to the dbWatch Engine.
end_availability_timestamp	00:00	Availability timestamp (hh24:mi),used to define allowed DBMS shutdown period.
end_availability_duration	0	Availability duration in minutes.

<< DBA jobs check / Export log Check ('old style' backups) >>

Export log Check (‘old style’ backups)

Name:	Export log Check (‘old style’ backups)
Platform:	Oracle
Category:	Availability
Description:	Checks for errors in a log file (i.e. export log).
Long description:	Task checks for errors in the export log file (‘old style’ backups)
Version:	3.7
Default schedule:	15 5 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & _priv_read_dba_directories = ‘1’ & can_get_grant_create_any_directory=‘1’

Parameters		
Name	Default value	Description
export log name	export.log	The name of the export log file. However, if your export script makes a new name for each export log file, you can use the “$” signs to append the date-format to the filename. For example, if the filename is exp_weekly_$(date “+%m%d%y”).log which would become, for 25Th of May 2010, exp_weekly_052510.log, you set the parameters to the following value: exp_weekly_$MMDDYY$.log
export log path	null	The path name of the export log file which is used to create directory object.
export log directory name	DBW_EXPORT_DIR	The name of the directory object created with the DDL – “create directory”. Directory object value should be the path to the export.log file.
export log error text	ORA-	Defines which strings (errors) the Check must look for. Values for this parameter must be separated with commas (,).
export log line HWM	0	Each time the Check read the export log, it registers how many rows have been checked. The next time the Check executes, it browses pass the rows already checked. The value of how many rows of the export log have already been checked are registered by this parameter.
export log max lines	200000	If the export log has more lines than the value of this parameter, then the Check terminates and returns WARNING status.

export log max elapsed time	60	Define the maximum running time the Check can execute (values are in seconds). If the execution time reach the ~~export log max elapsed time~~ value before reaching the end of file, the Check terminates and returns WARNING status.
export log error text allowed	null	Specifies the error strings which are excluded.
export log confirm text	terminated successfully without warnings	The Check loop throughout the log file, searching for a confirm text. If the Check do not find at least one of the confirm strings, it returns WARNING status. You can specify more than one confirm string for the Check to look for, the strings should be separated by comma (,).
export log date format	dd-mm-yyyy hh24:mi	If a value is assigned to the parameter ~~export log date position~~, you have to specify the date-format by using the Oracle date format (ex. Fri 22:30:04 10-Jan-03 , dd hh24:mi:ss dd-mon-yy).
export log date position	export finished	If the log-file has a date/time stamp in it, you can have the Check to check how recently the log-file has been created (so you do not read a “good” but old log). To be able to find the date/time stamp in the log-file, you have to specify a unique string which must appear on the same line as the date/time stamp or on the line above. The unique string value must be specified by using this parameter. If no value is assigned to that parameter, the Check do not check the date of the log-file. Otherwise you also have to specify the date-format by using the dbWatch parameter “export log date format”.
export log nls lang	English	Defines handling of language-specific date-format used by the dbwatch parameter “export log date format”.
export log expire time	24	You can specify how old the log-file is allowed to be before the Check returns the WARNING status. In other words, if the log-file is created at 22:00 and the Check which checks the log-file is executed at 06:00 (next morning), the value of the parameter must at least be 8 hours. Otherwise the Check will return WARNING status because the log-file is to “old”.

<< DBMS uptime / File status check >>

File status check

Name:	File status check
Platform:	Oracle
Category:	Availability
Description:	Reacts on any changes in status of data files and/or temporary files.
Long description:	Task reacts on any changes on the status of data files and/or temporary files.
Version:	3.2
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_datafile = '1' & _priv_read_v_tempfile = '1'

Parameters		
Name	Default value	Description
ignore files	0	file ID(s) excluded from being checked (comma separated). To avoid file ID duplicates, the file ID for temporary files must be specified with a minus sign in front of the file ID.
return status when file is OFFLINE	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when file(s) have status "OFFLINE" (and are not in "ignore files" list).
return status when file needs RECOVERY	2	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when file(s) have status "RECOVER" (and are not in "ignore files" list).
quarantine date	2023.01.01 12:00	
quarantine files	0	File IDs list (file#) to be ignored if having status OFFLINE or RECOVERY and having checkpoint time within "quarantine time" value.
quarantine date format	yyyy.mm.dd hh24:mi	Date format used by "quarantine date" parameter.

<< Export log Check ('old style' backups) / Index status check >>

Index status check

Name:	Index status check
Platform:	Oracle
Category:	Availability
Description:	Detecting indexes in status UNUSABLE
Long description:	Detecting indexes in status UNUSABLE
Version:	1.4
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_all_indexes = '1' & maj_version > '8'

Parameters		
Name	Default value	Description
ignore index		Indexes we dont give a warning for. Values for this parameter must be separated with commas (,).

<< File status check / Invalid objects check >>

Invalid objects check

Name:	Invalid objects check
Platform:	Oracle
Category:	Availability
Description:	Detecting invalid objects in the database.
Long description:	This check detects invalid objects in the database and gives a warning when new objects.
Version:	1.7
Default schedule:	10,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above = '1' & _priv_read_dba_objects = '1'

Parameters		
Name	Default value	Description
ignore object name		object name excluded from being checked (comma separated). The format is OWNER.OBJECT example SYS.LOCKING
ignore owner		owner excluded from being checked (comma separated).
return status on invalid objects	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when an invalid object appears (and are not in "ignore object name" list).

<< Index status check / Job scheduling check >>

Job scheduling check

Name:	Job scheduling check
Platform:	Oracle
Category:	Availability
Description:	Detecting failed scheduled jobs. The STATUS column in the Oracle dictionary view DBA_SCHEDULER_JOB_RUN_DETAILS is used to detect scheduled jobs that have failed.
Long description:	The STATUS column in the Oracle dictionary view DBA_SCHEDULER_JOB_RUN_DETAILS is used to detect scheduled jobs that have failed.
Version:	3.02
Default schedule:	5,25,45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above = '1'

Parameters		
Name	Default value	Description
ignore job names	AUTO_SPACE_ADVISOR_JOB,FGR$AUTOPURGE_JOB	Job names excluded from being checked (comma separated). You can use % (percent sign) to represent wild card characters.
return status on failed job	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when a job have status FAILED (and are not in "ignore job names" list).
last run	01.03.2010 00:00	The last date the dbWatch Job scheduling check was run. Only jobs run after this date will be checked (column actual_start_date in the dba_scheduler_job_run_details view). Value format DD.MM.YYYY HH24:MI

<< Invalid objects check / Listener log check >>

Listener log check

Name:	Listener log check
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Long description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Version:	1.8
Default schedule:	7,27,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & twelve_and_above = '1' & _priv_read_dba_directories = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
listener log name	listener.log	The name of the Listener log file.
listener log path	null	The path name of the Listener log file which is used to create directory object.
listener log directory name	DBW_LISTENER_DIR	The name of the directory object created with the DDL – "create directory".
listener log error text	TNS-	Defines which strings (errors) the Check must look for. Values for this parameter must be
listener log line HWM	1	Each time the Check read the listener log, it registers how many rows have been checked.
listener log max lines	200000	If the listener log has more lines than the value of this parameter, then the Check terminates
listener log max elapsed time	60	Define the maximum elaps time the Check can execute (values are in seconds). If the execution '
listener log error text allowed	TNS-12500	Specifies the error strings which are excluded.

<< Job scheduling check / Listener log check (10g) >>

Listener log check (10g)

Name:	Listener log check
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Long description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Version:	1.8
Default schedule:	7,27,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10%') & _priv_read_dba_directories = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
listener log name	listener.log	The name of the Listener log file.
listener log path	null	The path name of the Listener log file which is used to create directory object.
listener log directory name	DBW_LISTENER_DIR	The name of the directory object created with the DDL – "create directory".
listener log error text	TNS-	Defines which strings (errors) the Check must look for. Values for this parameter must be
listener log line HWM	1	Each time the Check read the listener log, it registers how many rows have been checked.
listener log max lines	200000	If the listener log has more lines than the value of this parameter, then the Check terminates
listener log max elapsed time	60	Define the maximum elaps time the Check can execute (values are in seconds). If the execution '
listener log error text allowed	TNS-12500	Specifies the error strings which are excluded.

<< Listener log check / Listener log check (11g) >>

Listener log check (11g)

Name:	Listener log check
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Long description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Version:	2.8
Default schedule:	7,27,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '11%') & _priv_read_dba_directories = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
listener log name	listener.log	The name of the Listener log file.
listener log path	null	The path name of the Listener log file which is used to create directory object.
listener log directory name	DBW_LISTENER_DIR	The name of the directory object created with the DDL – "create directory".
listener log error text	TNS-	Defines which strings (errors) the Check must look for. Values for this parameter must be
listener log line HWM	1	Each time the Check read the listener log, it registers how many rows have been checked.
listener log max lines	200000	If the listener log has more lines than the value of this parameter, then the Check terminates
listener log max elapsed time	60	Define the maximum elaps time the Check can execute (values are in seconds). If the execution '
listener log error text allowed	TNS-12500	Specifies the error strings which are excluded.

<< Listener log check (10g) / Listener log check (9i) >>

Listener log check (9i)

Name:	Listener log check
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Long description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Version:	1.9
Default schedule:	7,27,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '9%') & _priv_read_dba_directories = '1' & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
listener log name	listener.log	The name of the Listener log file.
listener log path	null	The path name of the Listener log file which is used to create directory object.
listener log directory name	DBW_LISTENER_DIR	The name of the directory object created with the DDL – "create directory".
listener log error text	TNS-	Defines which strings (errors) the Check must look for. Values for this parameter must be
listener log line HWM	1	Each time the Check read the listener log, it registers how many rows have been checked.
listener log max lines	200000	If the listener log has more lines than the value of this parameter, then the Check terminates
listener log max elapsed time	60	Define the maximum elaps time the Check can execute (values are in seconds). If the execution '
listener log error text allowed	TNS-12500	Specifies the error strings which are excluded.

<< Listener log check (11g) / Listener log check (8i) >>

Listener log check (8i)

Name:	Listener log check
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors from the database listener log. Using Oracle native code to read.
Long description:	This checks and reads errors from the database listener log. Using Oracle native code to read. (Oracle 8i only)
Version:	1.5
Default schedule:	7,27,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '8%') & can_get_grant_create_any_directory='1'

Parameters		
Name	Default value	Description
listener log name	listener.log	The name of the Listener log file.
listener log path	ORACLE_HOME\network\log	The path name of the Listener log file.
listener log error text	TNS-	Defines which strings (errors) the Check must look for. Values for this parameter must be
listener log line HWM	1	Each time the Check read the listener log, it registers how many rows have been checked.
listener log max lines	1	If the listener log has more lines than the value of this parameter, then the Check terminates
listener log max elapsed time	60	Define the maximum elaps time the Check can execute (values are in seconds). If the execution '
listener log error text allowed	TNS-12500	Specifies the error strings which are excluded.

<< Listener log check (9i) / Listener log check (java) >>

Listener log check (java)

Name:	Listener log check (java)
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors in database listener log. Using embedded java to read.
Long description:	Task checks and reads the database listener log for errors, using embedded java to read.
Version:	2.7
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & enable_java = '1' & eleven_and_above = '1'

Parameters		
Name	Default value	Description
listenerlog name	AUTO	The name of the Oracle listener log. If set to AUTO, we will try to figure out the listener log file name ourself.
listenerlog directory name	AUTO	The name of the directory we are looking for the listener log in. If set to AUTO, we will try to figure out where the listener log file is located.
listenerlog error text	TNS-	Defines which strings (errors) the Check must look for. Values must be separated with commas (,). Text is case and space sensitive, so " TNS-, Error" is looking for " TNS-" and " Error", different from "TNS-,Error" which looks for "TNS-" and "Error".
listenerlog line HWM	0	Each time the Check read the listener log, it registers how many rows have been checked. The next time
listenerlog read lines	4000	Number of lines to maximum read on each run.
listenerlog error text allowed	TNS-12500	Specifies the error strings which are excluded. Values must be separated with commas (,). Text is case and space sensitive, so " TNS-, Error" is looking for " TNS-" and " Error", different from "TNS-,Error" which looks for "TNS-" and "Error". This list is filtering away errors found by "listenerlog error text", so if "TNS-" is searched for in "listenerlog error text" parameter, and "TNS-1521" is in "listenerlog error text allowed" parameter, then "TNS-1521" messages are not reported on.

[<< Listener log check (8i)](#) / [Listener log check (java 10g) >>](#)

Listener log check (java 10g)

Name:	Listener log check (java)
Platform:	Oracle
Category:	Availability
Description:	This checks and reads errors in database listener log. Using embedded java to read.
Long description:	Task checks and reads the database listener log for errors, using embedded java to read.
Version:	2.7
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & enable_java = '1' & (version like '10.2%')]]]

Listener status check

Name:	Listener status check
Platform:	Oracle
Category:	Availability
Description:	Checks listener status to verify if up and running, requires java support in the database.
Long description:	Task checks listener status to verify that listener is up and running. This task requires java support in the database.
Version:	3.2
Default schedule:	0,5,10,15,20,25,30,35,40,45,50,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above = '1' & enable_java = '1'

Parameters		
Name	Default value	Description
Delete statistics older than (hours)	336	
Listener status string	AUTO	Give the alert the lsnrctl status exec command if it can not find it by itself. Possible values are "AUTO" or an exec string such as "/u01/app/oracle/10.0/db_1/bin/lsnrctl status".
Status match text	READY	This is the value to look for in the lsnrctl status line for this instance. Possible values includes, but are not limited to READY and UNKNOWN.
Named listener name		Put a value here, if you want to check a named listener

<< Listener log check (java 10g) / Max datafiles check >>

Max datafiles check

Name:	Max datafiles check
Platform:	Oracle
Category:	Availability
Description:	Checks the ‘soft limit’ and the ‘hard limit’ of the maximum number of physical OS files, that can be mapped to an Oracle instance.
Long description:	From Oracle 9i, an attempt to add a file whose number is greater than MAXDATAFILES (‘hard limit’), but less than or equal to DB_FILES (‘soft limit’), causes the Oracle control file to expand automatically. Within the ora parameter ‘db_files’ applies to the ‘soft limit’. The MAXDATAFILES value stored in the Oracle control files applies to the ‘hard limit’.
Version:	2.7
Default schedule:	15 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & _priv_read_v_controlfile_record_section = ‘1’ & _priv_read_v_parameter = ‘1’

Parameters		
Name	Default value	Description
warning threshold “soft limit”	5	Warning threshold – how many data files can be allocated to the Oracle instance before the init ora parameter “db_files” value is reached (“db_file” is a static parameter).
warning threshold “hard limit”	5	Warning threshed – how many data files can be allocated to the Oracle instance before the MAXDATAFILES limit, stored in the Oracle control files, is reached. Your will have to recreate control files to change this limit.
alarm threshold “soft limit”	2	Alarm threshed – how many data files can be allocated to the Oracle instance before the init ora parameter “db_files” value is reached (“db_file” is a static parameter).
alarm threshold “hard limit”	2	Alarm threshed – how many data files can be allocated to the Oracle instance before the MAXDATAFILES limit, stored in the Oracle control files, is reached. Your will have to recreate control files to change this limit.

<< Listener status check / Database mount state >>

Database mount state

Name:	Database mount state
Platform:	Oracle
Category:	Availability
Description:	Checks the ‘soft limit’ and the ‘hard limit’ of the maximum number of physical OS files, that can be mapped to an Oracle instance.
Long description:	
Version:	1.0
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]

Parameters		
Name	Default value	Description
correct_mount_state	READ WRITE	Correct mount status for OK signal

<< Max datafiles check / Datafile status >>

Datafile status

Name:	Datafile status
Platform:	Oracle
Category:	Availability
Description:	Checks the ‘soft limit’ and the ‘hard limit’ of the maximum number of physical OS files, that can be mapped to an Oracle instance.
Long description:	
Version:	1.0
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]

Parameters		
Name	Default value	Description
return_status_offline	1	Alarm or warning status if datafile is offline. 0=OK, 1=Warning, 2=Alarm
return_status_recovery	2	Alarm or warning status if datafile needs recovery. 0=OK, 1=Warning, 2=Alarm

<< Database mount state / Max datafiles >>

Max datafiles

Name:	Max datafiles
Platform:	Oracle
Category:	Availability
Description:	Checks the ‘soft limit’ and the ‘hard limit’ of the maximum number of physical OS files, that can be mapped to an Oracle instance.
Long description:	
Version:	1.0
Default schedule:	45m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]

Parameters		
Name	Default value	Description
warning_threshold	20	Number of database files left until max datafiles (db_files) is reached in order to generate warning
alarm_threshold	5	Number of database files left until max datafiles (db_files) is reached in order to generate alarm

<< Datafile status / RMAN backup status (noschema) >>

RMAN backup status (noschema)

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checking status of the last RMAN backup
Long description:	
Version:	1.0
Default schedule:	4h
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Password expire

Name:	Password expire
Platform:	Oracle
Category:	Availability
Description:	Checks for users whose password will soon expire.
Long description:	Task checks for users whose password will soon expire.
Version:	3.3
Default schedule:	0 10 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_dba_users = '1'

Parameters		
Name	Default value	Description
Days to warning	30	Send a warning if less than this about of days to expire
Days to alarm	7	Send an alarm if less than this about of days to expire
Disable warnings	NO	Enables/disables warnings. YES or NO values
Disable alarms	NO	Enables/disables alarms. YES or NO values
Keep history for	180	Days to keep history
exclude users		Name(s) of user(s) which should be excluded (separated by comma).

<< RMAN backup status (noschema) / PDB status >>

PDB status

Name:	PDB status
Platform:	Oracle
Category:	Availability
Description:	Alerts if PDB is not in correct state
Long description:	Alerts if PDB is not in correct state
Version:	0.1
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & is_containerdb='YES'

Parameters		
Name	Default value	Description
Correct state	READ WRITE	Job will send alerts if pluggable databases are not in this state
Ignored pdbs	PDB$SEED	Comma separated list of pdbs that are ignored
Type of alarm	2	What alarm signal to send if PDB's are in wrong state. 2=Alarm, 1=Warning, 0=OK
History max time	365	How long should we keep data for, in days

<< Password expire / RMAN backup status (10.1) >>

RMAN backup status (10.1)

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	2.4
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10.1%')

Parameters		
Name	Default value	Description
backup expire time	24	Backup expire time.
return status when backup RUNNING	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is RUNNING.
return status when backup FAILED	2	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is FAILED.

<< PDB status / RMAN backup status (10.2) >>

RMAN backup status (10.2)

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	3.2
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10.2%')

Parameters		
Name	Default value	Description
backup expire time	24	Backup expire time.
return status when backup RUNNING	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is RUNNING.
return status when backup FAILED	2	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is FAILED.

<< RMAN backup status (10.1) / RMAN backup status (11) >>

RMAN backup status (11)

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	3.2
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '11%')

Parameters		
Name	Default value	Description
backup expire time	24	Backup expire time.
return status when backup RUNNING	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is RUNNING.
return status when backup FAILED	2	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is FAILED.

<< RMAN backup status (10.2) / RMAN backup status (8) >>

RMAN backup status (8)

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checks status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Task checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view. (For 8.x)
Version:	1.4
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '8%')

Parameters		
Name	Default value	Description
backup expire time	24	Backup expire time (in hours).

<< RMAN backup status (11) / RMAN backup status (9) >>

RMAN backup status (9)

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	2.4
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '9%')

Parameters		
Name	Default value	Description
backup expire time	24	Backup expire time (in hours).

<< RMAN backup status (8) / RMAN archivelog backup status >>

RMAN archivelog backup status

Name:	RMAN archivelog backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN archivelog backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN archivelog backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	1.2
Default schedule:	15,45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _cdb_v_rman_backup_job_details = '1' & is_pluggdb = '0'

Parameters		
Name	Default value	Description
backup expire time	24	Backup expire time (in hours). If no valid backup exists within this time threshold a warning is returned.
return status when backup RUNNING	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is RUNNING.
return status when backup FAILED	2	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is FAILED.
return status when OLD backup	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup is older than "backup expire time" parameter value.
return status when backup NOT IMPLEMENTED	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup is not implemented.

<< RMAN backup status (9) / RMAN backup status >>

RMAN backup status

Name:	RMAN backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN full backup from V$RMAN_BACKUP_JOB_DETAILS performance view for input_type 'DB FULL'. The procedure can also check incremental (input_type 'DB INCR') level 0 backup (equivalent to full backup) by checking V$BACKUP_DATAFILE performance view (column INCREMENTAL_LEVEL=0).
Long description:	Checks the status of RMAN full backup from V$RMAN_BACKUP_JOB_DETAILS performance view for input_type 'DB FULL'. The procedure can also check incremental (input_type 'DB INCR') level 0 backup (equivalent to full backup) by checking V$BACKUP_DATAFILE performance view (column INCREMENTAL_LEVEL=0).
Version:	2.64
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _cdb_v_rman_backup_job_details = '1' & is_pluggdb = '0'

Parameters		
Name	Default value	Description
backup expire time	168	Backup expire time (in hours). If no valid backup exists within this time threshold a warning is returned.
return status when backup RUNNING	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is RUNNING.
return status when backup FAILED	2	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup status value is FAILED.
check incremental level 0	YES	If set to YES, the procedure checks for incremental level 0 backup if full backup (input_type = "DB FULL") does not exists, or if the last full backup is running or failed.
container ID	1	The ID of the container to which the data pertains.
return status	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN

when OLD backup		backup is older than “backup expire time” parameter value.
return status when backup NOT IMPLEMENTED	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when RMAN backup is not implemented.
history threshold	90	The number of days the statistics are held in the dbwatch history table.
min output/input bytes ratio	95	The minimum output/input bytes ratio for incremental backup (input_type = “DB INCR”) which indicates if this is a level 0 or level 1 incremental backup. If no RMAN backup information is found (input_type “DB FULL” and “DB INCR” level 0) in V$RMAN_BACKUP_JOB_DETAILS, V$BACKUP_DATAFILE and dbWatch history table, the procedure check output/input bytes ratio in V$RMAN_BACKUP_JOB_DETAILS.

<< RMAN archivelog backup status / RMAN incremental backup status >>

RMAN incremental backup status

Name:	RMAN incremental backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN incremental backup (input_type ‘DB INCR’) from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN incremental backup (input_type ‘DB INCR’) from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	2.0
Default schedule:	20 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & _cdb_v_rman_backup_job_details = ‘1’ & is_pluggdb = ‘0’

Parameters		
Name	Default value	Description

Test alert db

Name:	Test alert db
Platform:	Oracle
Category:	Availability
Description:	Test alert that alerts every ‘X’ minutes.
Long description:	Test alert that alerts every ‘X’ minutes. Usefull for testing extensions and not intended for use in production environments. Database centric, so runs on a single database or on one of the nodes in a RAC cluster.
Version:	2.9
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’]]]

Test alert

Name:	Test alert
Platform:	Oracle
Category:	Availability
Description:	Test alert that alerts every ‘X’ minutes.
Long description:	Test alert that alerts every ‘X’ minutes. Usefull for testing extensions and not intended for use in production environments. Instance centric, so intended for single instance or each instance in a RAC cluster.
Version:	3.0
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’]]]

TNSping check

Name:	TNSping check
Platform:	Oracle
Category:	Availability
Description:	Checks TNSping connectivity on listed targets. Task requires ‘java’ support. Supports Linux/ Unix and Windows, will react with an alarm if host is unreachable.
Long description:	Task checks TNSping connectivity on listed targets. Task requires ‘java’ support. Supports Linux/Unix and Windows and will react with an alarm if host is unreachable.
Version:	2.7
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & eleven_and_above = ‘1’ & enable_java = ‘1’]]]

ASM disk statistics

Name:	ASM disk statistics
Platform:	Oracle
Category:	Capacity
Description:	Checks ASM disks for status and statistics.
Long description:	Checks ASM disks for status and statistics.
Version:	3.2
Default schedule:	2,12,22,32,43,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_asm_disk_stat = '1']]]

ASM diskgroup space

Name:	ASM diskgroup space
Platform:	Oracle
Category:	Capacity
Description:	Checks ASM diskgroups space statistics.
Long description:	Task checks ASM diskgroups for space statistics.
Version:	3.6
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_asm_diskgroup = '1']]]

Aud$ table size check

Name:	AUD$ table size check
Platform:	Oracle
Category:	Capacity
Description:	Checks size of the AUD$ trail table.
Long description:	Checks size of the AUD$ trail table.
Version:	1.5
Default schedule:	15 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_dba_segments = '1' & _priv_read_sys_aud = '1' & _priv_read_v_parameter = '1']]]

Autoextensible data files

Name:	Autoextensible data files
Platform:	Oracle
Category:	Capacity
Description:	Checks all tablespaces for total disk space usage (autoextensible data files).
Long description:	Task checks all tablespaces for total disk space usage (autoextensible data files).
Version:	3.0
Default schedule:	55 6,11,18 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_dba_data_files = '1']]]

Disk space check

Name:	Disk space check
Platform:	Oracle
Category:	Capacity
Description:	Disk space check requiring java support database. (Supports Linux, Solaris, AIX, HPUX and Windows).
Long description:	Task checks disk space. This requires java support in the database and supports Linux, Solaris, AIX, HPUX and Windows. This check will react with a warning or an alarm only when BOTH warning pct AND warning mb is reached.
Version:	5.92
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_enable_java = '1']]]

Flash Recovery Area Usage

Name:	Flash Recovery Area Usage
Platform:	Oracle
Category:	Capacity
Description:	Checks space usage in the flash recovery area.
Long description:	Checks space usage in the flash recovery area.
Version:	2.8
Default schedule:	10 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above = '1' & _priv_read_v_flash_recovery_area_usage = '1'

Parameters		
Name	Default value	Description
warning threshold	80	Warning threshold for flash recovery area usage (percentage).
alarm threshold	95	Alarm threshold for flash recovery area usage (percentage).
warning threshold (GB)	5	Minimum of free space (in GB) for the flash recovery area. If the amount of free space falls below this parameter value, and at the same time the amount of free space in percent falls below the "warning threshold" parameter value, the alert will return a WARNING.
alarm threshold (GB)	2	Minimum of free space (in GB) for the flash recovery area. If the amount of free space falls below this parameter value, and at the same time the amount of free space in percent falls below the "alarm threshold" parameter value, the alert will return an ALARM.
history threshold	30	The number of days the statistics will be kept in the history table.
reclaimable space as free space	NO	If sett to "YES" the reclaimable space usage of Flash Recovery Area will be calculated as free space.

<< Disk space check / Flash Recovery Area Usage (10g) >>

Flash Recovery Area Usage (10g)

Name:	Flash Recovery Area Usage
Platform:	Oracle
Category:	Capacity
Description:	Checks space usage in the flash recovery area.
Long description:	Task checks space usage in the flash recovery area.
Version:	2.5
Default schedule:	10 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[version like '10.2%' & hasengine='YES' & _priv_read_v_flash_recovery_area_usage = '1']]]

Free extents

Name:	Free extents
Platform:	Oracle
Category:	Capacity
Description:	Checks for free extents in all tablespaces. A warning or an alarm is returned if segments storage value is higher than the largest free extent chunk.
Long description:	Task checks ‘free extents’ in all tablespaces. If there are any segments with storage parameter “next extent” values higher than the largest free extent chunk in the tablespace, a warning or an alarm is returned by the check.
Version:	1.9
Default schedule:	50 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’]]]

Max processes

Name:	Max processes
Platform:	Oracle
Category:	Capacity
Description:	Checks the maximum number of processes.
Long description:	Task checks the number of processes and the instance parameter value “processes”.
Version:	1.5
Default schedule:	0,4,8,12,16,20,24,28,32,36,40,44,48,52,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’]]]

ASM diskgroup space (noschema)

Name:	ASM diskgroup space
Platform:	Oracle
Category:	Capacity
Description:	Checking available space on the ASM diskgroups
Long description:	
Version:	1.1
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Extent fragmentation

Name:	Extent fragmentation
Platform:	Oracle
Category:	Capacity
Description:	Checking for extent fragmentation in dba_free_space
Long description:	
Version:	1.23
Default schedule:	12h
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Flash recovery space usage (noschema)

Name:	Flash recovery space usage
Platform:	Oracle
Category:	Capacity
Description:	Checking available space on the Flash recovery area
Long description:	
Version:	1.0
Default schedule:	25m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Max processes (noschema)

Name:	Max processes
Platform:	Oracle
Category:	Capacity
Description:	Checking if database reaches max processes (processes) parameter
Long description:	
Version:	1.0
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

SYS.AUD size

Name:	SYS.AUD size
Platform:	Oracle
Category:	Capacity
Description:	Checking if SYS.AUD$ fills up with data
Long description:	
Version:	1.1
Default schedule:	3h
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Tablespace free space (noschema)

Name:	Tablespace free space
Platform:	Oracle
Category:	Capacity
Description:	Checking tablespaces for free space
Long description:	
Version:	1.5
Default schedule:	2h
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Segment size collector (all segments -- aggregate)

Name:	Segment size collector (all segments — aggregate)
Platform:	Oracle
Category:	Capacity
Description:	Collects size and number totals on all segment types per segment-owner and tablespace_name.
Long description:	Task collects size and number total statistics on Oracle 10g or higher for all – (aggregate) type of segments per segment-owner and tablespace_name.
Version:	3.1
Default schedule:	50 6 1,5 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above = '1' & _priv_read_dba_segments = '1' & _priv_read_dba_free_space = '1' & _priv_read_dba_tablespaces = '1' & _priv_read_dba_temp_files = '1']]]

Segment size collector (all segments -- aggregate 9i)

Name:	Segment size collector (all segments — aggregate)	
Platform:	Oracle	
Category:	Capacity	
Description:	Collects size and number totals on all segment types per segment-owner and tablespace_name.	
Long description:	Task collects size and number total statistics on Oracle 9i for all – (aggregate) type of segments per segment-owner and tablespace_name.	
Version:	2.1	
Default schedule:	50 6 1,5 *	
Requires engine install:	Yes	
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '8%'	version like '9%')]]]

Segment size collector (large segments -- detail)

Name:	Segment size collector (large segments — detail)
Platform:	Oracle
Category:	Capacity
Description:	Collects size and extent number info for larger segments in the database.
Long description:	Task collects information (size and extent number) for the largest segments in the database.
Version:	2.0
Default schedule:	55 6,17 1,5 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Segment size status for old style tablespaces

Name:	Segment size status for old style tablespaces		
Platform:	Oracle		
Category:	Capacity		
Description:	Checks segment storage definitions in tablespaces where extent management is not set to local.		
Long description:	Task checks segment storage definitions in tablespaces where extent management is not set to local.		
Version:	1.8		
Default schedule:	40 6,17 * *		
Requires engine install:	Yes		
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10%'	version like '8%'	version like '9%')]]]

Temporary tablespace free space check

Name:	Temporary tablespace free space check
Platform:	Oracle
Category:	Capacity
Description:	Checks level of free space for temporary tablespaces.
Long description:	Task checks the level of free space for temporary tablespaces.
Version:	0.5
Default schedule:	45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_dba_tablespace_usage_metrics = '1' & _priv_read_dba_free_space = '1' & _priv_read_dba_tablespaces = '1' & _priv_read_dba_temp_files = '1' & _priv_read_dba_temp_free_space='1']]]

Tablespace free space check

Name:	Tablespace free space check
Platform:	Oracle
Category:	Capacity
Description:	Checks level of free space for all tablespaces.
Long description:	Task checks the level of free space for all tablespaces.
Version:	3.2
Default schedule:	45 6,11,18 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above = '1' & _priv_read_dba_tablespace_usage_metrics = '1' & _priv_read_dba_free_space = '1' & _priv_read_dba_tablespaces = '1' & _priv_read_dba_tablespaces = '1']]]

Blocking detector for RAC

Name:	Blocking detector for RAC
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checks if a session is waiting on a TX (transaction) lock. (RAC only)
Long description:	Task checks if a session is waiting on a TX (transaction) lock. (RAC only)
Version:	2.8
Default schedule:	0,4,8,12,16,20,24,28,32,36,40,44,48,52,56 6-18 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & iscluster='1' & eleven_and_above='1' & _priv_read_gv_lock = '1' & _priv_read_gv_session = '1' & _priv_read_gv_sqlarea = '1' & _priv_read_dba_objects = '1']]]

Memory session load for RAC

Name:	Memory session load for RAC
Platform:	Oracle
Category:	Cluster and Replication
Description:	Records the load memory of RAC active sessions over time.
Long description:	Task records the load memory of RAC active sessions over time.
Version:	2.4
Default schedule:	10,30,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & iscluster='1' & ten_and_above='1' & _priv_read_gv_session = '1']]]

MV Refresh Group(s)

Name:	MV Refresh Group(s)			
Platform:	Oracle			
Category:	Cluster and Replication			
Description:	Checks refresh date of scheduled jobs in refresh group(s).			
Long description:	Task checks refresh date of all scheduled jobs in refresh group(s).			
Version:	1.3			
Default schedule:	• * * *			
Requires engine install:	Yes			
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10%'	version like '11%'	version like '8%'	version like '9%')]]]

Dataguard apply time

Name:	Dataguard apply time
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checking apply time in dataguard standby nodes
Long description:	
Version:	1.0
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Dataguard archive sequence apply lag

Name:	Dataguard archive sequence apply lag
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checking lag of applied archive logs on dataguard standby node
Long description:	
Version:	1.1
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Dataguard standby archivelog gap

Name:	Dataguard standby archivelog gap
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checking for archivelog gap between threads on primary and standby instances in a replication/service group. Metadata “replicationgroup” must be set on all instances to the same as parameter replicationgroup. For upto 10 redo threads.
Long description:	
Version:	1.0
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]]]

Dataguard standby archivelog gap

Name:	Dataguard standby archivelog gap
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checking for archivelog gap between threads on primary and standby instances in a replication/service group. Metadata “replicationgroup” must be set on all instances to the same as parameter replicationgroup. For upto 10 redo threads.
Long description:	
Version:	1.0
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]]]

Dataguard startup time

Name:	Dataguard startup time
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checking startup time on dataguard standby nodes
Long description:	
Version:	1.0
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Dataguard transport time

Name:	Dataguard transport time
Platform:	Oracle
Category:	Cluster and Replication
Description:	Checking transport time in dataguard standby nodes
Long description:	
Version:	1.0
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

RAC instances status

Name:	RAC instances status
Platform:	Oracle
Category:	Cluster and Replication
Description:	Reacts if one or more RAC instances are not available.
Long description:	Reacts if one or more RAC instances are not available.
Version:	1.0
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & iscluster = '1']]]

Session load RAC

Name:	Session load Rac
Platform:	Oracle
Category:	Cluster and Replication
Description:	Records the number of active sessions over time (RAC only).
Long description:	Task records the number of active sessions over a period of time (RAC only). Can be configured to give alarms/warnings if the number of total sessions exceeds threshold values.
Version:	2.8
Default schedule:	5,10,15,20,25,30,35,40,45,50,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & iscluster='1' & ten_and_above='1' & _priv_read_gv_session = '1']]]

Snapshot Log(s) rows count

Name:	Snapshot Log(s) rows count		
Platform:	Oracle		
Category:	Cluster and Replication		
Description:	Checks row count on all snapshot logs for given schema.		
Long description:	Task checks the row count of all snapshot logs for a given schema.		
Version:	1.3		
Default schedule:	30 6,12,18 * *		
Requires engine install:	Yes		
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '9%'	version like '10%'	version like '11%') & _priv_read_dba_segments = '1']]]

Snapshot Log(s) size

Name:	Snapshot Log(s) size		
Platform:	Oracle		
Category:	Cluster and Replication		
Description:	Checks size of all snapshot logs.		
Long description:	Task checks the size of all snapshot logs.		
Version:	2.3		
Default schedule:	10 6,12,18 * *		
Requires engine install:	Yes		
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '9%'	version like '10%'	version like '11%')]]]

Data Guard role switch

Name:	Data Guard role switch
Platform:	Oracle
Category:	Cluster and Replication
Description:	This job checks if Primary/Standby switch occurs.
Long description:	
Version:	1.5
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Restore uptodate status

Name:	Restore uptodate status
Platform:	Oracle
Category:	Internal
Description:	Checking if database restore is uptodate
Long description:	
Version:	1.1
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']]]

Framework

Name:	Framework
Platform:	Oracle
Category:	Maintenance Maintenance
Description:	Engine framework is used when upgrading earlier versions of dbWatch. Ensures the dbWatch Engine and dbWatch Server run compatible codes.
Long description:	Engine framework is used when upgrading earlier versions of dbWatch Engine Framework. This task makes sure that the dbWatch Engine and the dbWatch Server run compatible codes. It also auto-tunes the dbWatch schema.
Version:	3.3
Default schedule:	50 5 1 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1']]]

Analyze tables

Name:	Analyze tables
Platform:	Oracle
Category:	Maintenance
Description:	Analyse tables automaticly.
Long description:	Analyses tables to improve performance
Version:	1.8
Default schedule:	15 0 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1']]]

Delete SYS.AUD$ data

Name:	Delete SYS.AUD$ data
Platform:	Oracle
Category:	Maintenance
Description:	Deletes old rows from the SYS.AUD$ trail table.
Long description:	Deletes old rows from the SYS.AUD$ trail table. Configurable time delay in days.
Version:	1.1
Default schedule:	15 3 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1']/.[is_rds = '0']]]

Rebuild indexes

Name:	Rebuild indexes
Platform:	Oracle
Category:	Maintenance
Description:	Rebuild indexs automaticly.
Long description:	Rebuild indexs based on structural statistics to improve performance
Version:	1.6
Default schedule:	15 2 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1']]]

Statistics migation

Name:	Statistics migration
Platform:	Oracle
Category:	License and compliance
Description:	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
Long description:	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
Version:	1.0
Default schedule:	1 1 1 1
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & _priv_read_v_datafile = '1' & _priv_read_v_tempfile = '1']]]

Blocking detector

Name:	Blocking detector
Platform:	Oracle
Category:	Performance
Description:	Checks if a session is waiting on a TX (transaction) lock.
Long description:	Task checks if a session is waiting on a TX (transaction) lock.
Version:	3.0
Default schedule:	0,4,8,12,16,20,24,28,32,36,40,44,48,52,56 6-18 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Buffer cache statistics

Name:	Buffer cache statistics
Platform:	Oracle
Category:	Performance
Description:	Gets data cache statistics for buffer cache in SGA.
Long description:	Task gets data cache statistic for buffer cache in SGA.
Version:	2.2
Default schedule:	1,11,21,31,41,51 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

CPU load

Name:	CPU load
Platform:	Oracle
Category:	Performance
Description:	Checks CPU load using Oracle performance view v$osstat.
Long description:	Checks and collects CPU load statistics. The task also aggregates statistics for three different periods of the day: night hours (NW 00:00-08:00), working hours (WH 08:00-16:00) and evening hours (EH 16:00-00:00).
Version:	3.6
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1']]]

Database network statistics (SQL*NET)

Name:	Database network statistics (SQL*NET)
Platform:	Oracle
Category:	Performance
Description:	Checks database network SQL*NET statistics.
Long description:	Task checks database network SQL*NET statistics.
Version:	2.0
Default schedule:	7,17,27,37,47,57 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Test DML performance

Name:	Test DML performance
Platform:	Oracle
Category:	Performance
Description:	Runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on schema test tables.
Long description:	Task runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on the dbWatch engine schema test tables.
Version:	1.9
Default schedule:	10 22 5 1,30
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & can_get_grant_execute_sys_dbms_lock='1']]]

File IO statistics

Name:	File IO statistics
Platform:	Oracle
Category:	Performance
Description:	Collects I/O statistics for data files.
Long description:	Returns I/O statistics for data files baseed on dynamic performance view gv$datafile
Version:	2.9
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above='1']]]

Instance memory statistics

Name:	Instance memory statistics
Platform:	Oracle
Category:	Performance
Description:	Collects statistics on instance memory.
Long description:	Task collects statistics on instance memory.
Version:	1.51
Default schedule:	3,13,23,33,43,53 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Latch statistics

Name:	Latch statistics
Platform:	Oracle
Category:	Performance
Description:	Collects latch status statistics.
Long description:	Task collects latch status statistics.
Version:	1.6
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Long running queries

Name:	Long running queries
Platform:	Oracle
Category:	Performance
Description:	Checks any long running queries and then alerts.
Long description:	Task checks for long running queries and then alerts.
Version:	3.4
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & eleven_and_above='1']]]

Open cursors

Name:	Open cursors
Platform:	Oracle
Category:	Performance
Description:	Checks and collects the amount of open cursors per session.
Long description:	Task checks and collects the amount of open cursors performed per session.
Version:	1.7
Default schedule:	10,30,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Disk read statistics

Name:	Disk read statistics
Platform:	Oracle
Category:	Performance
Description:	Collects disk read statistics from v$system_event performance view for db file sequential and dbfile scattered read event.
Long description:	Task collects disk read statistics from v$system_event performance view for db file sequential read event (a single-block read, for example index fetch by ROWID) and a db file scattered read event (a multiblock read, for example a full-table scan).
Version:	2.4
Default schedule:	4,14,24,34,44,54 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Redo statistics

Name:	Redo statistics
Platform:	Oracle
Category:	Performance
Description:	Gets redolog files statistics. Based on Oracle dictionary and performance views.
Long description:	Task gets redolog files statistics. Based on Oracle dictionary and performance views.
Version:	3.5
Default schedule:	6,16,26,36,46,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & nine_and_above='1']]]

Redo statistics (8i)

Name:	Redo statistics
Platform:	Oracle
Category:	Performance
Description:	Gets redolog files statistics. Based on Oracle dictionary and performance views.
Long description:	Task gets redolog files statistics. Based on Oracle dictionary and performance views.
Version:	1.7
Default schedule:	6,16,26,36,46,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '8%')]]]

Session load

Name:	Session load
Platform:	Oracle
Category:	Performance
Description:	Records the number of active and inactive sessions over time.
Long description:	Records the number of active and inactive sessions over time. Can be configured to give alarms/warnings if the number of total sessions exceeds threshold values.
Version:	3.4
Default schedule:	10,30,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above='1']]]

Session load (8i and 9i)

Name:	Session load	
Platform:	Oracle	
Category:	Performance	
Description:	Records the number of active sessions over time.	
Long description:	Task records the number of active sessions over a period of time.	
Version:	1.7	
Default schedule:	10,30,50 * * *	
Requires engine install:	Yes	
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '8%'	version like '9%')]]]

SQL waits

Name:	SQL waits	
Platform:	Oracle	
Category:	Performance	
Description:	Collects SQL wait statistics.	
Long description:	Task collects SQL wait statistics.	
Version:	2.8	
Default schedule:	50 0,4,8,12,16,20 * *	
Requires engine install:	Yes	
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (version like '10.2'	eleven_and_above = '1')]]]

Table statistics check

Name:	Table statistics check
Platform:	Oracle
Category:	Performance
Description:	Reacts on old (or missing) statistics in DBA_TABLES dictionary table (column LAST_ANALYZED).
Long description:	Reacts on old (or missing) statistics in DBA_TABLES dictionary table (column LAST_ANALYZED).
Version:	1.5
Default schedule:	10 7 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[ten_and_above='1' & hasengine='YES']]]

Top user memory usage

Name:	Top user memory usage
Platform:	Oracle
Category:	Performance
Description:	Check can be used to trace memory usage per server-process (session).
Long description:	This task check can be used to trace memory usage per server-process (session).
Version:	2.5
Default schedule:	6,16,26,36,46,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & ten_and_above = '1']]]

Undo statistics

Name:	Undo statistics
Platform:	Oracle
Category:	Performance
Description:	Collects rollback segment statistics.
Long description:	Task collects rollback segment statistics.
Version:	2.1
Default schedule:	8,18,28,38,48,58 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

User memory statistics

Name:	User memory statistics
Platform:	Oracle
Category:	Performance
Description:	Collects user memory statistics.
Long description:	Task collects user memory statistics.
Version:	2.4
Default schedule:	7,17,27,37,47,57 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES']]]

Wait statistics

Name:	Wait statistics	
Platform:	Oracle	
Category:	Performance	
Description:	Collects waits statistics encountered by threads that executed. This task is based on the v$system_event dynamic performance view.	
Long description:	Collects waits statistics encountered by threads that executed. This task is based on the v$system_event dynamic performance view.	
Version:	2.8	
Default schedule:	2,7,12,17,22,27,32,37,42,47,52,57 * * *	
Requires engine install:	Yes	
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & (eleven_and_above='1'	version like '10.2%')]]]

SQL statistics

Name:	SQL statistics
Platform:	Oracle
Category:	Performance
Description:	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views.
Long description:	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views. To calculate similarities between SQL statements, the Oracle internal procedure UTL_MATCH.EDIT_DISTANCE_similarity is used.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & is_rds=0]]]

SQL statistics (RDS)

Name:	SQL statistics
Platform:	Oracle
Category:	Performance
Description:	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views.
Long description:	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views. To calculate similarities between SQL statements, the Oracle internal procedure UTL_MATCH.EDIT_DISTANCE_similarity is used.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & is_rds=1]]]

RMAN recovery area backup status

Name:	RMAN recovery area backup status
Platform:	Oracle
Category:	Availability
Description:	Checks the status of RMAN recovery area backup (input_type ‘RECVR AREA’) from V$RMAN_BACKUP_JOB_DETAILS performance view.
Long description:	Checks the status of RMAN recovery area backup (input_type ‘RECVR AREA’) from V$RMAN_BACKUP_JOB_DETAILS performance view.
Version:	1.0
Default schedule:	25 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’ & _cdb_v_rman_backup_job_details = ‘1’ & is_pluggdb = ‘0’

Parameters		
Name	Default value	Description

Tablespace in BEGIN BACKUP mode

Name:	Tablespace in BEGIN BACKUP mode
Platform:	Oracle
Category:	Availability
Description:	Checks if any of the tablespaces are in ‘BEGIN BACKUP’ mode (caused by DDL ‘alter tablespace
Long description:	Checks if any of the tablespaces are in ‘BEGIN BACKUP’ mode (caused by DDL ‘alter tablespace
Version:	1.2
Default schedule:	10 10,15 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘oracle’]/.[hasengine=‘YES’

Parameters		
Name	Default value	Description

Applied archive log gap status

Name:	Applied archive log gap status
Platform:	Oracle
Category:	Cluster and Replication
Description:	Reacts on applied archived log gaps and errors.
Long description:	Reacts on applied archived log gaps and errors.
Version:	1.0
Default schedule:	4,14,24,34,44,54 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES'

Parameters		
Name	Default value	Description

SQL statistics (CDB)

Name:	SQL statistics
Platform:	Oracle
Category:	Performance
Extra cost package:	SQL Performance package
Description:	Collects SQL statements statistics (for container database) from V$SQLAREA and V$SQL_PLAN dynamic performance views.
Long description:	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views. To calculate similarities between SQL statements, the Oracle internal procedure UTL_MATCH.EDIT_DISTANCE_similarity is used.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & is_rds=0 & eleven_and_above='1' & is_containerdb='1'

Parameters		
Name	Default value	Description
history threshold	7	The maximum number of days to keep statistics for, in the repository tables. Remember that it can quickly take several GB of space for one week of statistics. It is important to monitor space consumption in the Management part of dbWatch Monitor GUI.
dbwatch tablespace max size	5000	Maximum size (in MB) of dbwatch tablespace.
collect execution plan	YES	Collects execution plan statistics into the history table if set to "YES".
collect internal statistics	30	How often (in minutes) size of internal objects should be checked. "SQL statistics" job collects a large amount of data in the dbWatch schema tables, so it is important to keep track of space usage in the dbWatch tablespace.
SQL	80	Maximum space consumption (in percentage) of maximum size of the dbWatch

repository max size		tablespace for internal/repository objects that contain statistics for SQL statements.
min buffer gets	100	Minimum buffer gets value (per execution) for a SQL statement to be registered into history tables (repository tables).
check similarities	YES	Check similarities between SQL statements while collecting statistics. This option can be switched off if the procedure consumes too much time and resources. To calculate similarities between SQL statements, the Oracle internal procedure UTL_MATCH.EDIT_DISTANCE_similarity is used
return status	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when reached maximum space consumption (in percentage) of maximum size of the dbWatch tablespace (parameter “SQL repository max size”)

SQL statistics (PDB)

Name:	SQL statistics
Platform:	Oracle
Category:	Performance
Extra cost package:	SQL Performance package
Description:	Collects SQL statements statistics (for pluggable database) from V$SQLAREA and V$SQL_PLAN dynamic performance views.
Long description:	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views. To calculate similarities between SQL statements, the Oracle internal procedure UTL_MATCH.EDIT_DISTANCE_similarity is used.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='oracle']/.[hasengine='YES' & is_rds=0 & eleven_and_above='1' & is_pluggdb='1'

Parameters		
Name	Default value	Description
history threshold	7	The maximum number of days to keep statistics for, in the repository tables. Remember that it can quickly take several GB of space for one week of statistics. It is important to monitor space consumption in the Management part of dbWatch Monitor GUI.
dbwatch tablespace max size	5000	Maximum size (in MB) of dbwatch tablespace.
collect execution plan	YES	Collects execution plan statistics into the history table if set to "YES".
collect internal statistics	30	How often (in minutes) size of internal objects should be checked. "SQL statistics" job collects a large amount of data in the dbWatch schema tables, so it is important to keep track of space usage in the dbWatch tablespace.
SQL	80	Maximum space consumption (in percentage) of maximum size of the dbWatch

repository max size		tablespace for internal/repository objects that contain statistics for SQL statements.
min buffer gets	100	Minimum buffer gets value (per execution) for a SQL statement to be registered into history tables (repository tables).
check similarities	YES	Check similarities between SQL statements while collecting statistics. This option can be switched off if the procedure consumes too much time and resources. To calculate similarities between SQL statements, the Oracle internal procedure UTL_MATCH.EDIT_DISTANCE_similarity is used
return status	1	Return status value (ALARM – 2, WARNING – 1, or OK – 0) when reached maximum space consumption (in percentage) of maximum size of the dbWatch tablespace (parameter “SQL repository max size”)

Jobs for MS SQL Server

Control Center Jobs	Description
Availability	
Agent Jobs Check	Checks whether there exists jobs on the SQL server which have not been executed, or have failed during execution.
Agent Jobs Check *for MS2000*	Checks whether there exists jobs on the SQL server which have not been executed or have failed during execution.
SQL Server Agent status	Checks if the SQL Server Agent is running.
Database backup *for MS2000*	This procedure analyzed the backup statistics from msdb.dbo.backupset table (excluding system databases: master, model and msdb).
Database backup *for system dbs MS2000*	This procedure analyzed the system database backups statistics from msdb.dbo.backupset table.
Database backup	This procedure analyzes the backup statistics (type D and I) from the msdb.dbo.backupset table (excluding the system databases: master, model and msdb).
Database backup *for SIMPLE recovery model*	This procedure analyzes the backup statistics (type D) from the msdb.dbo.backupset table for databases in SIMPLE recovery model (excluding the system databases: master, model and msdb).
Database backup *for system databases*	This procedure analyzes the FULL backup statistics for the system databases (master, model and msdb) using data from the msdb.dbo.backupset table.
Database Log backup	This procedure checks the database transaction log backups from the msdb.dbo.backupset table (excluding system databases: master, model and msdb).
Collation check	Checks if there is a collation conflict with temp tables and table variables.
Check database recovery mode	Checks if all databases are running in FULL or SIMPLE recovery mode.
Database status	Checks if all databases have status ONLINE
Database status *for MS2000*	Checks if all databases have status ONLINE
Database Server uptime	Collects database server uptime statistics.
Database Server uptime *for MS2000*	Collects database server uptime statistics.
Database Server uptime *for MS2005*	Collects database server uptime statistics.
Instance error log	Reads and checks the Instance error log file by using the sp_readerrorlog stored

	procedure.
Instance error log *for AWS*	Reads and checks the Instance error log file by using the AWS rdsadmin.dbo.rds_read_error_log stored procedure.
Instance status	This alert checks if the instance has been restarted since the last check
Database backup *noschema*	This alert checks if the instance has been restarted since the last check
Database log backup *noschema*	This alert checks if the instance has been restarted since the last check
Database status *noschema*	This alert checks if the instance has been restarted since the last check
Missing database backup *noschema*	This alert checks if the instance has been restarted since the last check
Missing database log backup *noschema*	This alert checks if the instance has been restarted since the last check
Program status	Checks for any program connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Report Server status	Checks if the program 'Report Server' is connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Restricted Enterprise edition features	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Restricted Enterprise edition features *for MS2000*	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Restricted Enterprise edition features *for MS2005*	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Capacity	
Auto growth event collector	Collects statistics on how often an auto-growth event has occurred.
Data file size check	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Data file size check *for MS2000*	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Database growth rate *aggregated*	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.
Database growth rate *aggregated for MS2000*	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.

Database growth rate *detailed*	Collects size of database files (including transaction log files) to visualize the growth rate for the largest databases.
Database growth rate *detailed for MS2000*	Collects size of database files (including transaction log files) to visualize the growth rate for the largest databases.
Database disk capacity	Checks free space on drives where data and transaction log files are defined. An alarm (or warning) is raised if the percentage limit is reached OR if the abosulte limit is reached. IF the xp_cmdshell Instance configuration option is enabled the alert can check disk and mounted volumes where data-files are not present.
Database disk space usage	Checks free space on drives where all data and transaction log files are defined. Drives where no data files exist will be ignored.
Databases NOT IN USE collector	Collects information about most inactive databases.
Disk space check	Checks the amount of free space on the available disk drives.
Instance error log file size check	Checks the size og the Instance error log file by using the extended stored procedure xp_cmdshell.
Filegroups growth rate	This procedure collects space usage for the largest filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined on a SQL Server instance.
Filegroups growth rate *for MS2000*	This procedure collects space usage for the largest filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined on a SQL Server instance.
Data file size check *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Database disk capacity *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Disk space check *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Transaction log space usage *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Objects size collector *all*	Collects table and index size information for the largest objects for all databases.

databases	
Temporary database space usage	Checks space usage in tempdb database, and collects statistics including size of data and transaction log files.
Transaction log size check	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Transaction log size check	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Transaction log space usage	This procedure checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Transaction log space usage *for MS2000, MS2005 and MS2008*	This procedure checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Truncate transaction log	This procedure truncates the transaction log for the dbWatch database.
Version store space usage	Checks total space in tempdb used by version store records for each database.
Cluster	
Database mirroring	Checks state information of all mirrored databases.
Failover cluster host switch	Checks if an instance switched to a differen host i a Windows Server Failover Cluster (WSFC).
Health state	Checks groups health state.
Log shipping monitor	Monitor the primary database in each log shipping configuration, including information about the last backup file and last restored file.
Log shipping monitor	Monitor the secondary database in each log shipping configuration.
Member state	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
AlwaysOn database backup alert	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
AlwaysOn transaction log backup alert	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
Replica states	Checks role state for all alwayson groups.
Replication status	The following check provides general info in regards to any replication going on in a server.
Compliance *optional*	

MS SQL Server patch status	Checks latest updates for Microsoft SQL Server.
Consolidation *optional*	
Database statistics	This procedure gathers statistics per database.
Internal *optional*	
This alert returns warning/alarm if there are several dbWatch engines installed on the same instance.	
dbWatch Server activity alert	This alert returns warning/alarm if the dbWatch Server is using consumes a lot of cpu over a period of time.
Maintenance	
Autogrow settings	Checks database files auto-growth settings.
framework	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
framework *for MS2000*	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Maintenance *optional*	
Backup All databases	Takes backup of all application and system databases.
Backup All transaction logs	Takes backup of all transaction logs for databases running in FULL recovery mode.
Check database and server principal mapping	Checks if the database owner (dbo) is maped into any Server Login (server principal).
Cleanup MSDB history tables	This task deletes entries in the MSDB database history tables which holds statistics of backup/restore, jobs and maintenance plan executions.
Cycle error log	This task cycle MS SQL Server error log and Agent error log files.
DBCC CHECKDB	Checks the logical and physical integrity of all the objects in all databases by performing the DBCC CHECKDB operation.
DBCC UPDATEUSAGE	This task is performing the DBCC UPDATEUSAGE operation to corrects pages and row count inaccuracies in the catalog views. These inaccuracies may cause incorrect space usage reports returned by the sp_spaceused system stored procedure.
External fragmentation *all databases*	External (logical) fragmentation occurs when an index leaf page is not in logical order. It occurs when the logical ordering of the index does not match the physical ordering of the index. This causes SQL Server to perform extra work to return ordered results

Internal fragmentation check	Checks the internal fragmentation for tables and indexes in all databases. The information is extracted from the dynamic management function (view) sys.dm_db_index_physical_stats.
SQL Server performance counters	Checks if SQL Server performance counters are missing.
Rebuild indexes	Rebuilds fragmented indexes in all databases.
Rebuld indexes in table	Rebuilds fragmented indexes in all tables listed in the 'table list' parameter.
Reorganize indexes	Reorganizes fragmented indexes in all databases.
Reorganize indexes in table	Reorganizing fragmented indexes in all tables listed in the 'table list' parameter.
Shrink transaction logs	This procedure shrinks transaction log files which are detected by 'Transaction log size check' alert which cheks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size).
Suspect pages	Monitors suspect pages statistics in suspect_pages table.
Update index statistics	Update statistics in all (non-system) databases.
Update statistics	Update statistics in all (non system) databases.
Migration *optional*	
Statistics migration	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
Performance	
Blocking detector	Checks whether there exists blocked session.
Blocking statistics	Checks whether there exists any blocked sessions.
Cached query statistics	This task returns aggregate performance statistics based on cached query plans in SQL Server.
Data cache memory usage	Collects data cache memory usage per database (for top 10 databases).
Data cache memory usage *for MS20005 and MS2008*	Collects data cache memory usage per database (for top 10 databases).
Data hit ratio	Monitors the buffer cache hit ratio by extracting counter values from the master.dbo.sysperfinfo table for the counters 'Buffer cache hit ratio' and 'Buffer cache hit ratio base'.
Database session load	Shows the number of connections over time per database, host and application.
Test DML-DDL performance	Runs performance test on the database. The procedure executes SELECT, INSERT, UPDATE, DELETE (and TRUNCATE) statements on the test table.
File IO statistics	Shows the number of connections over time per database, host and application.

High activity monitor	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
High activity monitor *for MS2005*	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
Index usage statistics *all databases*	This procedure collects statistics from sys.dm_db_index_usage_stats performance view which gives information on how an index (or a table – heap) has been used to resolve queries.
Instance memory check	This job checks the target memory value of the SQL Server instance, and gives a warning/alarm if the instance is not able to allocate a
Instance memory usage	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Instance memory usage *for MS2005 and MS2008*	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Lazy writer and Checkpoint statistics	Monitors ‘Checkpoint pages/sec’, ‘Lazy writes/sec’ and ‘Page life expectancy’ by extracting counter values from the sys.dm_os_performance_counters performace table.
Memory objects statistics	This task collects the size of memory objects that are currently allocated by the SQL Server
Blocking detector *noschema*	Checks whether there exists any blocked sessions.
Instance memory check *noschema*	The alert checks the target memory value of the SQL Server instance, and gives a warning/alarm if the instance is not able to allocate a certain percentage of the total server/machine memory.
Server memory statistics	This tasks collects statistics from the sys.dm_os_sys_info performance view about the memory resources available to and consumed by the SQL Server.
Sessions per database	This task returns aggregate performance statistics based on sessions connected to the SQL Server instance.
Session load	Shows the number of active sessions over time.
Transactions log flushed bytes load	Shows bytes flushed to transaction logs over time, total and top 5 databases.
Transactions log flushed bytes load *for MS2000*	Shows bytes flushed to transaction logs over time, total and top 5 databases.
Transactions load	Shows the transactions load over time, total and top 5 databases.
Wait statistics	Collects statistics about all waits encountered by threads that executed. This task is based on the sys.dm_os_wait_stats dynamic performance view.
Performance *optional*	

SQL statistics	Collects performance statistics for the SQL statements.
Query Store status	Checks Query Store space usage in every database where it is enabled.

Agent Jobs Check

Name:	Agent Jobs Check
Platform:	Sqlserver
Category:	Availability
Description:	Checks whether there exists jobs on the SQL server which have not been executed, or have failed during execution.
Long description:	Checks whether there exists jobs on the SQL server which have not been executed, or have failed during execution.
Version:	2.8
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Agent Jobs Check (MS2000)

Name:	Agent Jobs Check (MS2000)
Platform:	Sqlserver
Category:	Availability
Description:	Checks whether there exists jobs on the SQL server which have not been executed or have failed during execution.
Long description:	
Version:	1.4
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

SQL Server Agent status

Name:	SQL Server Agent status
Platform:	Sqlserver
Category:	Availability
Description:	Checks if the SQL Server Agent is running.
Long description:	This alert checks the master.dbo.sysprocesses table for the program_name value for the SQL Server Agent process. The program name value is controlled by the dbWatch parameter ‘sqlserver agent process name’. Besides this parameter values, the following program names are checked too: ‘SQLAgent – Generic Refresher’ and ‘SQLAgent – Job invocation engine’.
Version:	1.2
Default schedule:	0,15,30,45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘1999’ & hasengine=‘YES’ & engine_edition = ‘Microsoft SQL Server’ & eng_inst_priv = 0

Database backup (MS2000)

Name:	Database backup (2000)
Platform:	Sqlserver
Category:	Availability
Description:	This procedure analyzed the backup statistics from msdb.dbo.backupset table (excluding system databases: master, model and msdb).
Long description:	
Version:	1.2
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & alwayson_active='NO' & engine_edition = 'Microsoft SQL Server']]]

Database backup (system dbs)

Name:	Database backup (system dbs)
Platform:	Sqlserver
Category:	Availability
Description:	This procedure analyzed the system database backups statistics from msdb.dbo.backupset table.
Long description:	
Version:	1.1
Default schedule:	05 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & alwayson_active='NO' & engine_edition = 'Microsoft SQL Server']]]

Database backup

Name:	Database backup
Platform:	Sqlserver
Category:	Availability
Description:	This procedure analyzes the backup statistics (type D and I) from the msdb.dbo.backupset table (excluding the system databases: master, model and msdb).
Long description:	This procedure analyzes the backup statistics (type D and I) from the msdb.dbo.backupset table (excluding the system databases: master, model and msdb).
Version:	2.1
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & alwayson_active='NO' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database backup (SIMPLE)

Name:	Database backup (SIMPLE)
Platform:	Sqlserver
Category:	Availability
Description:	This procedure analyzes the backup statistics (type D) from the msdb.dbo.backupset table for databases in SIMPLE recovery model (excluding the system databases: master, model and msdb).
Long description:	
Version:	2.1
Default schedule:	50 5 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & alwayson_active='NO' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database backup (system databases)

Name:	Database backup (system databases)
Platform:	Sqlserver
Category:	Availability
Description:	This procedure analyzes the FULL backup statistics for the system databases (master, model and msdb) using data from the msdb.dbo.backupset table.
Long description:	This procedure analyzes the FULL backup statistics for the system databases (master, model and msdb) using data from the msdb.dbo.backupset table.
Version:	1.4
Default schedule:	25 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & alwayson_active='NO' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database log backup

Name:	Database Log backup
Platform:	Sqlserver
Category:	Availability
Description:	This procedure checks the database transaction log backups from the msdb.dbo.backupset table (excluding system databases: master, model and msdb).
Long description:	
Version:	2.0
Default schedule:	20 0,2,4,6,8,10,12,14,16,18,20,22 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & alwayson_active='NO' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Collation check

Name:	Collation check
Platform:	Sqlserver
Category:	Availability
Description:	Checks if there is a collation conflict with temp tables and table variables.
Long description:	You may get a collation conflict when the collation of your user database does not match the collation of tempdb database. Whenever you join with the temp table (without specifing the collation for string columns) they will inherit the default collation for tempdb databae causing collation conflict.
Version:	1.4
Default schedule:	30 7 7 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Check database recovery mode

Name:	Check database recovery mode
Platform:	Sqlserver
Category:	Availability
Description:	Checks if all databases are running in FULL or SIMPLE recovery mode.
Long description:	Checks if all databases are running in FULL or SIMPLE recovery mode. This procedure can be configured to change the recovery mode for a database to a value defined by the configuration parameter ‘allow recovery mode’ (and then switch back to status OK). Only ONLINE databases are checked.
Version:	1.61
Default schedule:	5 5,19 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2000’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Database status

Name:	Database status
Platform:	Sqlserver
Category:	Availability
Description:	Checks if all databases have status ONLINE
Long description:	A database state specifies the current running mode of that database. The database can be running in one state at a given time and there are seven main states in which a database can exist. The current state of a database can be checked by selecting the state_desc column of the sys.databases catalog view and can have following values: ONLINE, OFFLINE, RESTORING, RECOVERING, RECOVERY PENDING, SUSPECT and EMERGENCY. This alert can be configured to return values (warnings and alarms) based on different states of the database.
Version:	1.5
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database status (MS2000)

Name:	Database status
Platform:	Sqlserver
Category:	Availability
Description:	Checks if all databases have status ONLINE
Long description:	
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Database server uptime

Name:	Database Server uptime
Platform:	Sqlserver
Category:	Availability
Description:	Collects database server uptime statistics.
Long description:	
Version:	2.2
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2005' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database Server uptime (MS2000)

Name:	Database Server uptime
Platform:	Sqlserver
Category:	Availability
Description:	Collects database server uptime statistics.
Long description:	
Version:	1.9
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Database server uptime (MS2005)

Name:	Database Server uptime
Platform:	Sqlserver
Category:	Availability
Description:	Collects database server uptime statistics.
Long description:	
Version:	2.2
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2005' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Instance error log

Name:	Instance error log
Platform:	Sqlserver
Category:	Availability
Description:	Reads and checks the Instance error log file by using the sp_readerrorlog stored procedure.
Long description:	The sp_readerrorlog stored procedure allows to read the contents of the SQL Server error log file. This procedure checks for errors defined by the ‘error text’ parameter.
Version:	1.4
Default schedule:	6,26,46 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & engine_edition = ‘Microsoft SQL Server’ & eng_inst_priv = 0

Instance error log (AWS)

Name:	Instance error log (AWS)
Platform:	Sqlserver
Category:	Availability
Description:	Reads and checks the Instance error log file by using the AWS rdsadmin.dbo.rds_read_error_log stored procedure.
Long description:	The rdsadmin.dbo.rds_read_error_log stored procedure allows to read the contents of the SQL Server error log file. This procedure checks for errors defined by the ‘error text’ parameter.
Version:	1.4
Default schedule:	6,26,46 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & engine_edition = ‘Microsoft SQL Server’ & eng_inst_priv = 1 & eng_inst_aws_priv = 0 & rds_read_error_log_exists = 1

Instance status

Name:	Instance status
Platform:	Sqlserver
Category:	Availability
Description:	This alert checks if the instance has been restarted since the last check
Long description:	
Version:	1.3
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database backup (noschema)

Name:	Database backup
Platform:	Sqlserver
Category:	Availability
Description:	This alert checks if the instance has been restarted since the last check
Long description:	This job checks the backup statistics (type D and I) from the msdb.dbo.backupset table (excluding the system databases: master, model and msdb).
Version:	2
Default schedule:	10 6 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database log backup (noschema)

Name:	Database log backup
Platform:	Sqlserver
Category:	Availability
Description:	This alert checks if the instance has been restarted since the last check
Long description:	This job checks the transaction log backup statistics (type L) from the msdb.dbo.backupset table (excluding the system databases: master, model and msdb).
Version:	2
Default schedule:	30m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database status (noschema)

Name:	Database status
Platform:	Sqlserver
Category:	Availability
Description:	This alert checks if the instance has been restarted since the last check
Long description:	
Version:	2
Default schedule:	5m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine ='NO' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Missing database backup (noschema)

Name:	Missing database backup
Platform:	Sqlserver
Category:	Availability
Description:	This alert checks if the instance has been restarted since the last check
Long description:	This job checks for each database if backup (type D and I) exists.
Version:	2
Default schedule:	30 7 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Missing database log backup (noschema)

Name:	Missing database log backup
Platform:	Sqlserver
Category:	Availability
Description:	This alert checks if the instance has been restarted since the last check
Long description:	This job checks for each database if transaction log backup (type L) exists.
Version:	2
Default schedule:	45 7 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Program status

Name:	Program status
Platform:	Sqlserver
Category:	Availability
Description:	Checks for any program connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Long description:	
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Report Server status

Name:	Report Server status
Platform:	Sqlserver
Category:	Availability
Description:	Checks if the program ‘Report Server’ is connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Long description:	Checks if the program ‘Report Server’ is connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2008’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Restricted Enterprise edition features

Name:	Restricted Enterprise edition features
Platform:	Sqlserver
Category:	Availability
Description:	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Long description:	
Version:	1.2
Default schedule:	0 6 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Restricted Enterprise edition features (MS2000)

Name:	Restricted Enterprise edition features (2000)
Platform:	Sqlserver
Category:	Availability
Description:	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Long description:	
Version:	1.1
Default schedule:	0 6 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Restricted Enterprise edition features (MS2005)

Name:	Restricted Enterprise edition features 2005
Platform:	Sqlserver
Category:	Availability
Description:	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Long description:	
Version:	1.1
Default schedule:	0 6 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2005' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Auto growth event collector

Name:	Auto growth event collector
Platform:	Sqlserver
Category:	Capacity
Description:	Collects statistics on how often an auto-growth event has occurred.
Long description:	
Version:	1.1
Default schedule:	12 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Data file size check

Name:	Data file size check
Platform:	Sqlserver
Category:	Capacity
Description:	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Long description:	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Version:	2.1
Default schedule:	5 7,12,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Data file size check (MS2000)

Name:	Data file size check
Platform:	Sqlserver
Category:	Capacity
Description:	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Long description:	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Version:	1.9
Default schedule:	5 7,12,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Database growth rate (aggregated)

Name:	Database growth rate (aggregated)
Platform:	Sqlserver
Category:	Capacity
Description:	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.
Long description:	
Version:	1.2
Default schedule:	10 5 5 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database growth rate (aggregated) (MS2000)

Name:	Database growth rate (aggregated)
Platform:	Sqlserver
Category:	Capacity
Description:	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.
Long description:	
Version:	1.2
Default schedule:	15 4 5 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database growth rate (detailed)

Name:	Database growth rate (detailed)
Platform:	Sqlserver
Category:	Capacity
Description:	Collects size of database files (including transaction log files) to visualize the growth rate for the largest databases.
Long description:	Collects size of database files (including transaction log files) to visualize the growth rate for the largest databases.
Version:	1.8
Default schedule:	10 5 1,4 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database growth rate (detailed) (MS2000)

Name:	Database growth rate (detailed)
Platform:	Sqlserver
Category:	Capacity
Description:	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.
Long description:	
Version:	1.2
Default schedule:	10 5 1,4 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database disk capacity

Name:	Database disk capacity
Platform:	Sqlserver
Category:	Capacity
Description:	Checks free space on drives where data and transaction log files are defined. An alarm (or warning) is raised if the percentage limit is reached OR if the abosulte limit is reached. IF the xp_cmdshell Instance configuration option is enabled the alert can check disk and mounted volumes where data-files are not present.
Long description:	From version 1.7 an alarm (or warning) is raised if the percentage limit is reached OR the abosulte limit is reached. Not both as in older versions of this task. IF the xp_cmdshell Instance configuration option is enabled the alert can check disk and mounted volumes where data-files are not present.
Version:	2.2
Default schedule:	15 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2008' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Database disk space usage

Name:	Database disk space usage
Platform:	Sqlserver
Category:	Capacity
Description:	Checks free space on drives where all data and transaction log files are defined. Drives where no data files exist will be ignored.
Long description:	Checks free space on drives where all data and transaction log files are defined. Drives where no data files exist will be ignored.
Version:	1.6
Default schedule:	10 5,15 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Databases NOT IN USE collector

Name:	Databases NOT IN USE collector
Platform:	Sqlserver
Category:	Capacity
Description:	Collects information about most inactive databases.
Long description:	
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Disk space check

Name:	Disk space check
Platform:	Sqlserver
Category:	Capacity
Description:	Checks the amount of free space on the available disk drives.
Long description:	Checks the amount of free space on the available disk drives using an undocumented SQL Server extended stored procedure xp_fixeddrives.
Version:	1.2
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Instance error log file size check

Name:	Instance error log file size check
Platform:	Sqlserver
Category:	Capacity
Description:	Checks the size og the Instance error log file by using the extended stored procedure xp_cmdshell.
Long description:	The xp_cmdshell stored procedure allows to read the contents of a directory.
Version:	1.5
Default schedule:	7 7 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Filegroups growth rate

Name:	Filegroups growth rate
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure collects space usage for the largest filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined on a SQL Server instance.
Long description:	This procedure collects space usage for the largest filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined on a SQL Server instance.
Version:	2.0
Default schedule:	35 6 5 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2000’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Filegroups growth rate (MS2000)

Name:	Filegroups growth rate
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Long description:	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version = ‘2000’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Data file size check (noschema)

Name:	Data file size check
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Long description:	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Version:	1
Default schedule:	90m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[engine_edition = ‘Microsoft SQL Server’

Database disk capacity (noschema)

Name:	Database disk capacity
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Long description:	This job checks free space on drives where data and transaction log files are defined.
Version:	2
Default schedule:	10 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & engine_edition = ‘Microsoft SQL Server’ & eng_inst_priv = 0

Disk space check (noschema)

Name:	Disk space check
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Long description:	Checks the amount of free space on the available disk drives using an undocumented SQL Server extended stored procedure xp_fixeddrives.
Version:	2
Default schedule:	20 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[engine_edition = ‘Microsoft SQL Server’

Transaction log space usage (noschema)

Name:	Transaction log space usage
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Long description:	This job checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Version:	2
Default schedule:	13 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[(engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Object size collector (all databases)

Name:	Objects size collector (all databases)
Platform:	Sqlserver
Category:	Capacity
Description:	Collects table and index size information for the largest objects for all databases.
Long description:	Collects table and index size information for the largest objects for all databases.
Version:	1.4
Default schedule:	30 6 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Temporary database space usage

Name:	Temporary database space usage
Platform:	Sqlserver
Category:	Capacity
Description:	Checks space usage in tempdb database, and collects statistics including size of data and transaction log files.
Long description:	Checks space usage in tempdb database, and collects statistics including size of data and transaction log files.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Transaction log size check

Name:	Transaction log size check
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Long description:	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Version:	2.1
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Transaction log size check (MS2000)

Name:	Transaction log size check
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Long description:	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Version:	2.0
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Transaction log space usage

Name:	Transaction log space usage
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Long description:	The procedure indicates (by warning or alarm) when to back up or truncate the transaction logs. If parameter ‘include unallocated free space’ is set to ‘YES’, the procedure will compute the free space in each transaction log file based on available space in each file plus all unallocated space this file (or files) could autoextend to (maxsize – current size).
Version:	2.1
Default schedule:	10,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Transaction log space usage (MS2000 / MS2005 / MS2008)

Name:	Transaction log space usage
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Long description:	The procedure indicates (by warning or alarm) when to back up or truncate the transaction logs. If parameter ‘include unallocated free space’ is set to ‘YES’, the procedure will compute the free space in each transaction log file based on available space in each file plus all unallocated space this file (or files) could autoextend to (maxsize – current size).
Version:	1.8
Default schedule:	10,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version < ‘2008’ & hasengine=‘YES’ & engine_edition = ‘Microsoft SQL Server’

Truncate transaction log

Name:	Truncate transaction log
Platform:	Sqlserver
Category:	Capacity
Description:	This procedure truncates the transaction log for the dbWatch database.
Long description:	This procedure truncates the transaction log for the dbWatch database.
Version:	1.3
Default schedule:	10 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Version store space usage (tempdb)

Name:	Version store space usage (tempdb)
Platform:	Sqlserver
Category:	Capacity
Description:	Checks total space in tempdb used by version store records for each database.
Long description:	Checks total space in tempdb used by version store records for each database.
Version:	1.1
Default schedule:	15,35,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2016' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database mirroring

Name:	Database mirroring
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks state information of all mirrored databases.
Long description:	Checks state information of all mirrored databases. The Catalog View master.sys.database_mirroring is used to alert if any Principals or Mirrors are in an abnormal state (normal states: SYNCHRONIZED,SYNCHRONIZING).
Version:	1.2
Default schedule:	• 7-17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Failover cluster host switch

Name:	Failover cluster host switch
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks if an instance switched to a differen host i a Windows Server Failover Cluster (WSFC).
Long description:	A Windows Server Failover Cluster (WSFC) is a group of independent servers that work together to increase the availability of applications and services. A warning or an alarm can be triggered if an instance switched to a differen host i a WSFC.
Version:	1.2
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2014' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Health state

Name:	Health state
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks groups health state.
Long description:	
Version:	1.2
Default schedule:	1,6,11,16,21,26,31,36,41,46,51,56 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & alwayson_active='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Log shipping monitor (primary)

Name:	Log shipping monitor (primary)
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Monitor the primary database in each log shipping configuration, including information about the last backup file and last restored file.
Long description:	Monitor the primary database in each log shipping configuration, including information about the last backup file and last restored file.
Version:	1.2
Default schedule:	5,15,25,35,45,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Log shipping monitor (secondary)

Name:	Log shipping monitor (secondary)
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Monitor the secondary database in each log shipping configuration.
Long description:	Monitor the secondary database in each log shipping configuration.
Version:	1.2
Default schedule:	7,17,27,37,47,57 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Member state

Name:	Member state
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
Long description:	
Version:	1.1
Default schedule:	1,6,11,16,21,26,31,36,41,46,51,56 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & alwayson_active='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

AlwaysOn database backup alert

Name:	AlwaysOn database backup alert
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
Long description:	
Version:	2.6
Default schedule:	5 6 * *
Requires engine install:	No
Compatibility tag:	[CDATA[instance[databasetype='sqlserver' & maj_version > '2005' & alwayson_active='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0]

AlwaysOn transaction log backup alert

Name:	AlwaysOn transaction log backup alert
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
Long description:	
Version:	2.6
Default schedule:	5,35 6-18 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & alwayson_active='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Replica states

Name:	Replica states
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	Checks role state for all alwayson groups.
Long description:	
Version:	1.3
Default schedule:	1,6,11,16,21,26,31,36,41,46,51,56 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & alwayson_active='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Replication status

Name:	Replication status
Platform:	Sqlserver
Category:	Cluster and Replication
Description:	The following check provides general info in regards to any replication going on in a server.
Long description:	This procedure analyze MSdistribution_agents table which contains one row for each Distribution Agent running at the local Distributor. This table is stored in the distribution database.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

MS SQL Server patch status

Name:	MS SQL Server patch status
Platform:	Sqlserver
Category:	Compliance
Description:	Checks latest updates for Microsoft SQL Server.
Long description:	Checks latest updates for Microsoft SQL Server.
Version:	1.8
Default schedule:	5 5,19 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Database statistics

Name:	Database statistics
Platform:	Sqlserver
Category:	Availability
Description:	This procedure gathers statistics per database.
Long description:	
Version:	1.8
Default schedule:	0 5 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance']/instance[databasetype='sqlserver'

dbWatch engine alert

Name:	dbWatch engine alert
Platform:	Sqlserver
Category:	Internal Internal
Description:	This procedure gathers statistics per database.
Long description:	This alert returns warning/alarm if there are several dbWatch engines installed on the same instance.
Version:	1
Default schedule:	90m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance']/instance[databasetype='sqlserver'&maj_version > '2000' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

dbWatch Server activity alert

Name:	dbWatch Server activity alert
Platform:	Sqlserver
Category:	Internal Internal
Description:	This procedure gathers statistics per database.
Long description:	This alert returns warning/alarm if the dbWatch Server is using consumes a lot of cpu over a period of time.
Version:	1.6
Default schedule:	2m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Autogrow settings

Name:	Autogrow settings
Platform:	Sqlserver
Category:	Maintenance
Description:	Checks database files auto-growth settings.
Long description:	The default auto-grow settings associated with the model database are not the best settings for how databases grows. If you haven't been diligent at setting the auto-grow parameters when you created databases then you might want scan your instance to determine which databases are using the default setting. We have split the possible file sizes and growth settings into 4 renges. The first range, which we do not analyze, are files smaller than 'minimum file size STEP 1'. The second rage, files between 'minimum file size STEP 1' and 'minumum file size STEP 2', are verified to have file growth settings equal 'minimum growth size SPEP 1'. The third ragen, files between 'minimum file size STEP 2' and 'minumum file size STEP 3', are verified to have file growth settings equal 'minimum growth size STEP 2'. The last rage verified that all files (bigger than 'minimum file size STEP 3') have minimum growth settings equal or higher than 'minimum growth size STEP 3' and at the same time lower than 'maximum growth size STEP 3'. Only ONLINE databases with the following properties are included: UserAccess = MULTI_USER, Updateability = READ_WRITE, IsInStandBy = 0, IsMergePublished = 0, IsPublished = 0 and IsSubscribed = 0.
Version:	2.1
Default schedule:	15 15 1 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Framework

Name:	Framework
Platform:	Sqlserver
Category:	Maintenance
Description:	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Long description:	
Version:	2.6
Default schedule:	50 5 1 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Framework (MS2000)

Name:	Framework
Platform:	Sqlserver
Category:	Maintenance
Description:	dbWatch engine framework task (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Long description:	
Version:	1.7
Default schedule:	50 5 1 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES'

Backup All databases

Name:	Backup All databases
Platform:	Sqlserver
Category:	Maintenance
Description:	Takes backup of all application and system databases.
Long description:	Takes backup of all application and system databases.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Backup All transaction logs

Name:	Backup All transaction logs
Platform:	Sqlserver
Category:	Maintenance
Description:	Takes backup of all transaction logs for databases running in FULL recovery mode.
Long description:	Takes backup of all transaction logs for databases running in FULL recovery mode.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Check database and server principal mapping

Name:	Check database and server principal mapping
Platform:	Sqlserver
Category:	Maintenance
Description:	Checks if the database owner (dbo) is maped into any Server Login (server principal).
Long description:	The Alert also collects statistics for all database users (in all none system databases) including Server login information. Users with principal_id 2, 3 and 4 will not be checked (guest, INFORMATION_SCHEMA and sys).
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Cleanup MSDB history tables

Name:	Cleanup MSDB history tables
Platform:	Sqlserver
Category:	Maintenance
Description:	This task deletes entries in the MSDB database history tables which holds statistics of backup/restore, jobs and maintenance plan executions.
Long description:	This task reduces the size of the history tables which holds statistics of backup/restore, jobs and maintenance plan executions by deleting the entries older than the specified date.
Version:	1.5
Default schedule:	35 1 1 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Cycle error log

Name:	Cycle error log
Platform:	Sqlserver
Category:	Maintenance
Description:	This task cycle MS SQL Server error log and Agent error log files.
Long description:	By default, there are seven SQL Server error logs – Errorlog and Errorlog.1 through Errorlog.6. The name of the current, most recent log is Errorlog with no extension. The log is re-created every time that you restart SQL Server. This task cycle error log and Agent error log files.
Version:	1.1
Default schedule:	15 0 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

DBCC CHECKDB

Name:	DBCC CHECKDB
Platform:	Sqlserver
Category:	Maintenance
Description:	Checks the logical and physical integrity of all the objects in all databases by performing the DBCC CHECKDB operation.
Long description:	By default only ONLINE databases with the following properties are included: UserAccess = MULTI_USER, Updateability = READ_WRITE, IsInStandBy = 0, IsMergePublished = 0, IsPublished = 0 and IsSubscribed = 0.
Version:	2.54
Default schedule:	15 4 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

DBCC UPDATEUSAGE

Name:	DBCC UPDATEUSAGE
Platform:	Sqlserver
Category:	Maintenance
Description:	This task is performing the DBCC UPDATEUSAGE operation to corrects pages and row count inaccuracies in the catalog views. These inaccuracies may cause incorrect space usage reports returned by the sp_spaceused system stored procedure.
Long description:	Do not run DBCC UPDATEUSAGE routinely unless you suspect incorrect values are being returned by sp_spaceused or when geting error number 2508 from DBCC CHECKDB procedure.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

External fragmentation (all databases)

Name:	External fragmentation (all databases)
Platform:	Sqlserver
Category:	Maintenance
Description:	External (logical) fragmentation occurs when an index leaf page is not in logical order. It occurs when the logical ordering of the index does not match the physical ordering of the index. This causes SQL Server to perform extra work to return ordered results.
Long description:	Collects external fragmentation statistics for tables and indexes. The information is extracted from the dynamic management function (view) sys.dm_db_index_physical_stats.
Version:	1.5
Default schedule:	20 6 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Internal fragmentation check

Name:	Internal fragmentation check
Platform:	Sqlserver
Category:	Maintenance
Description:	Checks the internal fragmentation for tables and indexes in all databases. The information is extracted from the dynamic management function (view) sys.dm_db_index_physical_stats.
Long description:	Internal fragmentation occurs when there is too much free space in the index or table pages. Typically, some free space is desirable, especially when the index is created or rebuilt. You can specify the Fill Factor setting when the index is created or rebuilt to indicate a percentage of how full the index pages are when created. If the pages are too fragmented, it will cause queries to take longer (because of the extra reads required to find the data set) and cause your indexes to grow larger than necessary. If no space is available in the index data pages, data changes (primarily inserts) will cause page splits, which also require additional system resources to perform.
Version:	2
Default schedule:	30 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

SQL Server performance counters

Name:	SQL Server performance counters
Platform:	Sqlserver
Category:	Maintenance
Description:	Checks if SQL Server performance counters are missing.
Long description:	
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & engine_edition = 'Microsoft SQL Server' & eng_inst_priv = 0

Rebuild indexes

Name:	Rebuild indexes
Platform:	Sqlserver
Category:	Maintenance
Description:	Rebuilds fragmented indexes in all databases.
Long description:	The main effect of fragmentation is that it slows down page read-ahead throughput during index scans, which causes slower response times. If the query workload on a fragmented table or index involve scans, removing fragmentation will have effect.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Rebuild indexes in table

Name:	Rebuld indexes in table
Platform:	Sqlserver
Category:	Maintenance
Description:	Rebuilds fragmented indexes in all tables listed in the ‘table list’ parameter.
Long description:	Rebuilds fragmented indexes in all tables listed in ‘table list’ parameter.
Version:	1.3
Default schedule:	5 22 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & eng_inst_priv = 0 & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Reorganize indexes

Name:	Reorganize indexes
Platform:	Sqlserver
Category:	Maintenance
Description:	Reorganizes fragmented indexes in all databases.
Long description:	Reorganizing an index uses minimal system resources. It defragments the leaf level of clustered and nonclustered indexes on tables and views by physically reordering the leaf-level pages to match the logical, left to right, order of the leaf nodes. Reorganizing also compacts the index pages. Compaction is based on the existing fill factor value.
Version:	2.6
Default schedule:	15 22 4 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Reorganize indexes in table

Name:	Reorganize indexes in table
Platform:	Sqlserver
Category:	Maintenance
Description:	Reorganizing fragmented indexes in all tables listed in the ‘table list’ parameter.
Long description:	Reorganizing an index uses minimal system resources. It defragments the leaf level of clustered and nonclustered indexes on tables and views by physically reordering the leaf-level pages to match the logical, left to right, order of the leaf nodes. Reorganizing also compacts the index pages. Compaction is based on the existing fill factor value.
Version:	1.2
Default schedule:	5 21 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & eng_inst_priv = 0 & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Shrink transaction logs

Name:	Shrink transaction logs
Platform:	Sqlserver
Category:	Maintenance
Description:	This procedure shrinks transaction log files which are detected by ‘Transaction log size check’ alert which cheks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size).
Long description:	This procedure shrinks transaction log files which are detected by ‘Transaction log size check’ alert which cheks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). Only transaction logs with one file can be shrinked.
Version:	1.2
Default schedule:	10 5 6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & eng_inst_priv = 0 & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Suspect pages

Name:	Suspect pages
Platform:	Sqlserver
Category:	Maintenance
Description:	Monitors suspect pages statistics in suspect_pages table.
Long description:	This alert monitors suspect pages by parsing statistics from suspect_pages table.
Version:	1.2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Update index statistics

Name:	Update index statistics
Platform:	Sqlserver
Category:	Maintenance
Description:	Update statistics in all (non-system) databases.
Long description:	Runs UPDATE STATISTICS against all user-defined and internal tables in all (non-system) database. By default, the query optimizer already updates statistics as necessary to improve the query plan. In some cases you can improve query performance by updating statistics more frequently than the default updates.
Version:	1.7
Default schedule:	15 3 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Update statistics

Name:	Update statistics
Platform:	Sqlserver
Category:	Maintenance
Description:	Update statistics in all (non system) databases.
Long description:	
Version:	1.2
Default schedule:	15 22 3 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2008' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Statistics migation

Name:	Statistics migration
Platform:	Sqlserver
Category:	Compliance
Description:	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
Long description:	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
Version:	1.0
Default schedule:	1 1 1 1
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Blocking detector

Name:	Blocking detector
Platform:	Sqlserver
Category:	Performance
Description:	Checks whether there exists blocked session.
Long description:	
Version:	1.2
Default schedule:	0,5,10,15,20,25,30,35,40,45,50,55 7-17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version < '2012' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Blocking statistics

Name:	Blocking statistics
Platform:	Sqlserver
Category:	Performance
Description:	Checks whether there exists any blocked sessions.
Long description:	
Version:	1.7
Default schedule:	0,5,10,15,20,25,30,35,40,45,50,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Cached query statistics

Name:	Cached query statistics
Platform:	Sqlserver
Category:	Performance
Description:	This task returns aggregate performance statistics based on cached query plans in SQL Server.
Long description:	This task returns aggregate performance statistics based on cached query plans in SQL Server (the sys.dm_exec_query_stats dynamic management view). The view contains one row per query statement within the cached plan, and the lifetime of the rows are tied to the plan itself. When a plan is removed from the cache, the corresponding rows are eliminated from this view which might produce inaccurate results if there are many plans removed between sampling of statistics by the dbWatch task.
Version:	1.1
Default schedule:	16 2,5,8,11,14,17,20,23 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & maj_version < '2017' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Data cache memory usage

Name:	Data cache memory usage
Platform:	Sqlserver
Category:	Performance
Description:	Collects data cache memory usage per database (for top 10 databases).
Long description:	Collects data cache memory usage per database (for top 10 databases).
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2008' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Data cache memory usage (MS2005 / MS2008)

Name:	Data cache memory usage
Platform:	Sqlserver
Category:	Performance
Description:	Collects data cache memory usage per database (for top 10 databases).
Long description:	
Version:	1.3
Default schedule:	20,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & maj_version < '2012' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Data hit ratio

Name:	Data hit ratio
Platform:	Sqlserver
Category:	Performance
Description:	Monitors the buffer cache hit ratio by extracting counter values from the master.dbo.sysperfinfo table for the counters ‘Buffer cache hit ratio’ and ‘Buffer cache hit ratio base’.
Long description:	Monitors the buffer cache hit ratio by extracting counter values from the master.dbo.sysperfinfo table for the counters ‘Buffer cache hit ratio’ and ‘Buffer cache hit ratio base’.
Version:	1.8
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘1999’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Database session load

Name:	Database session load
Platform:	Sqlserver
Category:	Performance
Description:	Shows the number of connections over time per database, host and application.
Long description:	Each time the procedure is executed, statistics are collected on the number of connections per database, host, program and login. Then (once a day) the maximum connection and avarage connections (per database, host,
Version:	1.1
Default schedule:	0,20,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' or engine_edition = 'Azure SQL Managed Instance')

Test DML-DDL performance

Name:	Test DML-DDL performance
Platform:	Sqlserver
Category:	Performance
Description:	Runs performance test on the database. The procedure executes SELECT, INSERT, UPDATE, DELETE (and TRUNCATE) statements on the test table.
Long description:	Runs performance test on the database. The procedure executes SELECT, INSERT, UPDATE, DELETE (and TRUNCATE) statements on the test table.
Version:	1.7
Default schedule:	10 22 5 1,30
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

File IO statistics

Name:	File IO statistics
Platform:	Sqlserver
Category:	Performance
Description:	Collects I/O statistics for data and log files.
Long description:	Collects I/O statistics for data and log files baseed on dynamic management view sys.dm_io_virtual_file_stats.
Version:	1.6
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

High activity monitor

Name:	High activity monitor
Platform:	Sqlserver
Category:	Performance
Description:	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
Long description:	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance. Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

High activity monitor (MS2005)

Name:	High activity monitor
Platform:	Sqlserver
Category:	Performance
Description:	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
Long description:	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2005' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Index usage statistics (all databases)

Name:	Index usage statistics (all databases)
Platform:	Sqlserver
Category:	Performance
Description:	This procedure collects statistics from sys.dm_db_index_usage_stats performance view which gives information on how an index (or a table – heap) has been used to resolve queries.
Long description:	The procedure checks objects for all databases.
Version:	1.72
Default schedule:	12 1,4,7,10,13,16,19,22 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & eng_inst_priv = 0 & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Instance memory check

Name:	Instance memory check
Platform:	Sqlserver
Category:	Performance
Description:	This job checks the target memory value of the SQL Server instance, and gives a warning/ alarm if the instance is not able to allocate a
Long description:	This job checks the target memory value of the SQL Server instance, and gives a warning/ alarm if the instance is not able to allocate a
Version:	1.3
Default schedule:	3,23,43 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2008' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Instance memory usage

Name:	Instance memory usage
Platform:	Sqlserver
Category:	Performance
Description:	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Long description:	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2008' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Instance memory usage (MS2005 / MS2008)

Name:	Instance memory usage
Platform:	Sqlserver
Category:	Performance
Description:	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Long description:	
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & maj_version < '2012' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Lazy writer and checkpoint statistics

Name:	Lazy writer and Checkpoint statistics
Platform:	Sqlserver
Category:	Performance
Description:	Monitors ‘Checkpoint pages/sec’, ‘Lazy writes/sec’ and ‘Page life expectancy’ by extracting counter values from the sys.dm_os_performance_counters performace table.
Long description:	Lazy writer purpose is to release the buffer pool memory. When more memory is needed, lazy writer responds to a memory pressure releasing the “coldest” pages from the buffer pool, and makes more memory available for new pages to come in.
Version:	1.3
Default schedule:	4,14,24,34,44,54 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sqlserver’]/instance[maj_version > ‘2005’ & hasengine=‘YES’ & (engine_edition = ‘Microsoft SQL Server’ | engine_edition = ‘Azure SQL Managed Instance’)

Memory objects statistics

Name:	Memory objects statistics
Platform:	Sqlserver
Category:	Performance
Description:	This task collects the size of memory objects that are currently allocated by the SQL Server
Long description:	This task collects the size of memory objects that are currently allocated by the SQL Server, and is primarily used to analyze memory usage and to identify possible memory leaks.
Version:	1.3
Default schedule:	7,27,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & maj_version < '2012' & hasengine='YES' & engine_edition = 'Microsoft SQL Server'

Blocking detector (noschema)

Name:	Blocking detector
Platform:	Sqlserver
Category:	Performance
Description:	This task collects the size of memory objects that are currently allocated by the SQL Server
Long description:	Checks whether there exists any blocked sessions.
Version:	2
Default schedule:	2m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Instance memory check (noschema)

Name:	Instance memory check
Platform:	Sqlserver
Category:	Performance
Description:	This task collects the size of memory objects that are currently allocated by the SQL Server
Long description:	The alert checks the target memory value of the SQL Server instance, and gives a warning/ alarm if the instance is not able to allocate a certain percentage of the total server/machine memory.
Version:	2
Default schedule:	3,23,43 * * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[engine_edition = 'Microsoft SQL Server'

Server memory statistics

Name:	Server memory statistics
Platform:	Sqlserver
Category:	Performance
Description:	This tasks collects statistics from the sys.dm_os_sys_info performance view about the memory resources available to and consumed by the SQL Server.
Long description:	This tasks collects statistics from the sys.dm_os_sys_info performance view about the memory resources available to and consumed by the SQL Server.
Version:	1.1
Default schedule:	28,58 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & maj_version < '2012' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Sessions per database

Name:	Sessions per database
Platform:	Sqlserver
Category:	Performance
Description:	This task returns aggregate performance statistics based on sessions connected to the SQL Server instance.
Long description:	This task returns aggregate performance statistics based on sys.dm_exec_sessions dynamic management view and sysprocesses view. The views contains one row per connection, and the lifetime of the rows are tied to connection itself. When a session is disconnected/ terminated, the corresponding row are eliminated from this views which might produce inaccurate results if there are sessions disconnected between sampling of statistics by the dbWatch task.
Version:	1.4
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Session load

Name:	Database session load
Platform:	Sqlserver
Category:	Performance
Description:	Shows the number of connections over time per database, host and application.
Long description:	Each time the procedure is executed, statistics are collected on the number of connections per database, host, program and login. Then (once a day) the maximum connection and avarage connections (per database, host,
Version:	1.1
Default schedule:	0,20,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Transactions log flushed bytes load

Name:	Transactions log flushed bytes load
Platform:	Sqlserver
Category:	Performance
Description:	Shows bytes flushed to transaction logs over time, total and top 5 databases.
Long description:	Shows bytes flushed to transaction logs over time, total and top 5 databases. Shows bytes flushed to transaction logs over time, total and top 5 databases.
Version:	1.3
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Transactions log flushed bytes load (MS2000)

Name:	Transactions log flushed bytes load
Platform:	Sqlserver
Category:	Performance
Description:	Shows bytes flushed to transaction logs over time, total and top 5 databases.
Long description:	Shows bytes flushed to transaction logs over time, total and top 5 databases. Shows bytes flushed to transaction logs over time, total and top 5 databases.
Version:	1.3
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version = '2000' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Transactions load

Name:	Transactions load
Platform:	Sqlserver
Category:	Performance
Description:	Shows the transactions load over time, total and top 5 databases.
Long description:	Shows the transactions load over time, total and top 5 databases.
Version:	1.2
Default schedule:	1,11,21,31,41,51 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '1999' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Wait statistics

Name:	Wait statistics
Platform:	Sqlserver
Category:	Performance
Description:	Collects statistics about all waits encountered by threads that executed. This task is based on the sys.dm_os_wait_stats dynamic performance view.
Long description:	Collects statistics about all waits encountered by threads that executed. This task is based on the sys.dm_os_wait_stats dynamic performance view.
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2005' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

SQL statistics

Name:	SQL statistics
Platform:	Sqlserver
Category:	Performance
Description:	Collects performance statistics for the SQL statements.
Long description:	Collects performance statistics for cached query plans from system dynamic management views: sys.dm_exec_query_stats, sys.dm_exec_sql_text and sys.dm_exec_text_query_plan
Version:	2
Default schedule:	0 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2012' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Query store status

Name:	Query Store status
Platform:	Sqlserver
Category:	Capacity
Description:	Checks Query Store space usage in every database where it is enabled.
Long description:	Checks Query Store space usage in every database where it is enabled by analyzing statistic from sys.database query store_options system catalog view.
Version:	1.1
Default schedule:	30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sqlserver']/instance[maj_version > '2014' & hasengine='YES' & (engine_edition = 'Microsoft SQL Server' | engine_edition = 'Azure SQL Managed Instance')

Jobs for PostgreSQL

Control Center Jobs	Description
Availability	
Backup check – pg_dump	Checks that there exists an up to date backup file.
Backup check – WAL	Checks that the WAL files have been backed up.
DBMS uptime	Collects database uptime statistics.
dbWatch engine alert	Collects database uptime statistics.
Capacity	
Log size statistics	Checks the size of the log file.
Schema growth collector	Checks the size of the log file.
Vaccum alert	Checks the size of the log file.
Schema growth and information	Collects detailed schema growth rate information.
Schema growth statistics	Collects and analyzes schema growth statistics from data collected by
Tablespace growth and information	Shows an overview of tablespace growth and statistics.
Cluster	
BDR Replication lag	Checks for BDR replication lag
Replication delay alert	Checks for BDR replication lag
Replication slave client alert	Checks for BDR replication lag
Replication slave client alert	Checks for BDR replication lag
Maintenance	
Backup job – pg_dump_all	Runs a backup of all the postgres databases on an instance using pg_dumpall through an ssh connection.
framework	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Analyze tables	dbWatch engine framework job (for internal use only). Used for patching or

	upgrading of dbWatch engine framework.
Vaccum tables	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Performance	
Buffer cache reads per database	Collects the number of block fetch requests for table or index per database.
Buffer cache statistics	Gets data cache statistics from pg_stat_database view.
Disk block hitrate	Checks the hitrate for disk block requests.
Disk block reads per database	Collects the number of disk block fetch requests for table or index per database.
Test DML performance	Runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on schema test tables.
Index block hitrate	Checks the hitrate for index disk block requests.
Locks held and statistics	Shows locks held statistics.
Table and index scan collector	Shows locks held statistics.
Session load	Gathers session load statistics.
Table and index scan statistics	This task gathers scan statistics on tables and indexes from pg_stat_all_tables view.
Transaction statistics	Gathers information on commits and rollbacks in the databases and generates statistics.
Performance ***optional***	
SQL statistics *for 13 and 14*	Collects performance statistics for the SQL statements.
SQL statistics *for 9.4 to 12*	Collects performance statistics for the SQL statements.

Backup check – pg_dump

Name:	Backup check – pg_dump
Platform:	Postgres
Category:	Availability
Description:	Checks that there exists an up to date backup file.
Long description:	Checks that there exists an up to date backup file.
Version:	1.6
Default schedule:	10 06 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0' & newer_than_ninefour = 1

Backup check – WAL

Name:	Backup check – WAL
Platform:	Postgres
Category:	Availability
Description:	Checks that the WAL files have been backed up.
Long description:	Checks that the WAL files have been backed up.
Version:	1.5
Default schedule:	19 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0' & maj_version > '11'

DBMS uptime

Name:	DBMS uptime
Platform:	Postgres
Category:	Availability
Description:	Collects database uptime statistics.
Long description:	Collects database uptime statistics.
Version:	1.4
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninefour = 1

dbWatch engine alert

Name:	dbWatch engine alert
Platform:	Postgres
Category:	Internal
Description:	Collects database uptime statistics.
Long description:	
Version:	1
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	

Log size statistics

Name:	Log size statistics
Platform:	Postgres
Category:	Capacity
Description:	Checks the size of the log file.
Long description:	Checks the size of the log file.
Version:	1.2
Default schedule:	19 1,7,13,19 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0'

Schema growth collector

Name:	Schema growth collector
Platform:	Postgres
Category:	Capacity
Description:	Checks the size of the log file.
Long description:	This job collects growth rate statistics for all schemas i all databases.
Version:	1.3
Default schedule:	10 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Vacuum alert

Name:	Vacuum alert
Platform:	Postgres
Category:	Capacity
Description:	Checks the size of the log file.
Long description:	This job hecks if vacuum has been performed recently.
Version:	2.6
Default schedule:	10 7 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[newer_than_ninetwo = '1'

Schema growth and information

Name:	Schema growth and information
Platform:	Postgres
Category:	Capacity
Description:	Collects detailed schema growth rate information.
Long description:	Collects detailed schema growth rate information.
Version:	1.7
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Schema growth statistics

Name:	Schema growth statistics
Platform:	Postgres
Category:	Capacity
Description:	Collects and analyzes schema growth statistics from data collected by Schema growth collector job
Long description:	Collects and analyzes schema growth statistics from data collected by Schema growth collector job
Version:	1.7
Default schedule:	30 6 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Tablespace growth and information

Name:	Tablespace growth and information
Platform:	Postgres
Category:	Capacity
Description:	Shows an overview of tablespace growth and statistics.
Long description:	Shows an overview of tablespace growth and statistics.
Version:	1.4
Default schedule:	10 18 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES'

BDR Replication lag

Name:	BDR Replication lag
Platform:	Postgres
Category:	Cluster
Description:	Checks for BDR replication lag
Long description:	Task analyses BDR replication lag
Version:	0.3
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & replication_used > 0

Replication delay alert

Name:	Replication delay alert
Platform:	Postgres
Category:	Cluster
Description:	Checks for BDR replication lag
Long description:	
Version:	1.01
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[newer_than_ninetwo = '1'

Replication slave client alert

Name:	Replication slave client alert
Platform:	Postgres
Category:	Cluster
Description:	Checks for BDR replication lag
Long description:	
Version:	1
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[newer_than_ninetwo = '1' & maj_version ='9'

Backup job – pg_dump_all (ssh)

Name:	Backup job – pg_dump_all (ssh)
Platform:	Postgres
Category:	Maintenance
Description:	Runs a backup of all the postgres databases on an instance using pg_dumpall through an ssh connection.
Long description:	Runs a backup of all the postgres databases on an instance using pg_dumpall through an ssh connection.
Version:	0.4
Default schedule:	10 18 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Framework

Name:	Framework
Platform:	Postgres
Category:	Maintenance
Description:	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Long description:	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Version:	1.1
Default schedule:	40 5 1 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Analyze tables

Name:	Analyze tables
Platform:	Postgres
Category:	Maintenance
Description:	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Long description:	This job runs ANALYZE statement on every database to collect statistics about the contents of tables.
Version:	2.0
Default schedule:	40 6 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[newer_than_ninetwo = '1'

Vacuum tables

Name:	Vaccum tables
Platform:	Postgres
Category:	Maintenance
Description:	This job runs VACUUM statement on every database and reclaims storage occupied by dead tuples. It’s necessary to do VACUUM periodically, especially on frequently-updated tables.
Long description:	This job runs VACUUM statement on every database and reclaims storage occupied by dead tuples. It’s necessary to do VACUUM periodically, especially on frequently-updated tables.
Version:	2.0
Default schedule:	10 6 * *
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘postgres’]/.[newer_than_ninetwo = ‘1’

Buffer cache reads per database

Name:	Buffer cache reads per database
Platform:	Postgres
Category:	Performance
Description:	Collects the number of block fetch requests for table or index per database.
Long description:	Collects the number of block fetch requests and hits for table or index using the pg_stat_get_blocks_fetched and pg_stat_get_db_blocks_hit functions. The statistics are collected per database.
Version:	1.3
Default schedule:	1,11,21,31,41,51 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0' & newer_than_ninefour = 1

Buffer cache statistics

Name:	Buffer cache statistics
Platform:	Postgres
Category:	Performance
Description:	Gets data cache statistics from pg_stat_database view.
Long description:	Task gets data cache statistic from pg_stat_database view (columns: blks_read and blks_hit).
Version:	1.3
Default schedule:	4,14,24,34,44,54 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0' & newer_than_ninefour = 1

Disk block hitrate

Name:	Disk block hitrate
Platform:	Postgres
Category:	Performance
Description:	Checks the hitrate for disk block requests.
Long description:	Task checks the hitrate for disk block requests.
Version:	1.4
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES'

Disk block reads per database

Name:	Disk block reads per database
Platform:	Postgres
Category:	Performance
Description:	Collects the number of disk block fetch requests for table or index per database.
Long description:	Collects the number of disk block fetch requests for table or index using the pg_stat_get_blocks_fetched and pg_stat_get_db_blocks_hit functions. The statistics are collected per database.
Version:	1.3
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0' & newer_than_ninefour = 1

Test DML performance

Name:	Test DML performance
Platform:	Postgres
Category:	Performance
Description:	Runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on schema test tables.
Long description:	Task runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on the dbWatch engine schema test tables.
Version:	1.2
Default schedule:	10 22 5 1,30
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Index block hitrate

Name:	Index block hitrate
Platform:	Postgres
Category:	Performance
Description:	Checks the hitrate for index disk block requests.
Long description:	Task checks the hitrate for index disk block requests.
Version:	1.5
Default schedule:	7,37 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES'

Locks held and statistics

Name:	Locks held and statistics
Platform:	Postgres
Category:	Performance
Description:	Shows locks held statistics.
Long description:	Task shows locks held statistics.
Version:	1.6
Default schedule:	2,17,32,47 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & pg_stat_activity_type3 = '1'

Table and index scan collector

Name:	Table and index scan collector
Platform:	Postgres
Category:	Performance
Description:	Shows locks held statistics.
Long description:	This job collects table and index scan statistics for all schemas i all databases.
Version:	1.31
Default schedule:	10,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Session load

Name:	Session load
Platform:	Postgres
Category:	Performance
Description:	Gathers session load statistics.
Long description:	Task gathers session load statistics.
Version:	1.7
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & is_awsrds = '0' & newer_than_ninefour = 1

Table and index scan statistics

Name:	Table and index scan statistics
Platform:	Postgres
Category:	Performance
Description:	This task gathers scan statistics on tables and indexes from pg_stat_all_tables view.
Long description:	Statistics from pg_stat_all_tables view are gathered for the following columns:
Version:	1.8
Default schedule:	15,45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninetwo = '1'

Transaction statistics

Name:	Transaction statistics
Platform:	Postgres
Category:	Performance
Description:	Gathers information on commits and rollbacks in the databases and generates statistics.
Long description:	Task gathers information about commits and rollbacks in the database and generates statistics.
Version:	1.8
Default schedule:	0,20,40 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES'

SQL statistics

Name:	SQL statistics
Platform:	Postgres
Category:	Performance
Description:	Collects performance statistics for the SQL statements.
Long description:	Collects performance statistics for SQL statements from pg_stat_statements view. During installation it checks if the pg_stat_statements extention is created, and create it if it is not. For tracking which queries get executed in your database you need to add the following entries to your postgresql.conf file:
Version:	1.1
Default schedule:	0,5,10,15,20,25,30,35,40,45,50,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & maj_version > 12 & maj_version < 15

SQL statistics

Name:	SQL statistics
Platform:	Postgres
Category:	Performance
Description:	Collects performance statistics for the SQL statements.
Long description:	Collects performance statistics for SQL statements from pg_stat_statements view. During installation it checks if the pg_stat_statements extention is created, and create it if it is not. For tracking which queries get executed in your database you need to add the following entries to your postgresql.conf file:
Version:	1.1
Default schedule:	0,5,10,15,20,25,30,35,40,45,50,55 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='postgres']/.[hasengine='YES' & newer_than_ninefour = 1 & maj_version < 13

Jobs for MySQL

Control Center Jobs	Description
Availability	
DBMS uptime	Collects uptime statistics in the database.
MySQL aborted connects	Checking if there is many aborted connects
MySQL max connections	Checking if the system is near maximum connections
MySQL mysqld process	Checking if mysqld process is running
Capacity	
Database growth rate	Collects database size and visualizes the growth rate of the server.
Database growth rate	Collects detailed growth rate information.
Cluster	
NDB data memory usage check	NDB data memory usage check
NDB data node status	NDB data node status
Innodb cluster status	This job checks state of all members in Innodb cluster
Innodb cluster switch	This job checks if Innodb cluster PRIMARY/SECONDARY switch occurs.
Replica count	This job checks all of replicas currently registered with the source.
Replica delay	This job checks how “late” the replica is, and measures the time difference in seconds between the replication SQL (applier) thread and the replication I/O (receiver) thread.
Replica state	This job checks if replica changes occurs. For example, if an instance becomes a replica or if an instance is no longer a replica.
Performance	
Binlog cache check	Binlog cache performance check
Innodb buffer	Checks the hit rate of the innodb buffer pool

pool check	
Key buffer check	Checks key buffer efficiency
Database load	Provides information on the server load
Lock statistics	Collects lock statistics
Memory setup	Analyzes the memory setup of the server
Network traffic	Shows the network traffic
Query cache hitrate	Shows the query cache hit rate
Session load	Shows connection statistics
Thread cache hitrate	Shows the thread cache hit rate
Temporary table check	Shows percentage of temporary tables written to disk

DBMS uptime

Name:	DBMS uptime
Platform:	Mysql
Category:	Availability
Description:	Collects uptime statistics in the database.
Long description:	Task collecs uptime data statistics in the database server.
Version:	2.3
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

MySQL aborted connects

Name:	Aborted connections
Platform:	Mysql
Category:	Availability
Description:	Checks the number of aborted client connections.
Long description:	Checks the number of aborted client connections. Usually this is because of incorrect password or no matching host for the user. Aborted connections can also be caused by clients trying to connect with invalid or malformed connection strings.
Version:	1.2
Default schedule:	9,19,29,39,49,59 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & use_global_variables_performance_schema = '1'

Parameters		
Name	Default value	Description
warning threshold	20	The maximum value of aborted connects (over a period of time defined by the
alarm threshold	100	The maximum value of aborted connects (over a period of time defined by the
threshold (time)	60	A period of time (in minutes) which must be passed (combined with the number of aborted connects) before an alarm or a warning is returned by the procedure.
enable warnings and alarms	NO	If set to "NO" (default), the alert will only collect statistics without returning status warning or alarm. Value "YES" will activate the alert.
history threshold	7	The maximum number of day to kept statistics for in the historic tables.

<< DBMS uptime / MySQL max connections >>

MySQL max connections

Name:	MySQL max connections
Platform:	Mysql
Category:	
Description:	Checking if the system is near maximum connections
Long description:	
Version:	1
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']]]

MySQL mysqld process

Name:	MySQL mysqld process
Platform:	Mysql
Category:	
Description:	Checking if mysqld process is running
Long description:	
Version:	1
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']]]

Database growth rate (aggregated)

Name:	Database growth rate (aggregated)
Platform:	Mysql
Category:	Capacity
Description:	Collects database size and visualizes the growth rate of the server.
Long description:	Collects database size and visualizes the growth rate of the database server (aggregated).
Version:	2.2
Default schedule:	15 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES']]]

Database growth rate (detailed)

Name:	Database growth rate (detailed)
Platform:	Mysql
Category:	Capacity
Description:	Collects detailed growth rate information.
Long description:	Collects detailed information on the growth rate of the database server (detailed).
Version:	2.1
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES']]]

Innodb buffer pool check

Name:	Innodb buffer pool check
Platform:	Mysql
Category:	Performance
Description:	Checks the hit rate of the innodb buffer pool
Long description:	This checks the hit rate of the innodb buffer pool
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Innodb cluster status

Name:	Innodb cluster status
Platform:	Mysql
Category:	Cluster and Replication
Description:	This job checks state of all members in Innodb cluster
Long description:	
Version:	1.2
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']]]

NDB data memory usage check

Name:	NDB data memory usage check
Platform:	Mysql
Category:	Cluster and Replication
Description:	NDB data memory usage check
Long description:	This checks the usage of data memory in NDB cluster database
Version:	0.1
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

NDB data node status

Name:	NDB data node status
Platform:	Mysql
Category:	Cluster and Replication
Description:	NDB data node status
Long description:	This checks the status of data nodes in NDB cluster database
Version:	0.1
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Innodb cluster switch

Name:	Innodb cluster switch
Platform:	Mysql
Category:	Cluster and Replication
Description:	This job checks if Innodb cluster PRIMARY/SECONDARY switch occurs.
Long description:	
Version:	1.7
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']]]

Replica count

Name:	Replica count
Platform:	Mysql
Category:	Cluster and Replication
Description:	This job checks all of replicas currently registered with the source.
Long description:	
Version:	1.5
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='mysql']/instance[is_mysql_branch='1']]]

Replica delay

Name:	Replica delay
Platform:	Mysql
Category:	Cluster and Replication
Description:	This job checks how “late” the replica is, and measures the time difference in seconds between the replication SQL (applier) thread and the replication I/O (receiver) thread.
Long description:	
Version:	1.5
Default schedule:	30s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘mysql’]/instance[is_mysql_branch=‘1’]]]

Replica state

Name:	Replica state
Platform:	Mysql
Category:	Cluster and Replication
Description:	This job checks if replica changes occurs. For example, if an instance becomes a replica or if an instance is no longer a replica.
Long description:	
Version:	1.2
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[./instance[databasetype='mysql']]]

Binlog cache check

Name:	Binlog cache check
Platform:	Mysql
Category:	Performance
Description:	Binlog cache performance check
Long description:	This checks the performance of the ‘Binlog cache’
Version:	2.31
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & is_mysql_branch=‘1’]/.[hasengine=‘YES’ & global_status_performance_schema = ‘1’]]]

Key buffer check

Name:	Key buffer check
Platform:	Mysql
Category:	Performance
Description:	Checks key buffer efficiency
Long description:	Task checks key buffer efficiency
Version:	2.22
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Database load

Name:	Database load
Platform:	Mysql
Category:	Performance
Description:	Provides information on the server load
Long description:	Task provides infomation on the server load in the database server
Version:	2.23
Default schedule:	27,57 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Lock statistics

Name:	Lock statistics
Platform:	Mysql
Category:	Performance
Description:	Collects lock statistics
Long description:	This task collects lock data statistics
Version:	2.11
Default schedule:	55,25 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Memory setup

Name:	Memory setup
Platform:	Mysql
Category:	Performance
Description:	Analyzes the memory setup of the server
Long description:	Task analyzes memory setup of the server
Version:	2.11
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Network traffic

Name:	Network traffic
Platform:	Mysql
Category:	Performance
Description:	Shows the network traffic
Long description:	Task shows amount of network traffic in the server
Version:	2.2
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Query cache hitrate

Name:	Query cache hitrate
Platform:	Mysql
Category:	Performance
Description:	Shows the query cache hit rate
Long description:	Task shows the hit rate of query cache
Version:	2.42
Default schedule:	2,32 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1' & has_query_cache ='1']]]

Session load

Name:	Session load
Platform:	Mysql
Category:	Performance
Description:	Shows connection statistics
Long description:	Task shows connection data statistics
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Thread cache hitrate

Name:	Thread cache hitrate
Platform:	Mysql
Category:	Performance
Description:	Shows the thread cache hit rate
Long description:	Task shows the hit rate of the thread cache
Version:	2.41
Default schedule:	7,37 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Temporary table check

Name:	Temporary table check
Platform:	Mysql
Category:	Performance
Description:	Shows percentage of temporary tables written to disk
Long description:	Task shows the percentage of temporary tables which have been written to disk
Version:	2.31
Default schedule:	13,43 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mysql_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1']]]

Jobs for MariaDB

Control Center Jobs	Description
Availability	
DBMS uptime	Collects uptime statistics in the database.
Aborted connects alert	Collects uptime statistics in the database.
Maximum connections alert	Collects uptime statistics in the database.
MySQLd process alert	Collects uptime statistics in the database.
MariaDB Space Optimizer	Collects uptime statistics in the database.
Capacity	
Database growth rate	Collects database size and visualizes the growth rate of the server.
Database growth rate	Collects detailed growth rate information.
Cluster	
NDB data memory usage check	NDB data memory usage check
NDB data node status	NDB data node status
Replica count	NDB data node status
Replica delay	NDB data node status
Replica state	NDB data node status
Performance	
Binlog cache check	Binlog cache performance check
Innodb buffer pool check	Checks the hit rate of the innodb buffer pool
Key buffer check	Checks key buffer efficiency
Database load	Provides information on the server load
Lock statistics	Collects lock statistics
Memory setup	Analyzes the memory setup of the server
Network traffic	Shows the network traffic
Query cache hitrate	Shows the query cache hit rate
Session load	Shows connection statistics
Thread cache hitrate	Shows the thread cache hit rate
Temporary table check	Shows percentage of temporary tables written to disk
MariaDB Index Redundancy Checker	Shows percentage of temporary tables written to disk

DBMS uptime

Name:	DBMS uptime
Platform:	Mariadb
Category:	Availability
Description:	Collects uptime statistics in the database.
Long description:	Task collecs uptime data statistics in the database server.
Version:	2.3
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

DBMS uptime (old)

Name:	DBMS uptime
Platform:	Mariadb
Category:	Availability
Description:	Collects uptime statistics in the database.
Long description:	Task collecs uptime data statistics in the database server.
Version:	1.0
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Binlog cache check

Name:	Binlog cache check
Platform:	Mariadb
Category:	Performance
Description:	Binlog cache performance check
Long description:	This checks the performance of the ‘Binlog cache’
Version:	2.31
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & is_mariadb_branch=‘1’]/.[hasengine=‘YES’ & global_status_performance_schema = ‘1’

Binlog cache check (old)

Name:	Binlog cache check
Platform:	Mariadb
Category:	Performance
Description:	Binlog cache performance check
Long description:	This checks the performance of the ‘Binlog cache’
Version:	1.1
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & is_mariadb_branch=‘1’]/.[hasengine=‘YES’ & global_status_performance_schema = ‘0’ & global_status_information_schema = ‘1’

Database growth rate (aggregated)

Name:	Database growth rate (aggregated)
Platform:	Mariadb
Category:	Capacity
Description:	Collects database size and visualizes the growth rate of the server.
Long description:	Collects database size and visualizes the growth rate of the database server (aggregated).
Version:	2.2
Default schedule:	15 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES'

Database growth rate (detailed)

Name:	Database growth rate (detailed)
Platform:	Mariadb
Category:	Capacity
Description:	Collects detailed growth rate information.
Long description:	Collects detailed information on the growth rate of the database server (detailed).
Version:	2.1
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES'

MariaDB Index Redundancy Checker

Name:	MariaDB Index Redundancy Checker
Platform:	Mariadb
Category:	Performance
Description:	Shows percentage of temporary tables written to disk
Long description:	
Version:	1
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	—

Innodb buffer pool check

Name:	Innodb buffer pool check
Platform:	Mariadb
Category:	Performance
Description:	Checks the hit rate of the innodb buffer pool
Long description:	This checks the hit rate of the innodb buffer pool
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Innodb buffer pool check (old)

Name:	Innodb buffer pool check
Platform:	Mariadb
Category:	Performance
Description:	Checks the hit rate of the innodb buffer pool
Long description:	This checks the hit rate of the innodb buffer pool
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Key buffer check

Name:	Key buffer check
Platform:	Mariadb
Category:	Performance
Description:	Checks key buffer efficiency
Long description:	Task checks key buffer efficiency
Version:	2.22
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Key buffer check (old)

Name:	Key buffer check
Platform:	Mariadb
Category:	Performance
Description:	Checks key buffer efficiency
Long description:	Task checks key buffer efficiency
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Database load

Name:	Database load
Platform:	Mariadb
Category:	Performance
Description:	Provides information on the server load
Long description:	Task provides infomation on the server load in the database server
Version:	2.23
Default schedule:	27,57 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Database load (old)

Name:	Database load
Platform:	Mariadb
Category:	Performance
Description:	Provides information on the server load
Long description:	Task provides infomation on the server load in the database server
Version:	1.1
Default schedule:	27,57 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Lock statistics

Name:	Lock statistics
Platform:	Mariadb
Category:	Performance
Description:	Collects lock statistics
Long description:	This task collects lock data statistics
Version:	2.11
Default schedule:	55,25 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Lock statistics (old)

Name:	Lock statistics
Platform:	Mariadb
Category:	Performance
Description:	Collects lock statistics
Long description:	This task collects lock data statistics
Version:	2.11
Default schedule:	55,25 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Memory setup

Name:	Memory setup
Platform:	Mariadb
Category:	Performance
Description:	Analyzes the memory setup of the server
Long description:	Task analyzes memory setup of the server
Version:	2.11
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Memory setup (old)

Name:	Memory setup
Platform:	Mariadb
Category:	Performance
Description:	Analyzes the memory setup of the server
Long description:	Task analyzes memory setup of the server
Version:	2.11
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

NDB data memory usage check

Name:	NDB data memory usage check
Platform:	Mariadb
Category:	Cluster and Replication
Description:	NDB data memory usage check
Long description:	This checks the usage of data memory in NDB cluster database
Version:	0.1
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

NDB data node status

Name:	NDB data node status
Platform:	Mariadb
Category:	Cluster and Replication
Description:	NDB data node status
Long description:	This checks the status of data nodes in NDB cluster database
Version:	0.1
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Network traffic

Name:	Network traffic
Platform:	Mariadb
Category:	Performance
Description:	Shows the network traffic
Long description:	Task shows amount of network traffic in the server
Version:	2.2
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Network traffic (old)

Name:	Network traffic
Platform:	Mariadb
Category:	Performance
Description:	Shows the network traffic
Long description:	Task shows amount of network traffic in the server
Version:	2.2
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Aborted connects alert

Name:	Aborted connects alert
Platform:	Mariadb
Category:	Availability
Description:	Checking if there is many aborted connects
Long description:	
Version:	1
Default schedule:	1h
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']

Maximum connections alert

Name:	Maximum connections alert
Platform:	Mariadb
Category:	Availability
Description:	Checking if the system is near maximum connections
Long description:	
Version:	1
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']

MySQLd process alert

Name:	MySQLd process alert
Platform:	Mariadb
Category:	Availability
Description:	Checking if mysqld process is running
Long description:	
Version:	1
Default schedule:	15m
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']

Query cache hitrate

Name:	Query cache hitrate
Platform:	Mariadb
Category:	Performance
Description:	Shows the query cache hit rate
Long description:	Task shows the hit rate of query cache
Version:	2.42
Default schedule:	2,32 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1' & has_query_cache ='1'

Query cache hitrate (old)

Name:	Query cache hitrate
Platform:	Mariadb
Category:	Performance
Description:	Shows the query cache hit rate
Long description:	Task shows the hit rate of query cache
Version:	2.42
Default schedule:	2,32 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Replica count

Name:	Replica count
Platform:	Mariadb
Category:	Cluster and Replication
Description:	This job checks all of replicas currently registered with the source.
Long description:	
Version:	1.5
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type='instance' & databasetype='mariadb']/instance[is_mariadb_branch='1'

Replica delay

Name:	Replica delay
Platform:	Mariadb
Category:	Cluster and Replication
Description:	This job checks how “late” the replica is, and measures the time difference in seconds between the replication SQL (applier) thread and the replication I/O (receiver) thread.
Long description:	
Version:	1.5
Default schedule:	30s
Requires engine install:	No
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘mariadb’]/instance[is_mariadb_branch=‘1’

Replica state

Name:	Replica state
Platform:	Mariadb
Category:	Cluster and Replication
Description:	This job checks if replica changes occurs. For example, if an instance becomes a replica or if an instance is no longer a replica.
Long description:	
Version:	1.2
Default schedule:	60s
Requires engine install:	No
Compatibility tag:	[CDATA[./instance[databasetype='mariadb']

Session load

Name:	Session load
Platform:	Mariadb
Category:	Performance
Description:	Shows connection statistics
Long description:	Task shows connection data statistics
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Session load (old)

Name:	Session load
Platform:	Mariadb
Category:	Performance
Description:	Shows connection statistics
Long description:	Task shows connection data statistics
Version:	2.21
Default schedule:	0,30 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

MariaDB Space Optimizer

Name:	MariaDB Space Optimizer
Platform:	Mariadb
Category:	Availability
Description:	Tells the user how much space tables in a specific database consume.
Long description:	
Version:	1
Default schedule:	10m
Requires engine install:	No
Compatibility tag:	—

Thread cache hitrate

Name:	Thread cache hitrate
Platform:	Mariadb
Category:	Performance
Description:	Shows the thread cache hit rate
Long description:	Task shows the hit rate of the thread cache
Version:	2.41
Default schedule:	7,37 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Thread cache hitrate (old)

Name:	Thread cache hitrate
Platform:	Mariadb
Category:	Performance
Description:	Shows the thread cache hit rate
Long description:	Task shows the hit rate of the thread cache
Version:	2.41
Default schedule:	7,37 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Temporary table check

Name:	Temporary table check
Platform:	Mariadb
Category:	Performance
Description:	Shows percentage of temporary tables written to disk
Long description:	Task shows the percentage of temporary tables which have been written to disk
Version:	2.31
Default schedule:	13,43 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '1'

Temporary table check (old)

Name:	Temporary table check
Platform:	Mariadb
Category:	Performance
Description:	Shows percentage of temporary tables written to disk
Long description:	Task shows the percentage of temporary tables which have been written to disk
Version:	2.3
Default schedule:	13,43 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & is_mariadb_branch='1']/.[hasengine='YES' & global_status_performance_schema = '0' & global_status_information_schema = '1'

Jobs for Sybase

Control Center Jobs	Description
Availability	
Database backup	Checks backup status of all databases.
Database status	Checks status of all databases defined on the system. The following incidents triggers a warning or an alarm: Offline, offline until recovery complete, recovered, suspect pages, in the process of being upgraded
DBMS uptime	Collects database uptime statistics.
Capacity	
Database space check	Checks the amount of free space in all databases.
Database space check	Checks the amount of free space in all databases.
Database growth rate	Collects database growth rates statistics.
Transaction log space check	Checks the amount of free space in all transaction logs.
Performance	
Blocking detector	This check helps identify processes that are blocking other processes. The ‘time_blocked’ column in system table master..sysprocesses is checked for processes with status ‘lock sleep’.
Data cache monitor	Collects data cache statistics based on the dbWatch tasks ‘System monitor collector’ which runs the ‘dbcc monitor’ utility continuously.
Data cache statistics	Collects data cache statistics.
Disk activity monitor	Collects statistics created by the ‘System monitor collector’ task.
Test DML-DDL performance	Runs performance test on database.The procedure executes SELECT, INSERT, UPDATE and DELETE statements on dbwatch engine schema test tables.
Engine CPU monitor	Collects counter values for busy io and busy cpu, and are based on the dbWatch tasks ‘System monitor collector’ which runs the ‘dbcc monitor’ utility continuously.

Memory statistics	Collects memory statistics based on the dbWatch tasks ‘System monitor collector’ which runs the ‘dbcc monitor’ utility continuously.
Procedure cache check	Checks the procedure cache.
Procedure cache check	Checks the procedure cache.
Session load	Shows session statistics.
System monitor collector	This task collects statistics by running the ‘dbcc monitor’ utility which is also a part of the stored procedure sp_sysmon. It is important that the ‘System monitor collector’ is not used while the sp_sysmon procedure is executed from some where else, as this will clear all the counters, resulting in incorrect statistics. After successful execution of this procedure, the statistics are extracted from the master.dbo.sysmonitors table and stored in a local table (dbw_sysmon_stat) where they can be analyzed by other dbWatch performance tasks.
Temp cache statistics	Collects temp cache statistics.
User connections check	Checks the configuration setting of the ‘number of user connections’ from the master.dbo.syscurconfigs table against the number of connections on the instance (processes with suid != 0 in from the master.dbo.sysprocesses table).
User connections check	Checks the configuration setting of the ‘number of user connections’ from the master.dbo.syscurconfigs table against the number of connections on the instance (processes with suid != 0 in from the master.dbo.sysprocesses table).

Blocking detector

Name:	Blocking detector
Platform:	Sybase
Category:	Performance
Description:	This check helps identify processes that are blocking other processes. The ‘time_blocked’ column in system table master..sysprocesses is checked for processes with status ‘lock sleep’.
Long description:	
Version:	1.1
Default schedule:	1,6,11,16,21,26,31,36,41,46,51,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version > ‘11’ & hasengine=‘YES’

Database backup

Name:	Database backup
Platform:	Sybase
Category:	Availability
Description:	Checks backup status of all databases.
Long description:	
Version:	1.1
Default schedule:	50 5 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '14' & hasengine='YES'

Database space check

Name:	Database space check
Platform:	Sybase
Category:	Capacity
Description:	Checks the amount of free space in all databases.
Long description:	
Version:	2.1
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '14' & hasengine='YES'

Database space check (12)

Name:	Database space check
Platform:	Sybase
Category:	Capacity
Description:	Checks the amount of free space in all databases.
Long description:	
Version:	2.1
Default schedule:	10 5,17 * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version = '12' & hasengine='YES'

Database status

Name:	Database status
Platform:	Sybase
Category:	Availability
Description:	Checks status of all databases defined on the system. The following incidents triggers a warning or an alarm:
Long description:	
Version:	1.1
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '11' & hasengine='YES'

Data cache monitor

Name:	Data cache monitor
Platform:	Sybase
Category:	Performance
Description:	Collects data cache statistics based on the dbWatch tasks ‘System monitor collector’ which runs the ‘dbcc monitor’ utility continuously.
Long description:	
Version:	1.3
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version > ‘11’ & hasengine=‘YES’

Data cache statistics

Name:	Data cache statistics
Platform:	Sybase
Category:	Performance
Description:	Collects data cache statistics.
Long description:	
Version:	1.1
Default schedule:	6,16,26,36,46,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '12' & hasengine='YES'

Database growth rate (detailed)

Name:	Database growth rate (detailed)
Platform:	Sybase
Category:	Capacity
Description:	Collects database growth rates statistics.
Long description:	
Version:	1.3
Default schedule:	10 6 2,6 *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '11' & hasengine='YES'

DBMS uptime

Name:	DBMS uptime
Platform:	Sybase
Category:	Availability
Description:	Collects database uptime statistics.
Long description:	
Version:	1.2
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '11' & hasengine='YES'

Disk activity monitor

Name:	Disk activity monitor
Platform:	Sybase
Category:	Performance
Description:	Collects statistics created by the ‘System monitor collector’ task.
Long description:	
Version:	1.3
Default schedule:	1,11,21,31,41,51 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version = ‘12’ & hasengine=‘YES’

Engine CPU monitor

Name:	Engine CPU monitor
Platform:	Sybase
Category:	Performance
Description:	Collects counter values for busy io and busy cpu, and are based on the dbWatch tasks ‘System monitor collector’ which runs the ‘dbcc monitor’ utility continuously.
Long description:	
Version:	1.3
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version = ‘12’ & hasengine=‘YES’

Memory statistics

Name:	Memory statistics
Platform:	Sybase
Category:	Performance
Description:	Collects memory statistics based on the dbWatch tasks ‘System monitor collector’ which runs the ‘dbcc monitor’ utility continuously.
Long description:	
Version:	1.3
Default schedule:	3,13,23,33,43,53 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version = ‘12’ & hasengine=‘YES’

Procedure cache check

Name:	Procedure cache check
Platform:	Sybase
Category:	Performance
Description:	Checks the procedure cache.
Long description:	
Version:	1.1
Default schedule:	1,21,41 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '12' & hasengine='YES'

Procedure cache check (12)

Name:	Procedure cache check
Platform:	Sybase
Category:	Performance
Description:	Checks the procedure cache.
Long description:	
Version:	1.1
Default schedule:	1,11,21,31,41,51 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version = '12' & hasengine='YES'

Session load

Name:	Session load
Platform:	Sybase
Category:	Performance
Description:	Shows session statistics.
Long description:	
Version:	1.1
Default schedule:	0,10,20,30,40,50 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '11' & hasengine='YES'

System monitor collector

Name:	System monitor collector
Platform:	Sybase
Category:	Performance
Description:	This task collects statistics by running the ‘dbcc monitor’ utility which is also a part of the stored procedure sp_sysmon. It is important that the ‘System monitor collector’ is not used while the sp_sysmon procedure is executed from some where else, as this will clear all the counters, resulting in incorrect statistics. After successful execution of this procedure, the statistics are extracted from the master.dbo.sysmonitors table and stored in a local table (dbw_sysmon_stat) where they can be analyzed by other dbWatch performance tasks.
Long description:	
Version:	1.5
Default schedule:	• * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version = ‘12’ & hasengine=‘YES’

Temp cache statistics

Name:	Temp cache statistics
Platform:	Sybase
Category:	Performance
Description:	Collects temp cache statistics.
Long description:	
Version:	1.1
Default schedule:	6,16,26,36,46,56 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '12' & hasengine='YES'

Test DML-DDL performance

Name:	Test DML-DDL performance
Platform:	Sybase
Category:	Performance
Description:	Runs performance test on database.The procedure executes SELECT, INSERT, UPDATE and DELETE statements on dbwatch engine schema test tables.
Long description:	
Version:	1.1
Default schedule:	10 22 5 1,30
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '11' & hasengine='YES'

Transaction log space check

Name:	Transaction log space check
Platform:	Sybase
Category:	Capacity
Description:	Checks the amount of free space in all transaction logs.
Long description:	
Version:	1.3
Default schedule:	15,45 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type='instance' & databasetype='sybase']/instance[maj_version > '11' & hasengine='YES'

User connections check

Name:	User connections check
Platform:	Sybase
Category:	Performance
Description:	Checks the configuration setting of the ‘number of user connections’ from the master.dbo.syscurconfigs table against the number of connections on the instance (processes with suid != 0 in from the master.dbo.sysprocesses table).
Long description:	
Version:	1.2
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version > ‘12’ & hasengine=‘YES’

User connections check (12)

Name:	User connections check
Platform:	Sybase
Category:	Performance
Description:	Checks the configuration setting of the ‘number of user connections’ from the master.dbo.syscurconfigs table against the number of connections on the instance (processes with suid != 0 in from the master.dbo.sysprocesses table).
Long description:	
Version:	1.2
Default schedule:	2,12,22,32,42,52 * * *
Requires engine install:	Yes
Compatibility tag:	[CDATA[.[type=‘instance’ & databasetype=‘sybase’]/instance[maj_version = ‘12’ & hasengine=‘YES’

Jobs grouped by category

Jobs by category

Jobs in Control Center are grouped by category to make it easier for users to locate the right monitoring job.

The categories are:
Availability Jobs that checks that the database and database related processes are functioning
Capacity – Jobs related to tracking and alerting on capacity issues
Cluster and replication – Jobs designed for or targeting cluster and replication relates issues
Compliance – Jobs related to checking if a database instance is compliment to a standard or to a specific guideline
Consolidation – Jobs related to checking system usage for consolidation efforts
Maintenance – Maintenance jobs focusing on making the database instance run as smooth as possible
Migration – Jobs related to migration from one setup to another
Performance – Jobs related to tracking and alerting on performance issues.

Availability

Jobs in the Availability category

Control Center Jobs	Description
Oracle	
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log.
Alert log check	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Alert log check	This check reads and looks for errors in the database alert log. Uses embedded java to read files.
Archive status Check	Checks how many redolog files that are not archived.
Backup log Check 10g	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Backup log Check 8i	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Backup log Check 9i	Checks for errors in a log file (e.i. backup log). For backups not using RMAN.
Data Guard Archive Status	Checks that the standby database is receiving archive files from master database.
Data Guard Archive Status Check	Checks that the standby database is receiving archive files from master database.
Database link check	Checks database links.
Instance alert log	Reads and checks errors in the X$DBGALERTEXT performance view contained in the XML decoded version of the XML version of Alert log file.
Database uptime	Collects database uptime statistics for all instances mounted on the database.
Dba Jobs check	Detecting failed scheduled jobs in old style dba_jobs.
DBMS uptime	Collects uptime statistics in database.
Export log Check	Checks for errors in a log file (i.e. export log).

File status check	Reacts on any changes in status of data files and/or temporary files.
Index status check	Detecting indexes in status UNUSABLE
Invalid objects check	Detecting invalid objects in the database.
Job scheduling check	Detecting failed scheduled jobs.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors from the database listener log. Using Oracle native code to read.
Listener log check	This checks and reads errors in database listener log. Using embedded java to read.
Listener log check	This checks and reads errors in database listener log. Using embedded java to read.
Listener status check	Checks listener status to verify if up and running, requires java support in the database.
Max datafiles check	Checks the “soft limit” and the “hard limit” of maximum number of physical OS files, that can be mapped to an Oracle instance.
Database mount state	Checking if database is in correct mount status
Datafile status	Checking status for datafiles
Max datafiles	Checking if datafiles reaches max datafiles (db_files) parameter
RMAN backup status	Checking status of the last RMAN backup
Password expire	Checks for users whose password will soon expire.
PDB status	Alerts if PDB is not in correct state
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.

RMAN backup status	Checks status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN archivelog backup status	Checks the status of RMAN archivelog backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
RMAN backup status	Checks the status of RMAN backup from V$RMAN_BACKUP_JOB_DETAILS performance view.
Test alert db	Test alert that alerts every 'X' minutes.
Test alert	Test alert that alerts every 'X' minutes.
TNSping check	Checks TNSping connectivity on listed targets. Task requires 'java' support. Supports Linux/Unix and Windows, will react with an alarm if host is unreachable.
MS SQL Server	
Agent Jobs Check	Checks whether there exists jobs on the SQL server which have not been executed, or have failed during execution.
Agent Jobs Check *for MS2000*	Checks whether there exists jobs on the SQL server which have not been executed or have failed during execution.
SQL Server Agent status	Checks if the SQL Server Agent is running.
Database backup *for MS2000*	This procedure analyzed the backup statistics from msdb.dbo.backupset table (excluding system databases: master, model and msdb).
Database backup *for system dbs MS2000*	This procedure analyzed the system database backups statistics from msdb.dbo.backupset table.
Database backup	This procedure analyzes the backup statistics (type D and I) from the msdb.dbo.backupset table (excluding the system databases: master, model and msdb).
Database backup *for SIMPLE recovery model*	This procedure analyzes the backup statistics (type D) from the msdb.dbo.backupset table for databases in SIMPLE recovery model (excluding the system databases: master, model and msdb).
Database backup *for system databases*	This procedure analyzes the FULL backup statistics for the system databases (master, model and msdb) using data from the msdb.dbo.backupset table.
Database Log backup	This procedure checks the database transaction log backups from the msdb.dbo.backupset table (excluding system databases: master, model and msdb).
Collation check	Checks if there is a collation conflict with temp tables and table variables.
Check database recovery mode	Checks if all databases are running in FULL or SIMPLE recovery mode.

Database status	Checks if all databases have status ONLINE
Database status *for MS2000*	Checks if all databases have status ONLINE
Database Server uptime	Collects database server uptime statistics.
Database Server uptime *for MS2000*	Collects database server uptime statistics.
Database Server uptime *for MS2005*	Collects database server uptime statistics.
Instance error log	Reads and checks the Instance error log file by using the sp_readerrorlog stored procedure.
Instance error log *for AWS*	Reads and checks the Instance error log file by using the AWS rdsadmin.dbo.rds_read_error_log stored procedure.
Instance status	This alert checks if the instance has been restarted since the last check
Database backup *noschema*	This alert checks if the instance has been restarted since the last check
Database log backup *noschema*	This alert checks if the instance has been restarted since the last check
Database status *noschema*	This alert checks if the instance has been restarted since the last check
Missing database backup *noschema*	This alert checks if the instance has been restarted since the last check
Missing database log backup *noschema*	This alert checks if the instance has been restarted since the last check
Program status	Checks for any program connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Report Server status	Checks if the program 'Report Server' is connected to the SQL Server by checking the program_name column in the master.dbo.sysprocesses table.
Restricted Enterprise edition features	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Restricted Enterprise edition features *for MS2000*	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).
Restricted Enterprise edition	Checks if there are any restricted features in use (supported only by Enterprise or Developer SQL Server edition).

features *for MS2005*	
PostgreSQL	
Backup check – pg_dump	Checks that there exists an up to date backup file.
Backup check – WAL	Checks that the WAL files have been backed up.
DBMS uptime	Collects database uptime statistics.
dbWatch engine alert	Collects database uptime statistics.
MySQL	
DBMS uptime	Collects uptime statistics in the database.
MySQL aborted connects	Checking if there is many aborted connects
MySQL max connections	Checking if the system is near maximum connections
MySQL mysqld process	Checking if mysqld process is running
MariaDB	
DBMS uptime	Collects uptime statistics in the database.
DBMS uptime	Collects uptime statistics in the database.
Aborted connects alert	Checking if there is many aborted connects
Maximum connections alert	Checking if the system is near maximum connections
MySQLd process alert	Checking if mysqld process is running
MariaDB Space Optimizer	Tells the user how much space tables in a specific database consume.
Sybase	
Database backup	Checks backup status of all databases.
Database status	Checks status of all databases defined on the system. The following incidents triggers a warning or an alarm: Offline, offline until recovery complete, recovered, suspect pages, in the process of being upgraded
DBMS uptime	Collects database uptime statistics.

Capacity

Jobs in the Capacity category

Control Center Jobs	Description
Oracle	
ASM disk statistics	Checks ASM disks for status and statistics.
ASM diskgroup space	Checks ASM diskgroups space statistics.
Monitor SYS.AUD$ table size	Checks size of the AUD$ trail table.
Autoextensible data files	Checks all tablespaces for total disk space usage (autoextensible data files).
Disk space check	Disk space check requiring java support database. (Supports Linux, Solaris, AIX, HPUX and Windows).
Flash Recovery Area Usage	Checks space usage in the flash recovery area.
Flash Recovery Area Usage	Checks space usage in the flash recovery area.
Free extents	Checks for free extents in all tablespaces. A warning or an alarm is returned if segments storage value is higher than the largest free extent chunk.
Max processes	Checks the maximum number of processes.
ASM diskgroup space	Checking available space on the ASM diskgroups
Extent fragmentation	Checking for extent fragmentation in dba_free_space
Flash recovery space usage	Checking available space on the Flash recovery area
Max processes	Checking if database reaches max processes (processes) parameter
SYS.AUD size	Checking if SYS.AUD$ fills up with data
Tablespace free space	Checking tablespaces for free space
Segment size collector	Collects size and number totals on all segment types per segment-owner and tablespace_name.

Segment size collector	Collects size and number totals on all segment types per segment-owner and tablespace_name.
Segment size collector	Collects size and extent number info for larger segments in the database.
Segment size status for old style tablespaces	Checks segment storage definitions in tablespaces where extent management is not set to local.
Temporary tablespace free space check	Checks level of free space for temporary tablespaces.
Tablespace free space check	Checks level of free space for all tablespaces.
MS SQL Server	
Auto growth event collector	Collects statistics on how often an auto-growth event has occurred.
Data file size check	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Data file size check *for MS2000*	Checks the remaining space for all databases where the data files are set with limited growth rate (max size not unlimited).
Database growth rate *aggregated*	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.
Database growth rate *aggregated for MS2000*	Collects size of all database files (including transaction log files) to visualize the growth rate for all databases.
Database growth rate *detailed*	Collects size of database files (including transaction log files) to visualize the growth rate for the largest databases.
Database growth rate *detailed for MS2000*	Collects size of database files (including transaction log files) to visualize the growth rate for the largest databases.
Database disk capacity	Checks free space on drives where data and transaction log files are defined. An alarm (or warning) is raised if the percentage limit is reached OR if the abosulte limit is reached. IF the xp_cmdshell Instance configuration option is enabled the alert can check disk and mounted volumes where data-files are not present.
Database disk	Checks free space on drives where all data and transaction log files are defined. Drives

space usage	where no data files exist will be ignored.
Databases NOT IN USE collector	Collects information about most inactive databases.
Disk space check	Checks the amount of free space on the available disk drives.
Instance error log file size check	Checks the size og the Instance error log file by using the extended stored procedure xp_cmdshell.
Filegroups growth rate	This procedure collects space usage for the largest filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined on a SQL Server instance.
Filegroups growth rate *for MS2000*	This procedure collects space usage for the largest filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined on a SQL Server instance.
Data file size check *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Database disk capacity *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Disk space check *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Transaction log space usage *noschema*	This procedure collects space usage in all filegroups defined in each database. The undocumented ‘sp_MSforeachdb’ Stored Procedure is used to execute T-SQL statements against dbo.sysfiles table in every database defined to a SQL Server instance.
Objects size collector *all databases*	Collects table and index size information for the largest objects for all databases.
Temporary database space usage	Checks space usage in tempdb database, and collects statistics including size of data and transaction log files.
Transaction log size check	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.

Transaction log size check	This procedure checks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size). If the size of the transaction log file(s) in percentage is greater than the warning/alarm parameter value, the check returns a warning/alarm.
Transaction log space usage	This procedure checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Transaction log space usage *for MS2000, MS2005 and MS2008*	This procedure checks information returned by DBCC SQLPERF which is used to monitor the amount of space used in the transaction log.
Truncate transaction log	This procedure truncates the transaction log for the dbWatch database.
Version store space usage	Checks total space in tempdb used by version store records for each database.
PostgreSQL	
Log size statistics	Checks the size of the log file.
Schema growth collector	Checks the size of the log file.
Vaccum alert	Checks the size of the log file.
Schema growth and information	Collects detailed schema growth rate information.
Schema growth statistics	Collects and analyzes schema growth statistics from data collected by
Tablespace growth and information	Shows an overview of tablespace growth and statistics.
MySQL	
Database growth rate	Collects database size and visualizes the growth rate of the server.
Database growth rate	Collects detailed growth rate information.
MariaDB	
Database growth rate	Collects database size and visualizes the growth rate of the server.
Database growth rate	Collects detailed growth rate information.
Sybase	

Database space check	Checks the amount of free space in all databases.
Database space check	Checks the amount of free space in all databases.
Database growth rate	Collects database growth rates statistics.
Transaction log space check	Checks the amount of free space in all transaction logs.

Cluster and replication

Jobs in the Cluster and replication category

Control Center Jobs	Description
Oracle	
Blocking detector for RAC	Checks if a session is waiting on a TX (transaction) lock. (RAC only)
Memory session load for RAC	Records the load memory of RAC active sessions over time.
MV Refresh Group	Checks refresh date of scheduled jobs in refresh group(s).
Dataguard apply time	Checking apply time in dataguard standby nodes
Dataguard archive sequence apply lag	Checking lag of applied archive logs on dataguard standby node
Dataguard standby archivelog gap	Checking for archivelog gap between threads on primary and standby instances in a replication/service group. Metadata “replicationgroup” must be set on all instances to the same as parameter replicationgroup. For upto 10 redo threads.
Dataguard startup time	Checking startup time on dataguard standby nodes
Dataguard transport time	Checking transport time in dataguard standby nodes
RAC instances status	Reacts if one or more RAC instances are not available.
Session load Rac	Records the number of active sessions over time (RAC only).
Snapshot Log(s) rows count	Checks row count on all snapshot logs for given schema.
Snapshot Log(s) size	Checks size of all snapshot logs.

Data Guard role switch	This job checks if Primary/Standby switch occurs.
MS SQL Server	
Database mirroring	Checks state information of all mirrored databases.
Failover cluster host switch	Checks if an instance switched to a differen host i a Windows Server Failover Cluster (WSFC).
Health state	Checks groups health state.
Log shipping monitor	Monitor the primary database in each log shipping configuration, including information about the last backup file and last restored file.
Log shipping monitor	Monitor the secondary database in each log shipping configuration.
Member state	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
AlwaysOn database backup alert	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
AlwaysOn transaction log backup alert	Checks the status of each member node of the current WSFC cluster based on Dynamic Management Views sys.dm_hadr_cluster_members.
Replica states	Checks role state for all alwayson groups.
Replication status	The following check provides general info in regards to any replication going on in a server.
PostgreSQL	
BDR Replication lag	Checks for BDR replication lag
Replication delay alert	Checks for BDR replication lag
Replication slave client alert	Checks for BDR replication lag
Replication slave client alert	Checks for BDR replication lag

MySQL	
NDB data memory usage check	NDB data memory usage check
NDB data node status	NDB data node status
Innodb cluster status	This job checks state of all members in Innodb cluster
Innodb cluster switch	This job checks if Innodb cluster PRIMARY/SECONDARY switch occurs.
Replica count	This job checks all of replicas currently registered with the source.
Replica delay	This job checks how “late” the replica is, and measures the time difference in seconds between the replication SQL (applier) thread and the replication I/O (receiver) thread.
Replica state	This job checks if replica changes occurs. For example, if an instance becomes a replica or if an instance is no longer a replica.
MariaDB	
NDB data memory usage check	NDB data memory usage check
NDB data node status	NDB data node status
Replica count	This job checks all of replicas currently registered with the source.
Replica delay	This job checks how “late” the replica is, and measures the time difference in seconds between the replication SQL (applier) thread and the replication I/O (receiver) thread.
Replica state	This job checks if replica changes occurs. For example, if an instance becomes a replica or if an instance is no longer a replica.

Compliance

Jobs in the Compliance category

Control Center Jobs	Description
MS SQL Server	
Optional	
MS SQL Server patch status	Checks latest updates for Microsoft SQL Server.

Consolidation

Jobs in the Consolidation category

Control Center Jobs	Description
MS SQL Server	
Optional	
Database statistics	This procedure gathers statistics per database.

Maintenance

Jobs in the Maintenance category

Control Center Jobs	Description
Oracle	
Framework	Engine framework is used when upgrading earlier versions of dbWatch. Ensures the dbWatch Engine and dbWatch Server run compatible codes.
Optional	
Analyze tables	Analyse tables automatically.
Delete SYS.AUD$ data	Deletes old rows from the SYS.AUD$ trail table.
Rebuild indexes	Rebuild indexes automatically.
MS SQL Server	
Autogrow settings	Checks database files auto-growth settings.
framework	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
framework *for MS2000*	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Optional	
Backup All databases	Takes backup of all application and system databases.
Backup All transaction logs	Takes backup of all transaction logs for databases running in FULL recovery mode.
Check database and server principal mapping	Checks if the database owner (dbo) is maped into any Server Login (server principal).
Cleanup MSDB history tables	This task deletes entries in the MSDB database history tables which holds statistics of backup/restore, jobs and maintenance plan executions.
Cycle error log	This task cycle MS SQL Server error log and Agent error log files.
DBCC CHECKDB	Checks the logical and physical integrity of all the objects in all databases by performing the DBCC CHECKDB operation.
DBCC UPDATEUSAGE	This task is performing the DBCC UPDATEUSAGE operation to corrects pages and row count inaccuracies in the catalog views. These inaccuracies may cause incorrect space

	usage reports returned by the sp_spaceused system stored procedure.
External fragmentation *all databases*	External (logical) fragmentation occurs when an index leaf page is not in logical order. It occurs when the logical ordering of the index does not match the physical ordering of the index. This causes SQL Server to perform extra work to return ordered results
Internal fragmentation check	Checks the internal fragmentation for tables and indexes in all databases. The information is extracted from the dynamic management function (view) sys.dm_db_index_physical_stats.
SQL Server performance counters	Checks if SQL Server performance counters are missing.
Rebuild indexes	Rebuilds fragmented indexes in all databases.
Rebuld indexes in table	Rebuilds fragmented indexes in all tables listed in the ‘table list’ parameter.
Reorganize indexes	Reorganizes fragmented indexes in all databases.
Reorganize indexes in table	Reorganizing fragmented indexes in all tables listed in the ‘table list’ parameter.
Shrink transaction logs	This procedure shrinks transaction log files which are detected by ‘Transaction log size check’ alert which cheks the size of the transaction log (log file(s) size), and compares it to the database size (data file(s) size).
Suspect pages	Monitors suspect pages statistics in suspect_pages table.
Update index statistics	Update statistics in all (non-system) databases.
Update statistics	Update statistics in all (non system) databases.
PostgreSQL	
Backup job – pg_dump_all	Runs a backup of all the postgres databases on an instance using pg_dumpall through an ssh connection.
framework	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Analyze tables	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.
Vaccum tables	dbWatch engine framework job (for internal use only). Used for patching or upgrading of dbWatch engine framework.

Migration

Jobs in the Migration category

Control Center Jobs	Description
Oracle	
Optional	
Statistics migration	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.
MS SQL Server	
Optional	
Statistics migration	Migrates statistics from a number of tasks from dbWatch version 12 to dbWatch CC.

Performance

Jobs in the Performance category

Control Center Jobs	Description
Oracle	
Blocking detector	Checks if a session is waiting on a TX (transaction) lock.
Buffer cache statistics	Gets data cache statistics for buffer cache in SGA.
CPU load	Checks CPU load using Oracle performance view v$osstat.
Database network statistics	Checks database network SQL*NET statistics.
Test DML performance	Runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on schema test tables.
File IO statistics	Collects I/O statistics for data files.
Instance memory statistics	Collects statistics on instance memory.
Latch statistics	Collects latch status statistics.
Long running queries	Checks any long running queries and then alerts.
Open cursors	Checks and collects the amount of open cursors per session.
Disk read statistics	Collects disk read statistics from v$system_event performance view for db file sequential and dbfile scattered read event.
Redo statistics	Gets redolog files statistics. Based on Oracle dictionary and performance views.
Redo statistics	Gets redolog files statistics. Based on Oracle dictionary and performance views.
Session load	Records the number of active and inactive sessions over time.
Session load	Records the number of active sessions over time.

SQL waits	Collects SQL wait statistics.
Table statistics check	Reacts on old (or missing) statistisc in DBA_TABLES dictionary table (column LAST_ANALYZED).
Top user memory usage	Check can be used to trace memory usage per server-process (session).
Undo statistics	Collects rollback segment statistics.
User memory statistics	Collects user memory statistics.
Wait statistics	Collects waits statistics encountered by threads that executed. This task is based on the v$system_event dynamic performance view.
Optional	
SQL statistics	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views.
SQL statistics	Collects SQL statements statistics from V$SQLAREA and V$SQL_PLAN dynamic performance views. For RDS only.
MS SQL Server	
Blocking detector	Checks whether there exists blocked session.
Blocking statistics	Checks whether there exists any blocked sessions.
Cached query statistics	This task returns aggregate performance statistics based on cached query plans in SQL Server.
Data cache memory usage	Collects data cache memory usage per database (for top 10 databases).
Data cache memory usage *for MS20005 and MS2008*	Collects data cache memory usage per database (for top 10 databases).
Data hit ratio	Monitors the buffer cache hit ratio by extracting counter values from the master.dbo.sysperfinfo table for the counters ‘Buffer cache hit ratio’ and ‘Buffer cache hit ratio base’.

Database session load	Shows the number of connections over time per database, host and application.
Test DML-DDL performance	Runs performance test on the database. The procedure executes SELECT, INSERT, UPDATE, DELETE (and TRUNCATE) statements on the test table.
File IO statistics	Shows the number of connections over time per database, host and application.
High activity monitor	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
High activity monitor *for MS2005*	Collects SQL instance CPU usage, CPU usage of other processes, logical and physical reads, and active processes count on the SQL Server instance.
Index usage statistics *all databases*	This procedure collects statistics from sys.dm_db_index_usage_stats performance view which gives information on how an index (or a table – heap) has been used to resolve queries.
Instance memory check	This job checks the target memory value of the SQL Server instance, and gives a warning/ alarm if the instance is not able to allocate a
Instance memory usage	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Instance memory usage *for MS2005 and MS2008*	Collects total memory usage and data cache memory usage for SQL Server instance. This dbWatch task can be configured to automatically reduce the amount of memory used by the SQL Server instance.
Lazy writer and Checkpoint statistics	Monitors ‘Checkpoint pages/sec’, ‘Lazy writes/sec’ and ‘Page life expectancy’ by extracting counter values from the sys.dm_os_performance_counters performace table.
Memory objects statistics	This task collects the size of memory objects that are currently allocated by the SQL Server
Blocking detector *noschema*	Checks whether there exists any blocked sessions.
Instance memory check *noschema*	The alert checks the target memory value of the SQL Server instance, and gives a warning/ alarm if the instance is not able to allocate a certain percentage of the total server/machine memory.

Server memory statistics	This tasks collects statistics from the sys.dm_os_sys_info performance view about the memory resources available to and consumed by the SQL Server.
Sessions per database	This task returns aggregate performance statistics based on sessions connected to the SQL Server instance.
Session load	Shows the number of active sessions over time.
Transactions log flushed bytes load	Shows bytes flushed to transaction logs over time, total and top 5 databases.
Transactions log flushed bytes load *for MS2000*	Shows bytes flushed to transaction logs over time, total and top 5 databases.
Transactions load	Shows the transactions load over time, total and top 5 databases.
Wait statistics	Collects statistics about all waits encountered by threads that executed. This task is based on the sys.dm_os_wait_stats dynamic performance view.
Optional	
SQL statistics	Collects performance statistics for the SQL statements.
Query Store status	Checks Query Store space usage in every database where it is enabled.
PostgreSQL	
Buffer cache reads per database	Collects the number of block fetch requests for table or index per database.
Buffer cache statistics	Gets data cache statistics from pg_stat_database view.
Disk block hitrate	Checks the hitrate for disk block requests.
Disk block reads per database	Collects the number of disk block fetch requests for table or index per database.
Test DML performance	Runs performance test. Procedure executes SELECT, INSERT, UPDATE, DELETE and TRUNCATE statements on schema test tables.
Index block hitrate	Checks the hitrate for index disk block requests.

Locks held and statistics	Shows locks held statistics.
Table and index scan collector	Shows locks held statistics.
Session load	Gathers session load statistics.
Table and index scan statistics	This task gathers scan statistics on tables and indexes from pg_stat_all_tables view.
Transaction statistics	Gathers information on commits and rollbacks in the databases and generates statistics.
Optional	
SQL statistics *for 13 and 14*	Collects performance statistics for the SQL statements.
SQL statistics *for 9.4 to 12*	Collects performance statistics for the SQL statements.
MySQL	
Binlog cache check	Binlog cache performance check
Innodb buffer pool check	Checks the hit rate of the innodb buffer pool
Key buffer check	Checks key buffer efficiency
Database load	Provides information on the server load
Lock statistics	Collects lock statistics
Memory setup	Analyzes the memory setup of the server
Network traffic	Shows the network traffic
Query cache hitrate	Shows the query cache hit rate
Session load	Shows connection statistics

Thread cache hitrate	Shows the thread cache hit rate
Temporary table check	Shows percentage of temporary tables written to disk
MariaDB	
Binlog cache check	Binlog cache performance check
Binlog cache check	Binlog cache performance check
Innodb buffer pool check	Checks the hit rate of the innodb buffer pool
Innodb buffer pool check	Checks the hit rate of the innodb buffer pool
Key buffer check	Checks key buffer efficiency
Key buffer check	Checks key buffer efficiency
Database load	Provides information on the server load
Database load	Provides information on the server load
Lock statistics	Collects lock statistics
Lock statistics	Collects lock statistics
Memory setup	Analyzes the memory setup of the server
Memory setup	Analyzes the memory setup of the server
Network traffic	Shows the network traffic
Network traffic	Shows the network traffic
Query cache hitrate	Shows the query cache hit rate
Query cache	Shows the query cache hit rate

hitrate	
Session load	Shows connection statistics
Session load	Shows connection statistics
Thread cache hitrate	Shows the thread cache hit rate
Thread cache hitrate	Shows the thread cache hit rate
Temporary table check	Shows percentage of temporary tables written to disk
Temporary table check	Shows percentage of temporary tables written to disk
MariaDB Index Redundancy Checker	Shows percentage of temporary tables written to disk
Sybase	
Blocking detector	This check helps identify processes that are blocking other processes. The 'time_blocked' column in system table master..sysprocesses is checked for processes with status 'lock sleep'.
Data cache monitor	Collects data cache statistics based on the dbWatch tasks 'System monitor collector' which runs the 'dbcc monitor' utility continuously.
Data cache statistics	Collects data cache statistics.
Disk activity monitor	Collects statistics created by the 'System monitor collector' task.
Test DML-DDL performance	Runs performance test on database.The procedure executes SELECT, INSERT, UPDATE and DELETE statements on dbwatch engine schema test tables.
Engine CPU monitor	Collects counter values for busy io and busy cpu, and are based on the dbWatch tasks 'System monitor collector' which runs the 'dbcc monitor' utility continuously.
Memory statistics	Collects memory statistics based on the dbWatch tasks 'System monitor collector' which runs the 'dbcc monitor' utility continuously.
Procedure cache check	Checks the procedure cache.
Procedure cache check	Checks the procedure cache.
Session load	Shows session statistics.

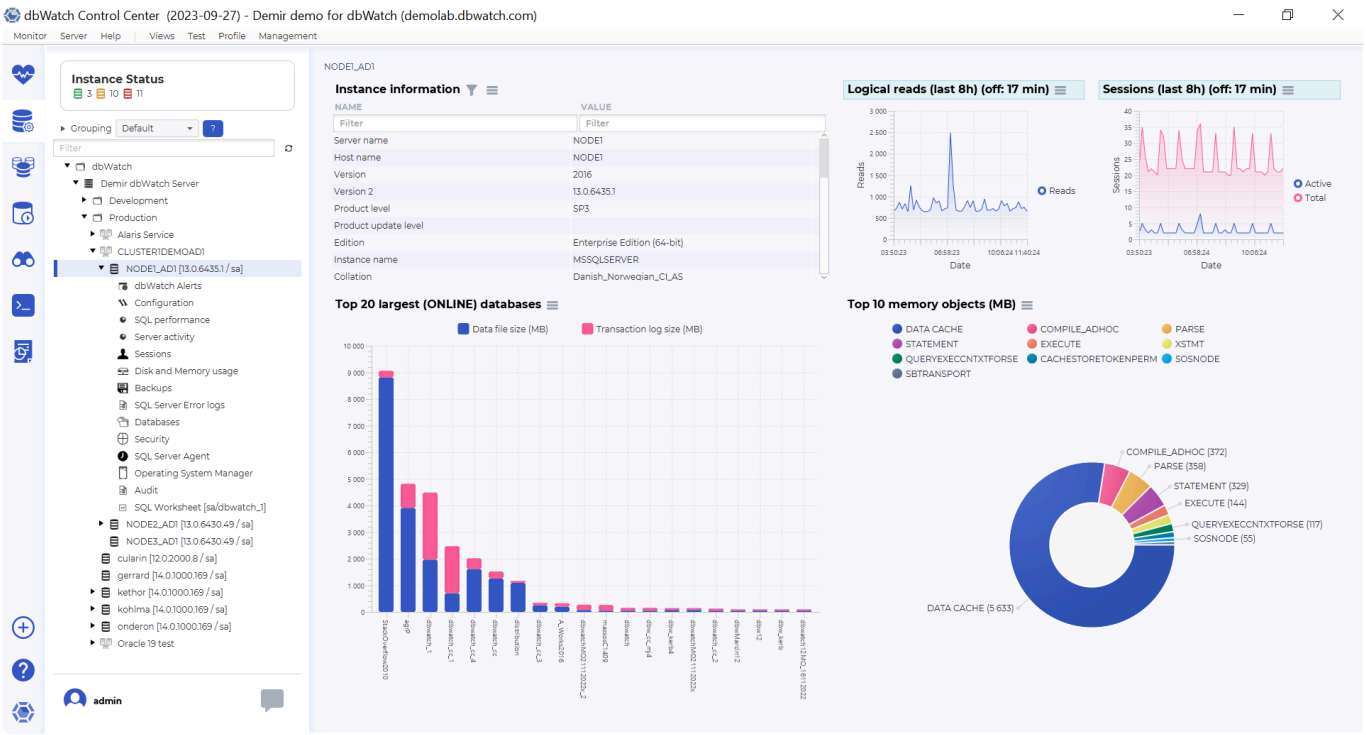
System monitor collector	This task collects statistics by running the ‘dbcc monitor’ utility which is also a part of the stored procedure sp_sysmon. It is important that the ‘System monitor collector’ is not used while the sp_sysmon procedure is executed from some where else, as this will clear all the counters, resulting in incorrect statistics. After successful execution of this procedure, the statistics are extracted from the master.dbo.sysmonitors table and stored in a local table (dbw_sysmon_stat) where they can be analyzed by other dbWatch performance tasks.
Temp cache statistics	Collects temp cache statistics.
User connections check	Checks the configuration setting of the ‘number of user connections’ from the master.dbo.syscurconfigs table against the number of connections on the instance (processes with suid != 0 in from the master.dbo.sysprocesses table).
User connections check	Checks the configuration setting of the ‘number of user connections’ from the master.dbo.syscurconfigs table against the number of connections on the instance (processes with suid != 0 in from the master.dbo.sysprocesses table).

Management

In this section, we'll be familiarized with how the Management Module works.

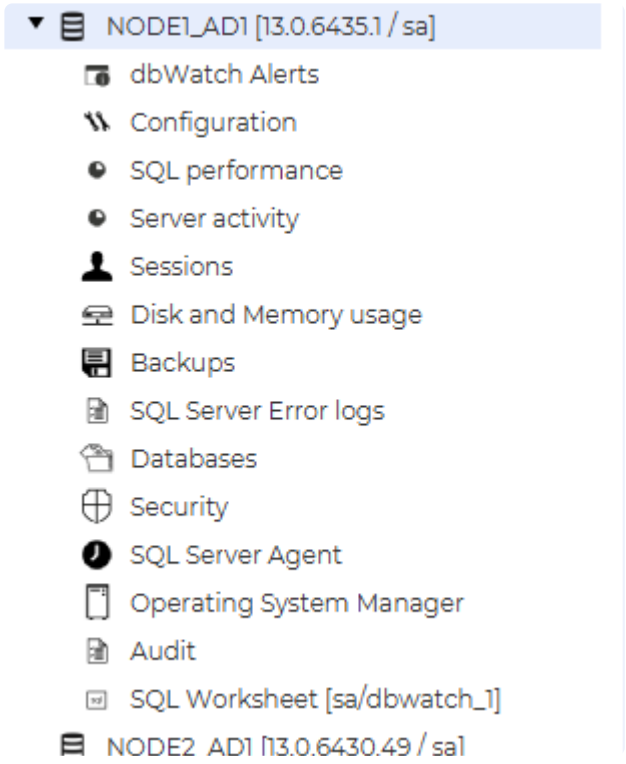
Examples in this documentation are from Microsoft SQL Server, but this is available on all platforms, however, contents are platform and license dependent.

When opening the management module in its initial state, you get the status dashboard:



This will give an overview of the system, displaying some instance information, the largest databases, logical reads, session load, and memory usage.

There is a tree structure on the left side of the main window, with areas focusing on specific areas of the database instance.



The management tree structure nodes are different from database platform to database platform, and can also depend on extracost licensing.

For more details on the management possibilities for a database platform, check out the individual pages for each platform:

Microsoft SQL Server

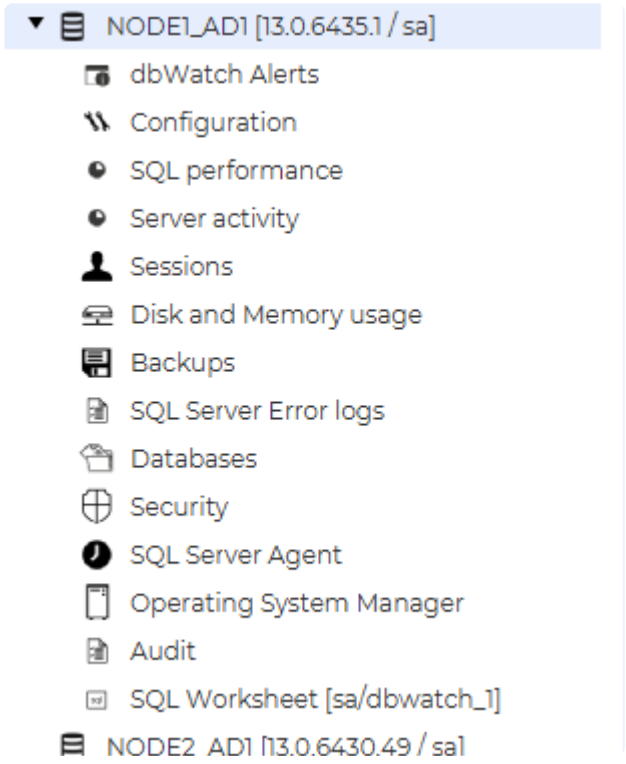
<< Performance / Management on SQL Server >>

Management on SQL Server

When opening the management module in its initial state, you get the status dashboard:

This will give an overview of the system, displaying some instance information, the largest databases, logical reads, session load, and memory usage. When a graph indicates “Off: 17 min” there is a time drift between the client and the database server.

There is a tree structure on the left side of the main window, with areas focusing on specific areas of the database instance.



The subnodes of this wiki page will go into detail for each node in the management tree.

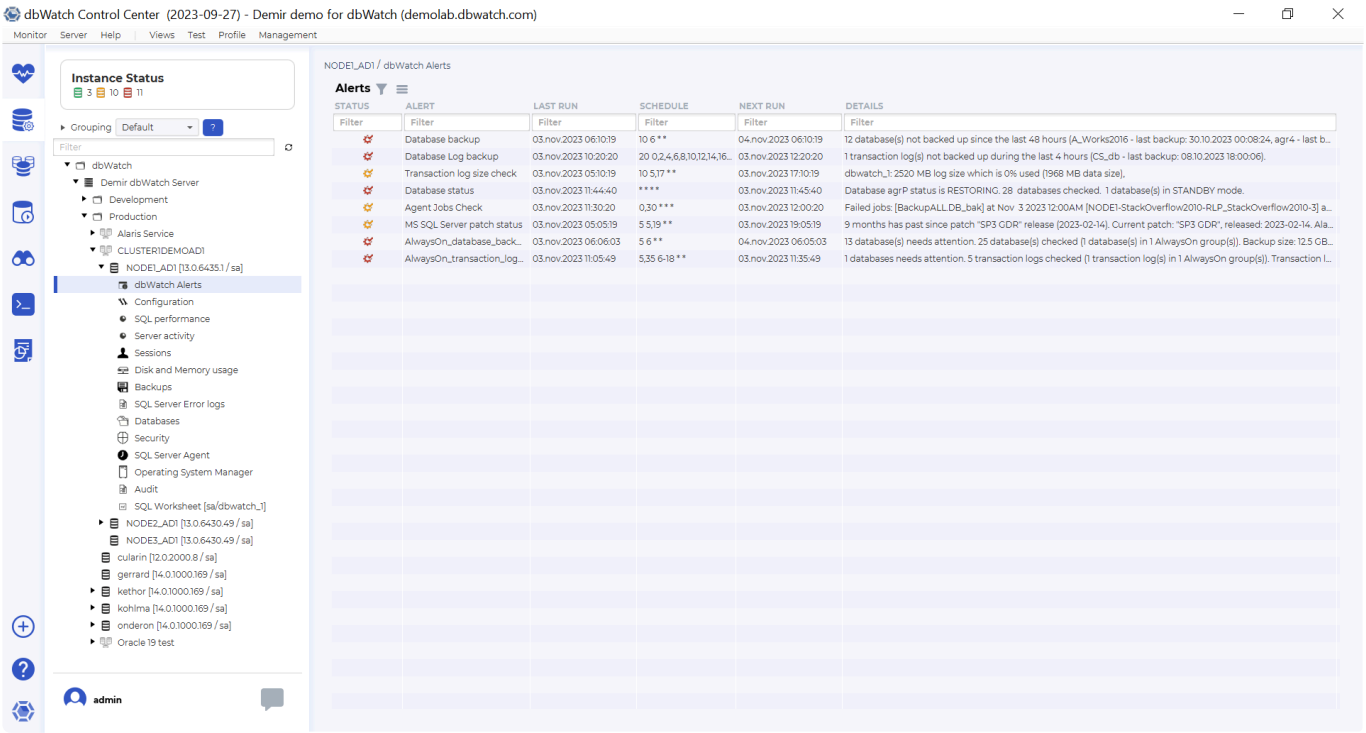
- dbWatch alerts
- Configuration
- SQL performance
- Server activity
- Sessions
- Disk and Memory usage
- Backups
- SQL Server Error logs
- Databases
- Security
- SQL Server Agent
- Operating System Manager
- Audit
- SQL Worksheet

<< Management / dbWatch alerts >>

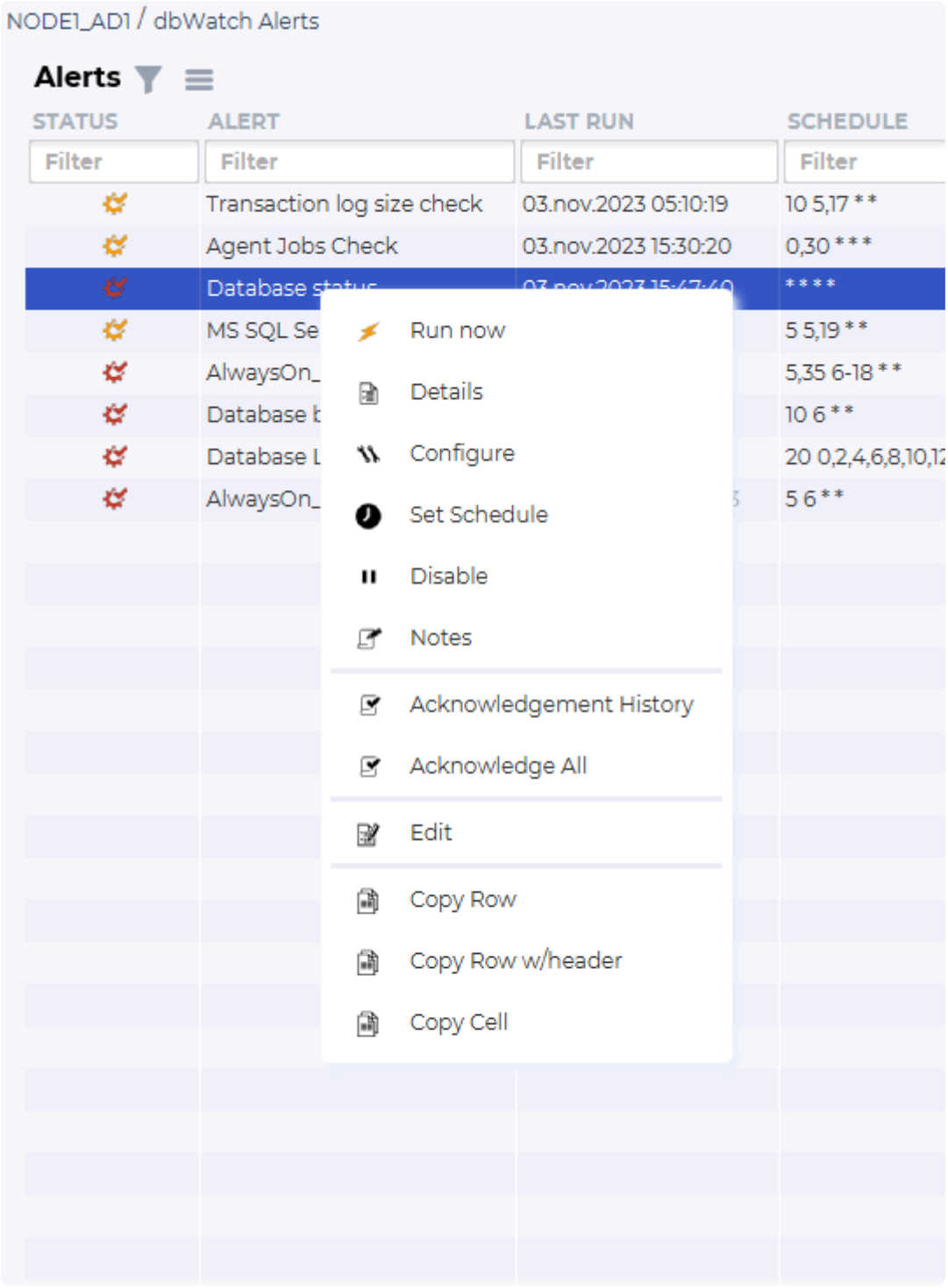
dbWatch alerts

dbWatch alerts is a list of monitoring jobs in the database in warning or alarm state for this database instance.

Example:



This is handy so you know you are working on the right database instance when you are correcting issues.

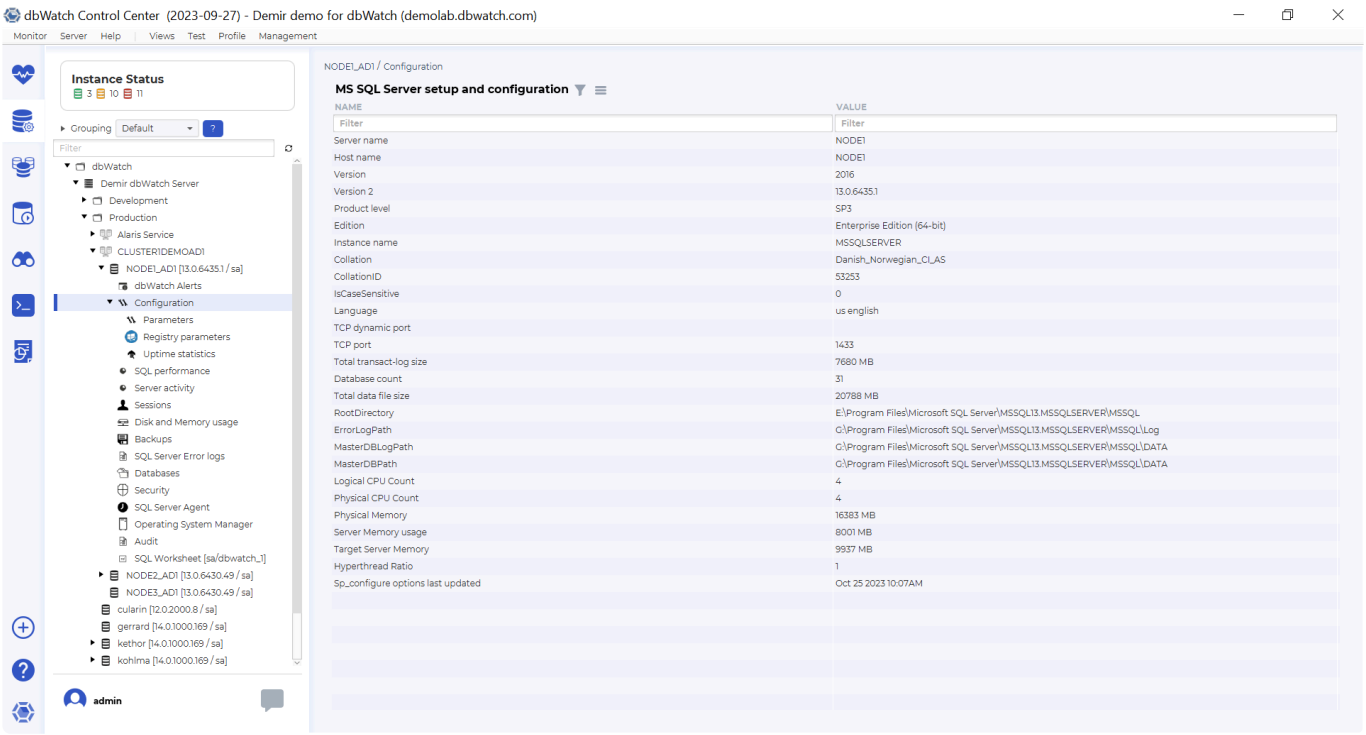


You can interact with the alert lines, to re-run the monitoring jobs, get the monitoring job report, configure the monitoring job, set schedule, disable job, edit notes and all other functions available from the monitoring interface for this job.

<< Management on SQL Server / Configuration >>

Configuration

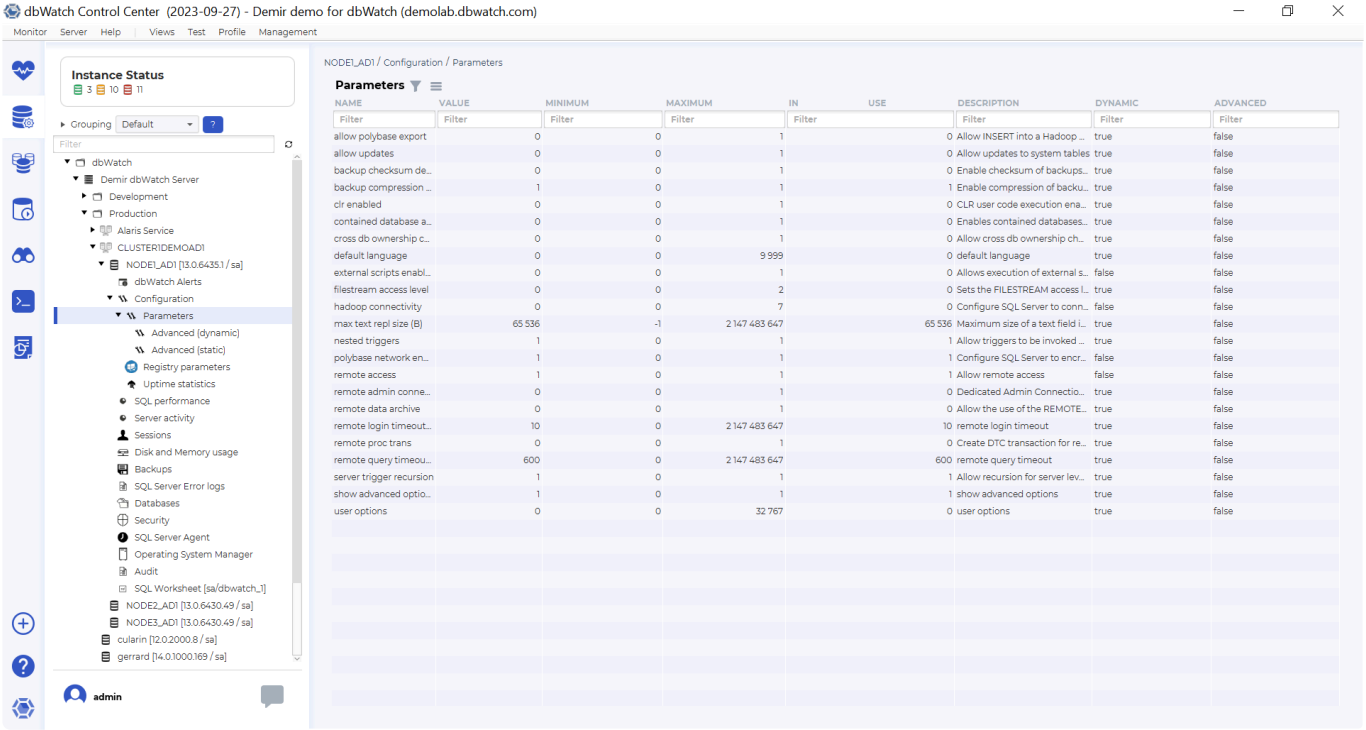
Configuration gives an overview of the current setup of the database instance, including parameters and uptime statistics.



<< dbWatch alerts / Parameters >>

Parameters

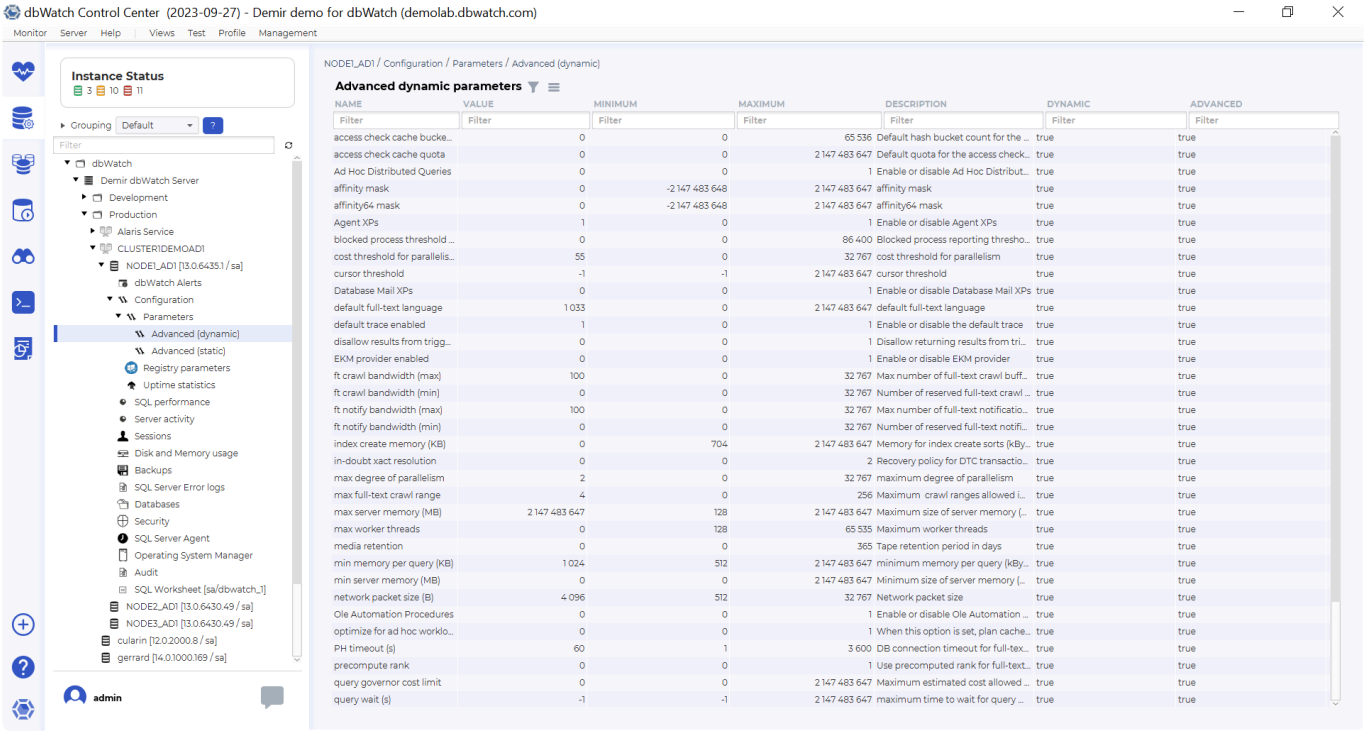
First the basic parameters are showed. You can right_click on any of these to configure new values.



[<< Configuration](#) / [Advanced dynamic parameters >>](#)

Advanced dynamic parameters

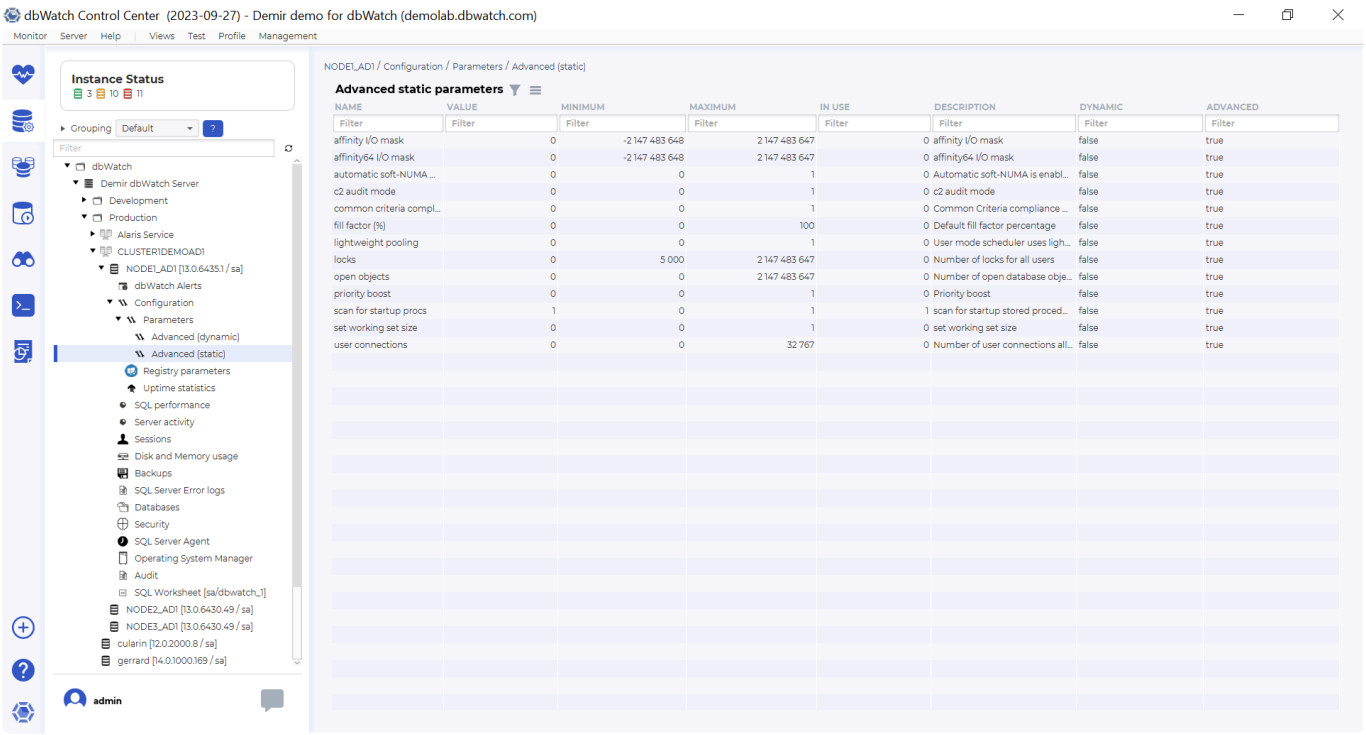
Advanced dynamic parameters are more advanced, but they can be configured by right-clicking and choosing configure. They take effect immediately.



<< Parameters / Advanced static parameters >>

Advanced static parameters

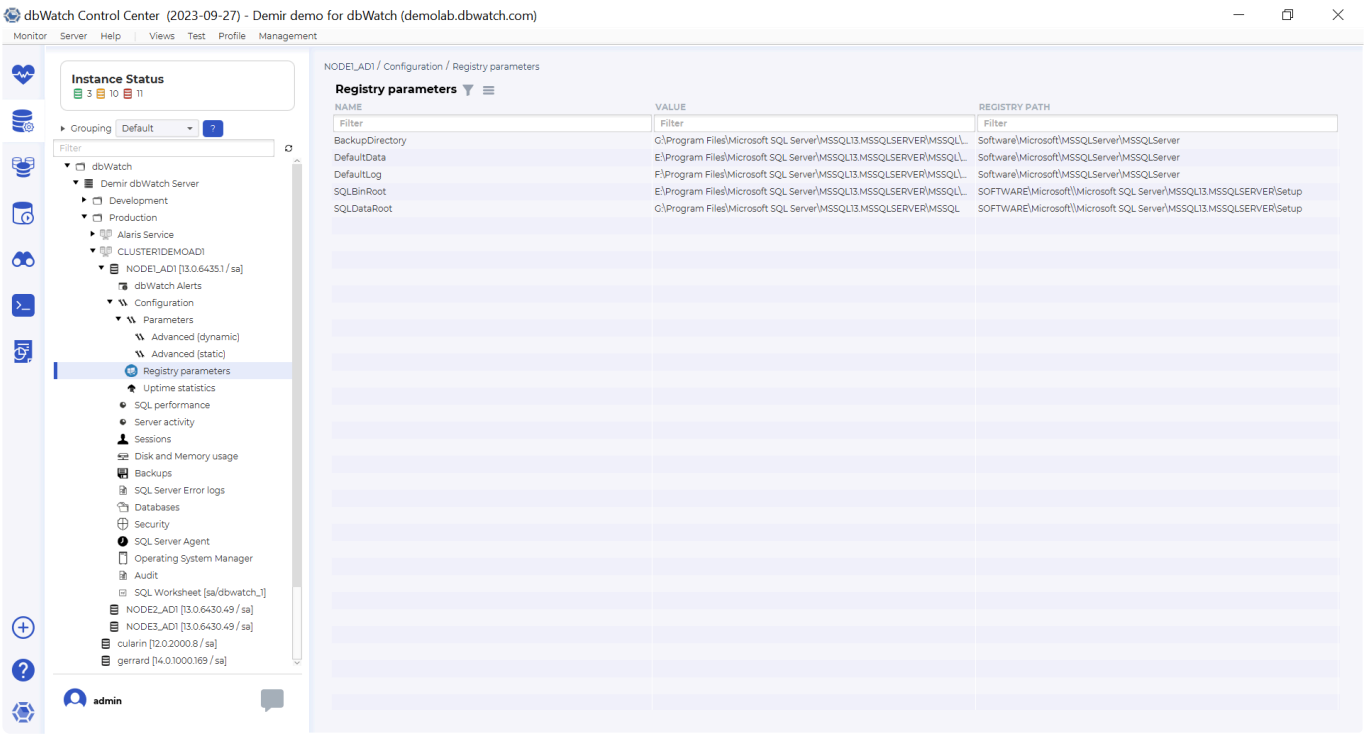
Advanced static parameters will need an instance restart to take effect. You can configure these by right-clicking on the parameter and choose configure.



<< Advanced dynamic parameters / Registry parameters >>

Registry parameters

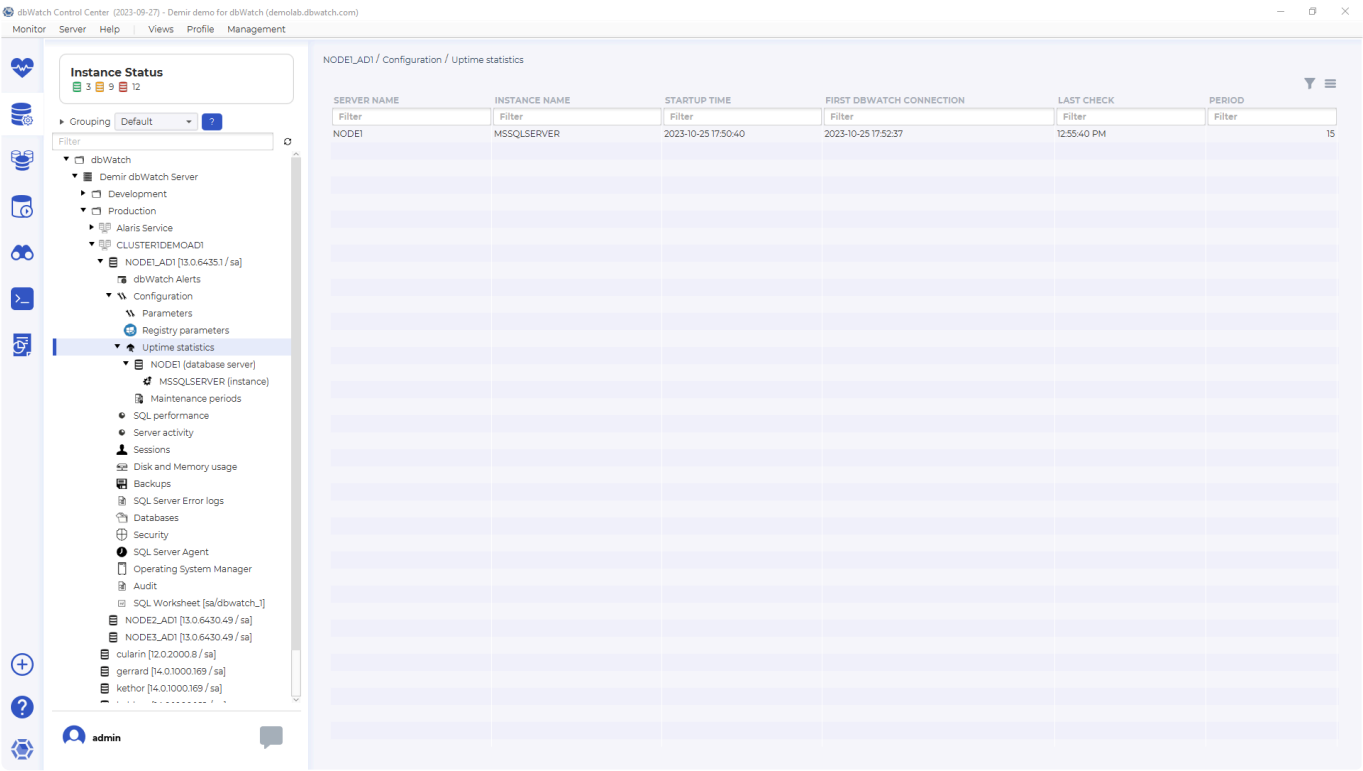
Registry parameters are primarily directory defaults that are saved in the registry. You can change these by right-clicking and choosing configure.



<< Advanced static parameters / Uptime statistics >>

Uptime statistics

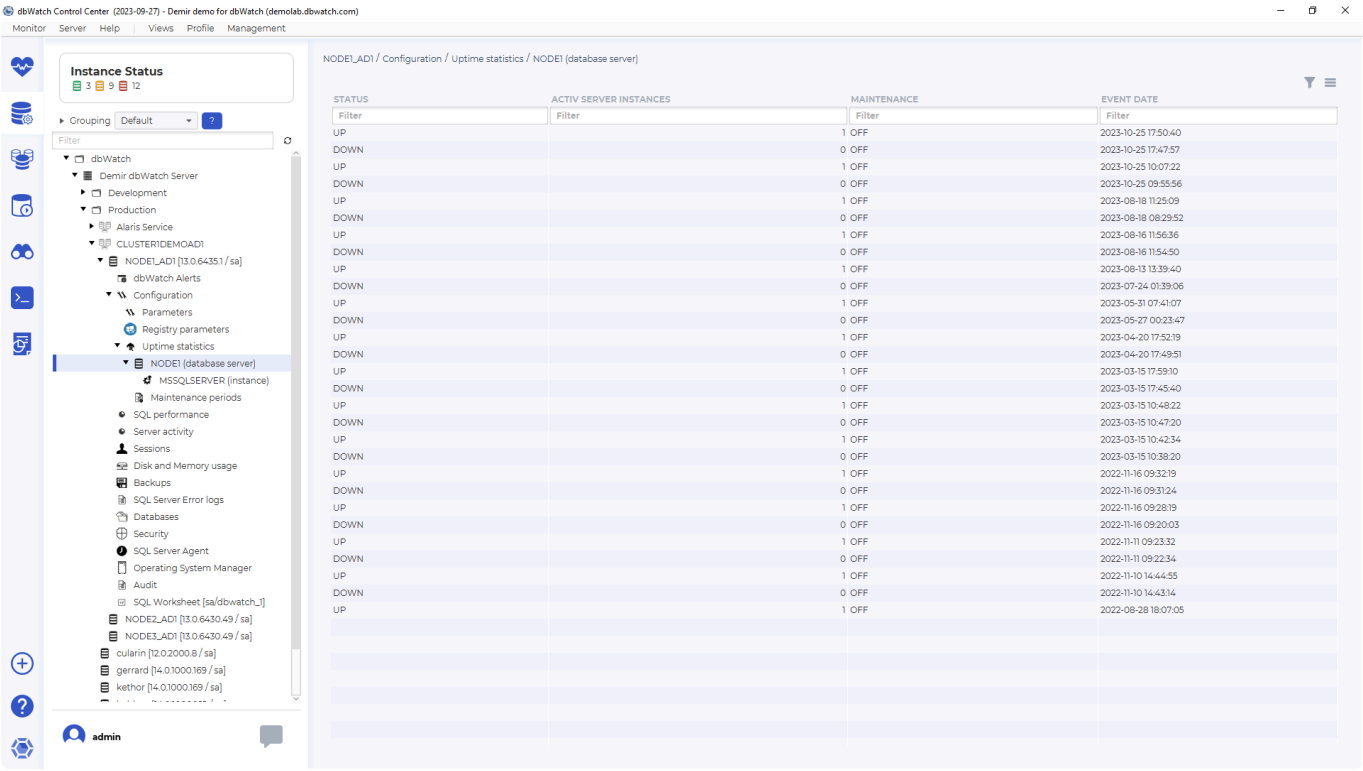
Control Center provides uptime statistics reports, and this is the interface to look at uptime periods for an instance and add scheduled maintenance periods that will be taken into consideration when uptime statistics is generated for this report.



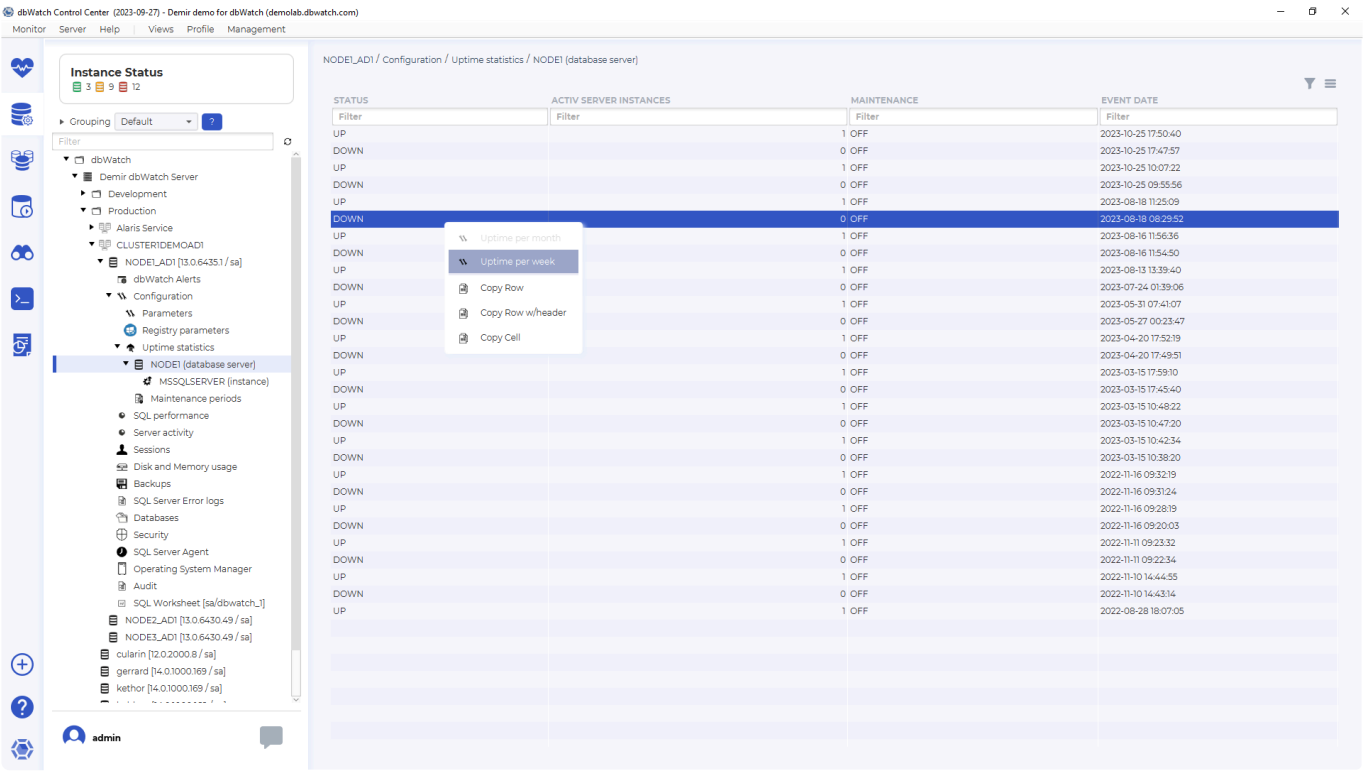
<< Registry parameters / (database server) >>

[Database name] (database server)

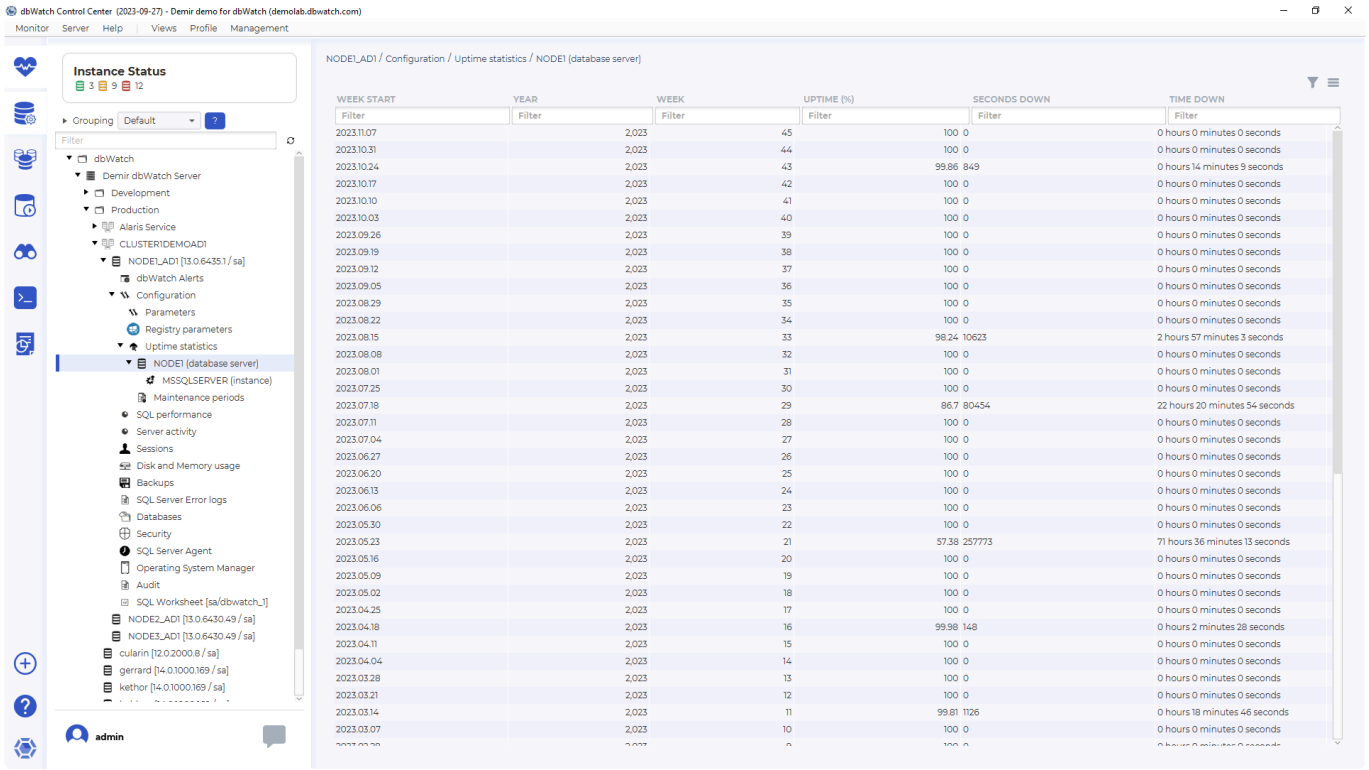
This gives an overview of up and down periods for the database server.



You can right-click on the screen to switch between different ways to display uptime:



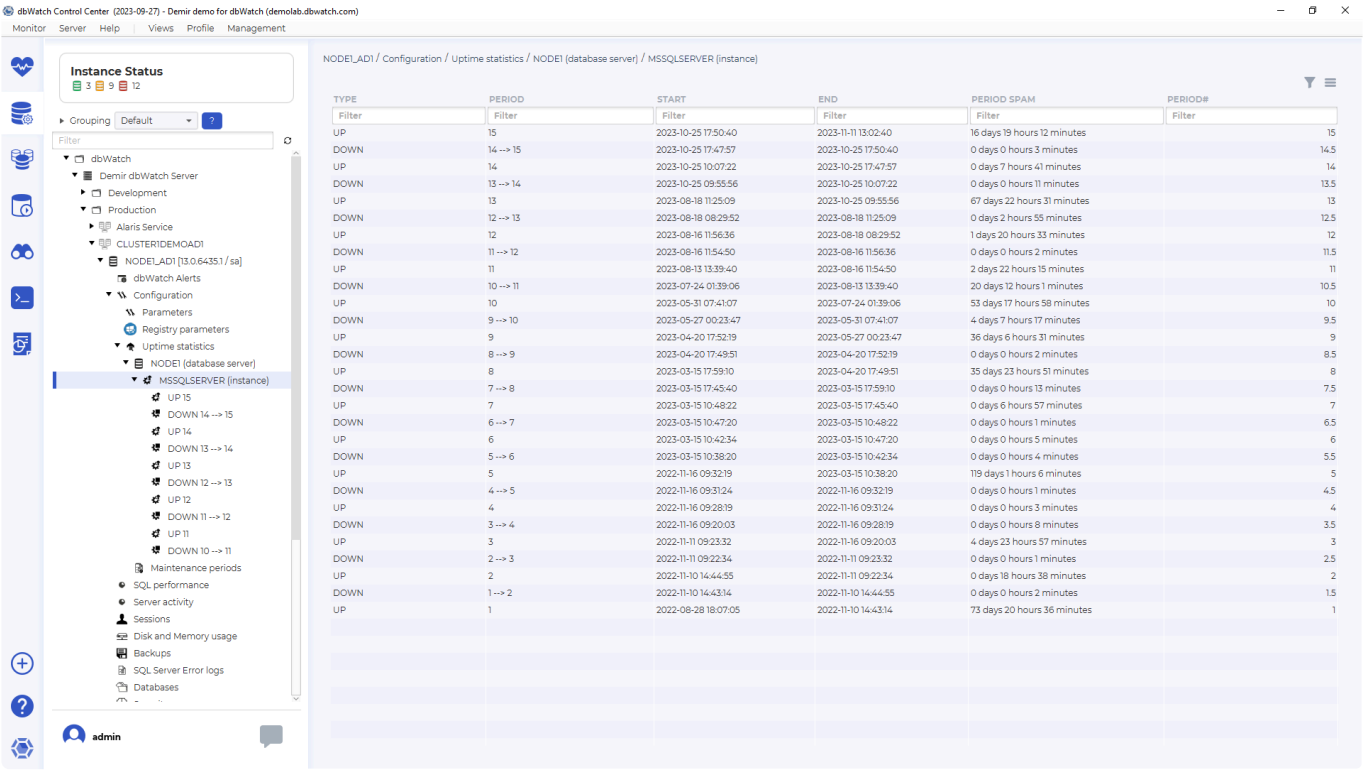
Like uptime per week:



<< Uptime statistics / (instance) >>

[Instance name] (instance)

This gives an overview of when the database instance has been up or down.

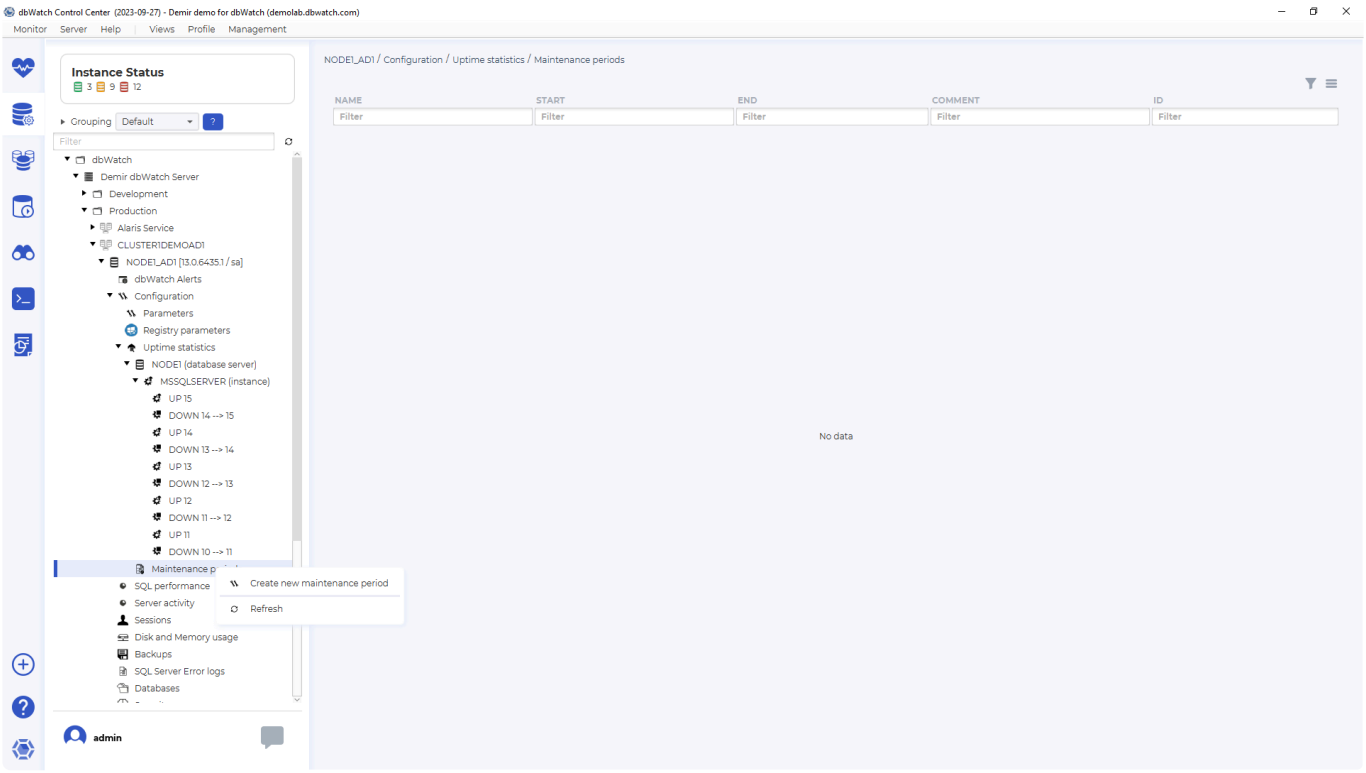


<< [Database name] (database server) / Maintenance periods >>

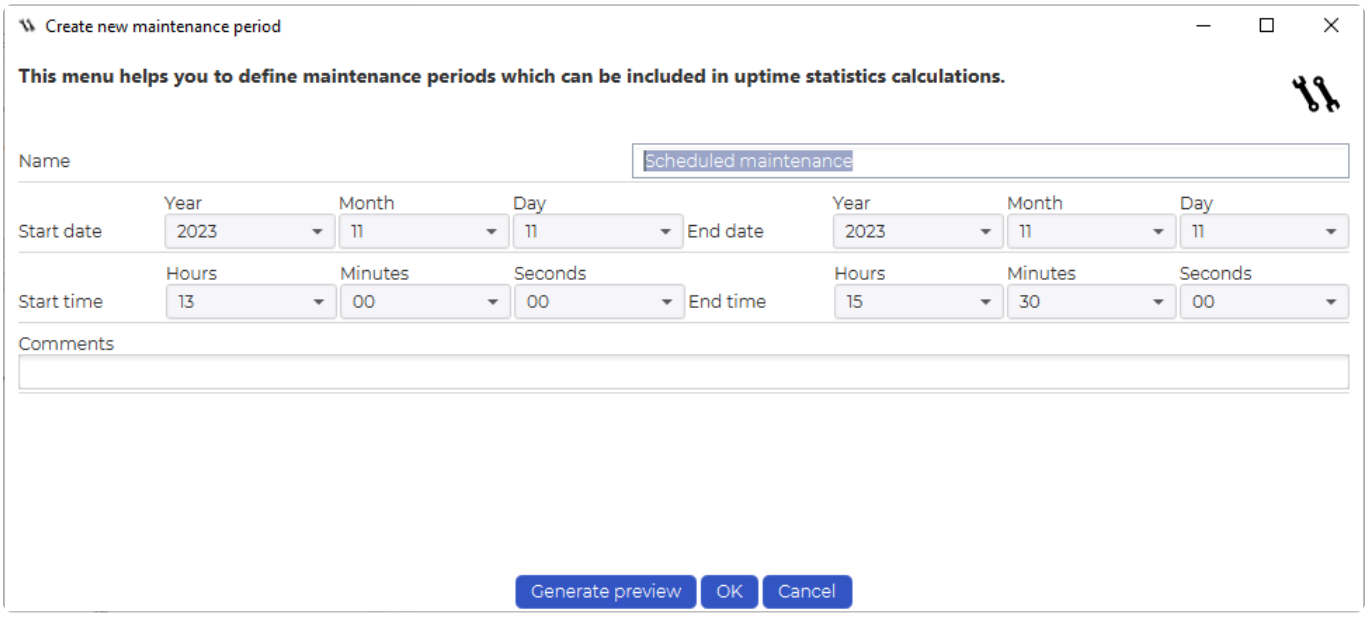
Maintenance periods

This is where you can add maintenance periods to correct the uptime report for known scheduled maintenance periods that should not count as part of the report.

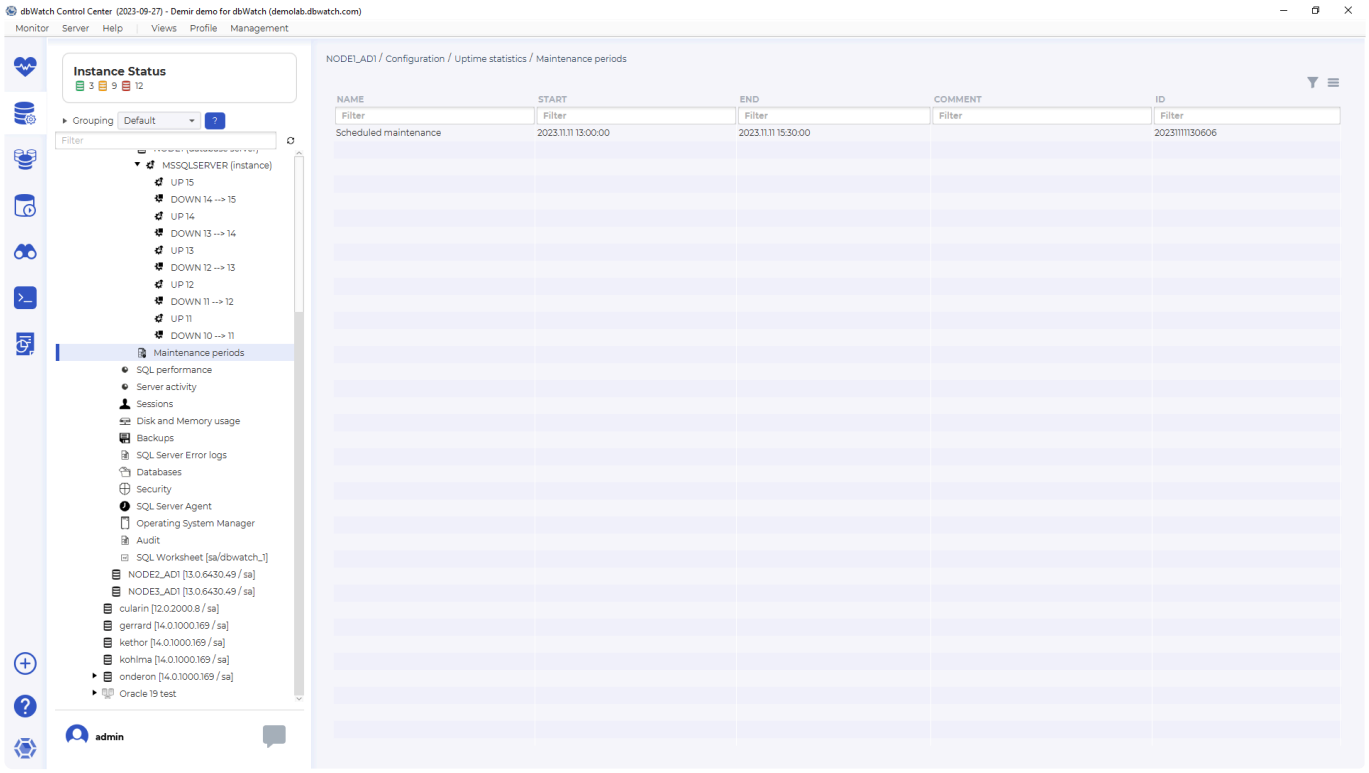
Right click on the maintenance periods to create a new known maintenance period.



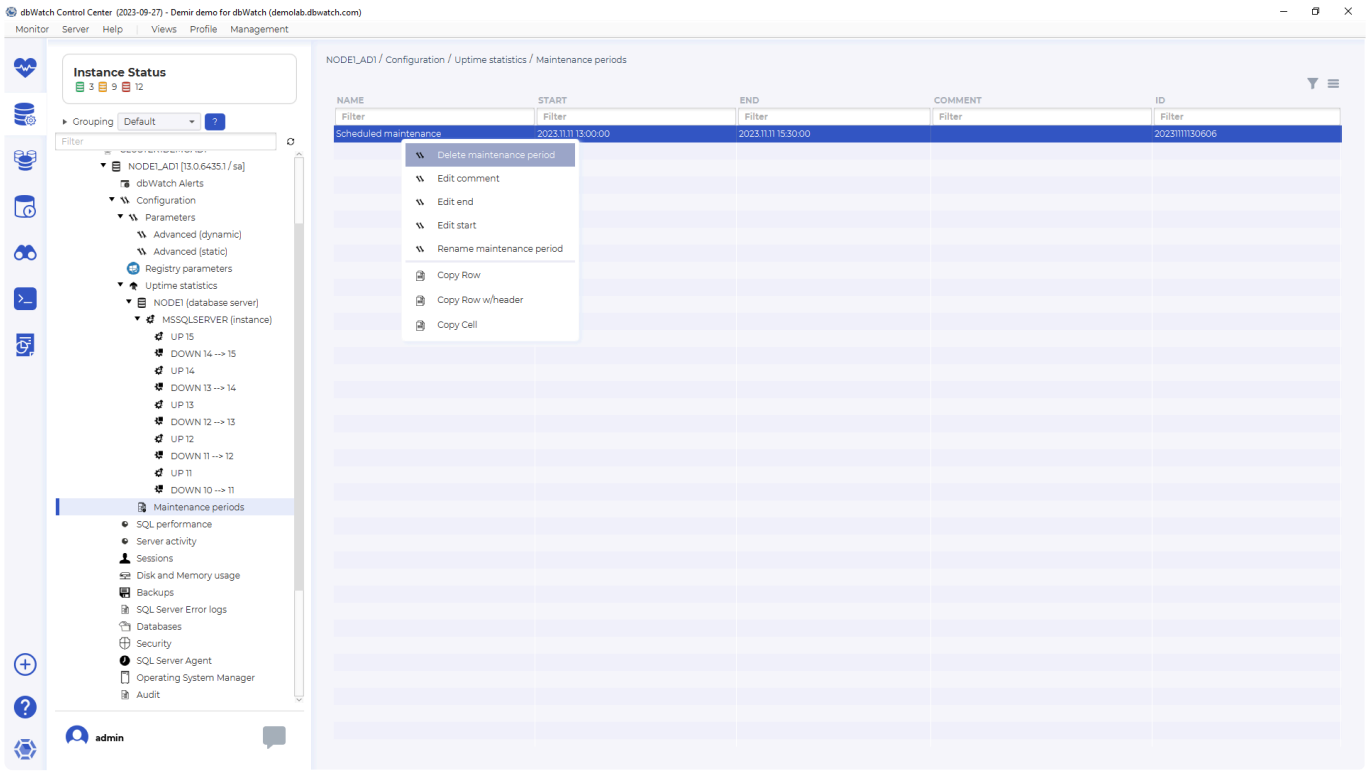
This allows you fine tune the maintenance periods with names and comments that will be used by the report.



Once you create maintenance periods the table will be populated.



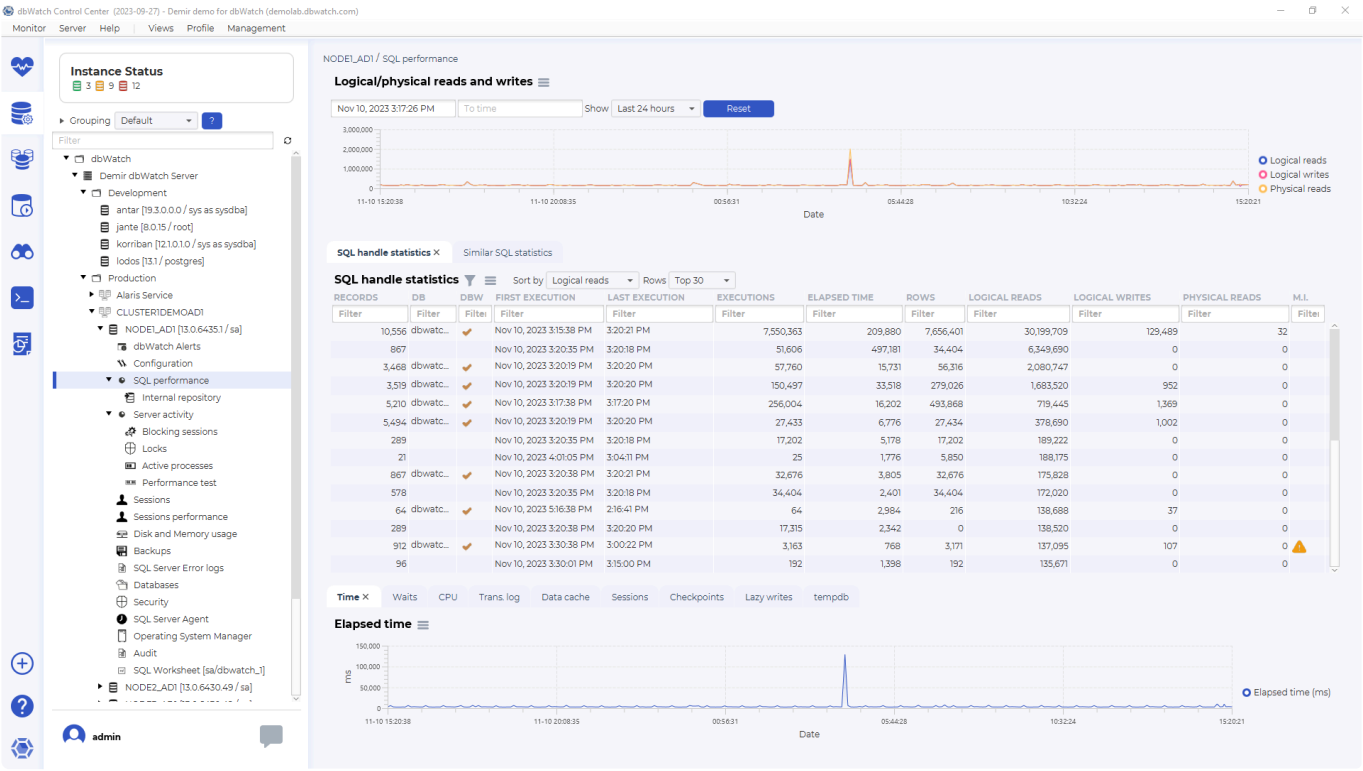
You can delete and edit the maintenance period if you right-click on any created maintenance period.



[<< [Instance name] (instance)](#) / [SQL performance >>](#)

SQL performance

SQL performance is an extra cost module, that might not be available if your license has not enabled it.

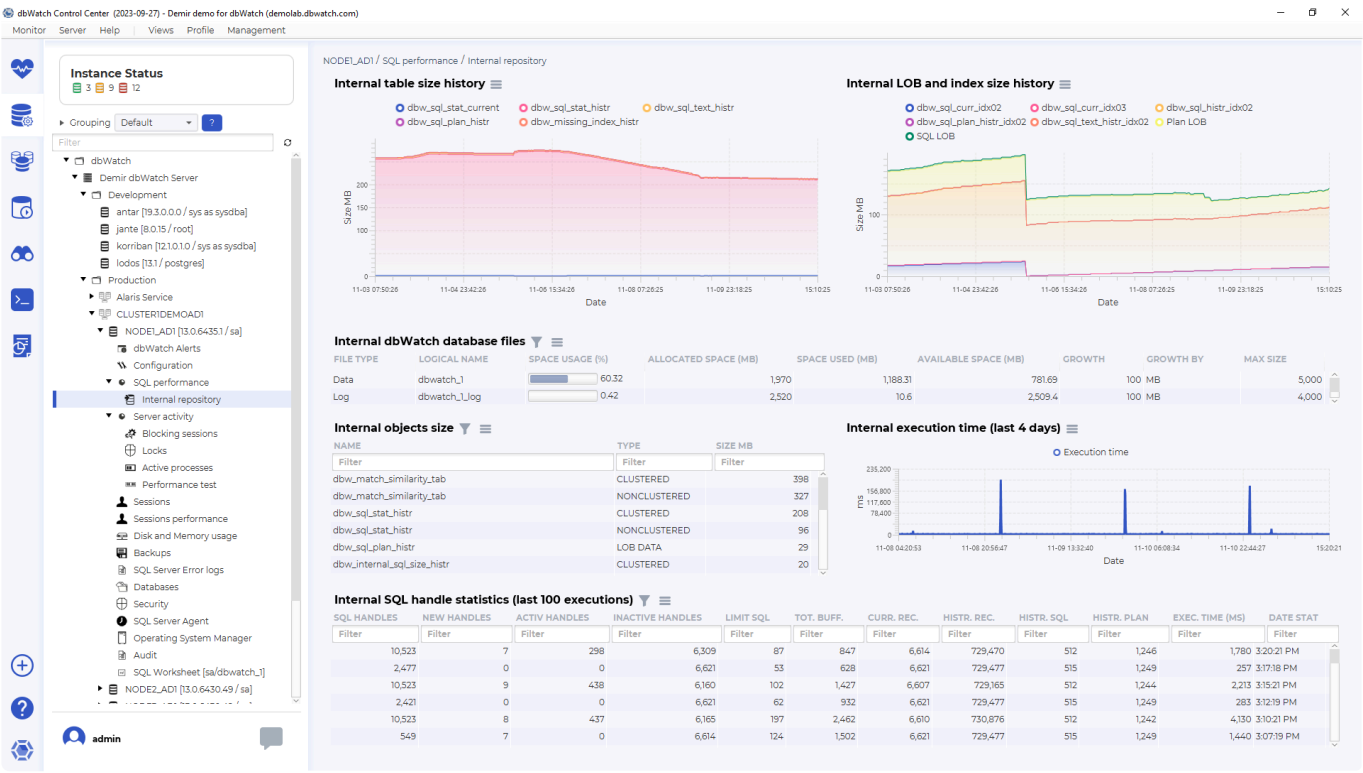


For more details look at the documentation for this module here

<< Maintenance periods / Internal repository >>

Internal repository

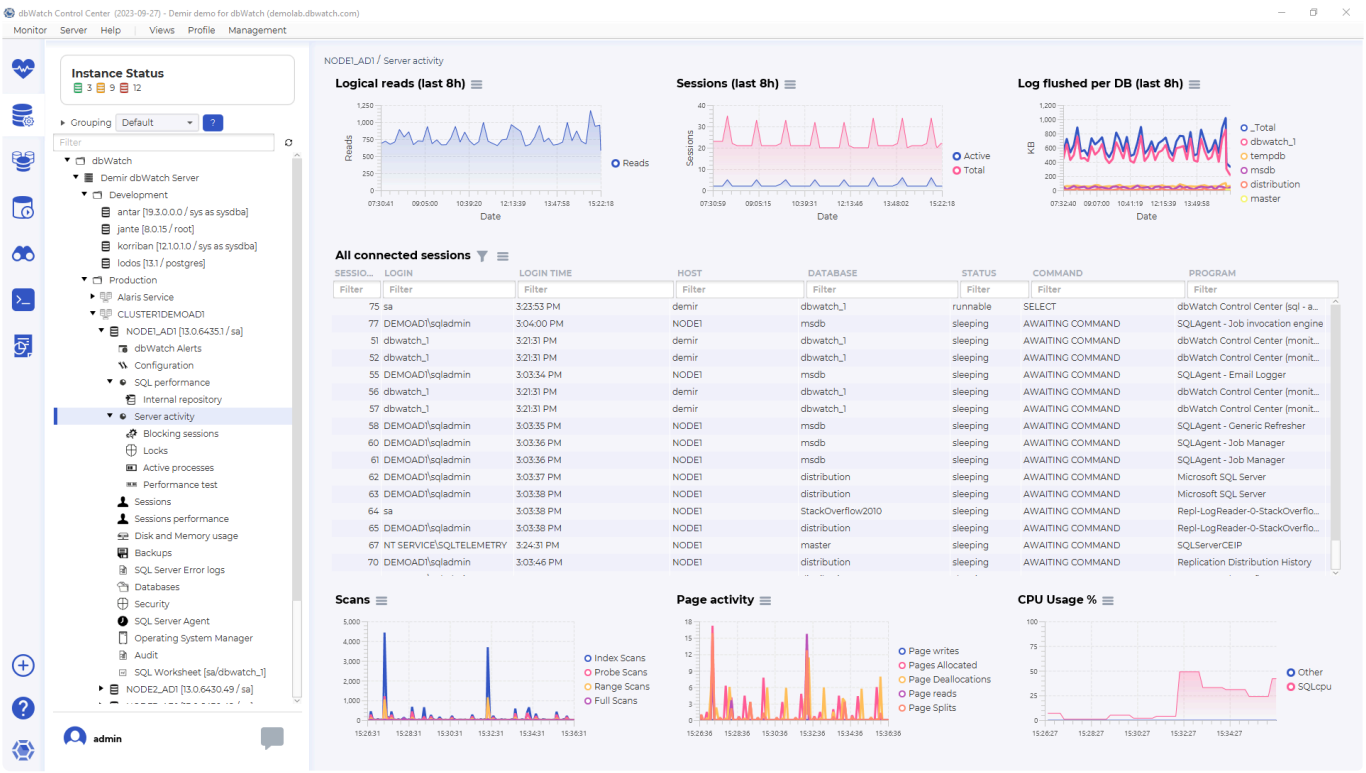
The internal repository is a dashboard to see how the gathering process of SQL performance data is working, finetune and analyze load and statistics.



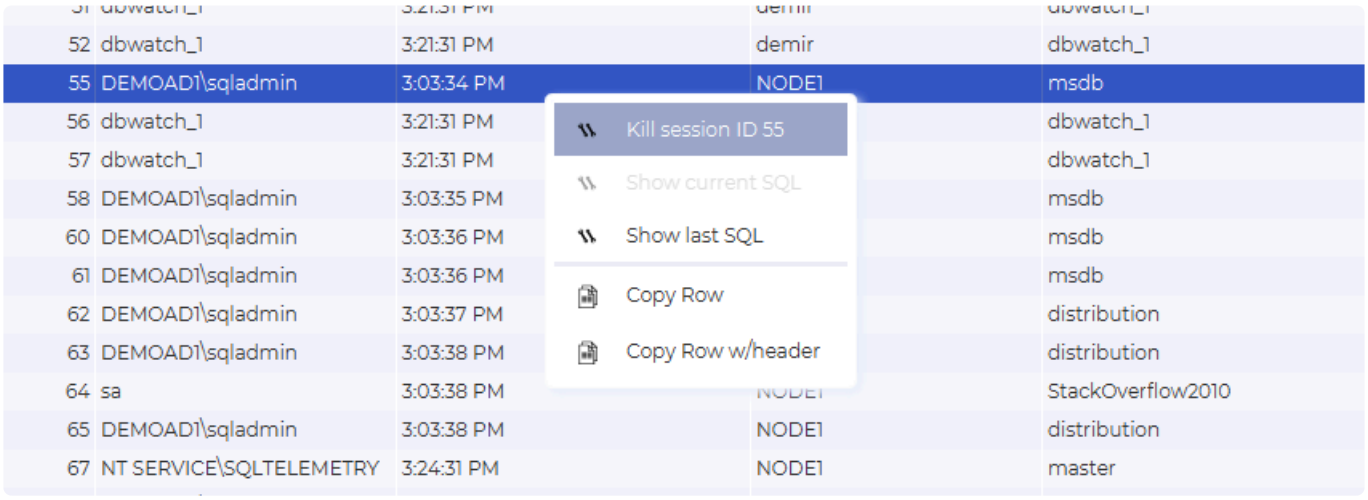
<< SQL performance / Server activity >>

Server activity

Server activity is a dashboard to get an overview of current sessions and statistics from multiple performance counters viewed together.



There is also a menu to kill sessions or view last sql statements.



<< Internal repository / Blocking sessions >>

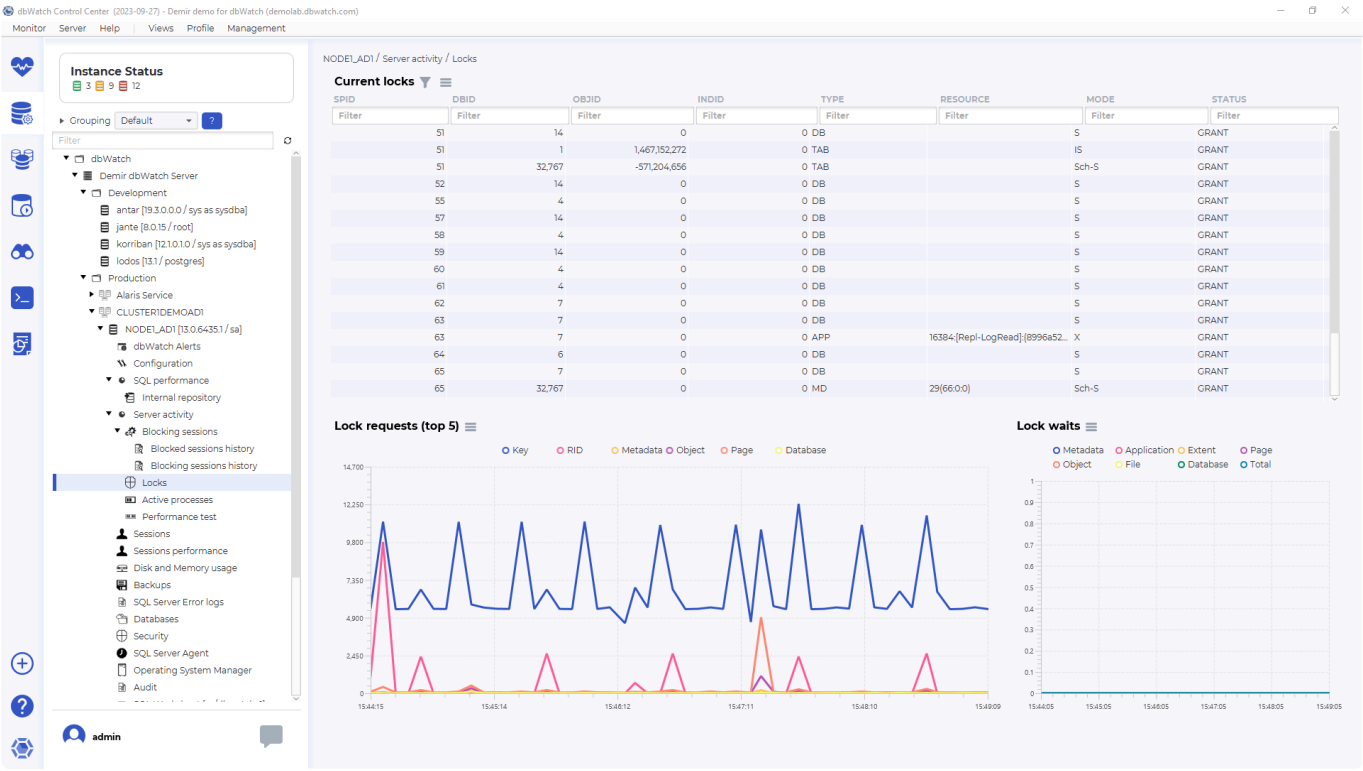
Blocking sessions

Blocked sessions history

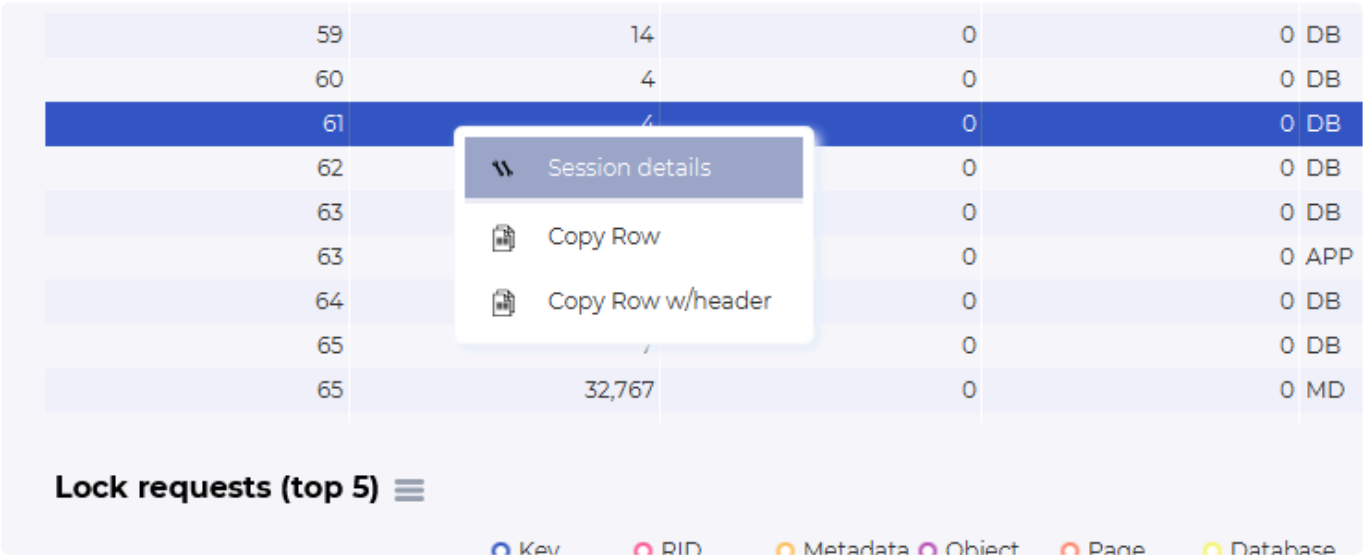
Blocking sessions history

Locks

Locks view current locks in the database and statistics.



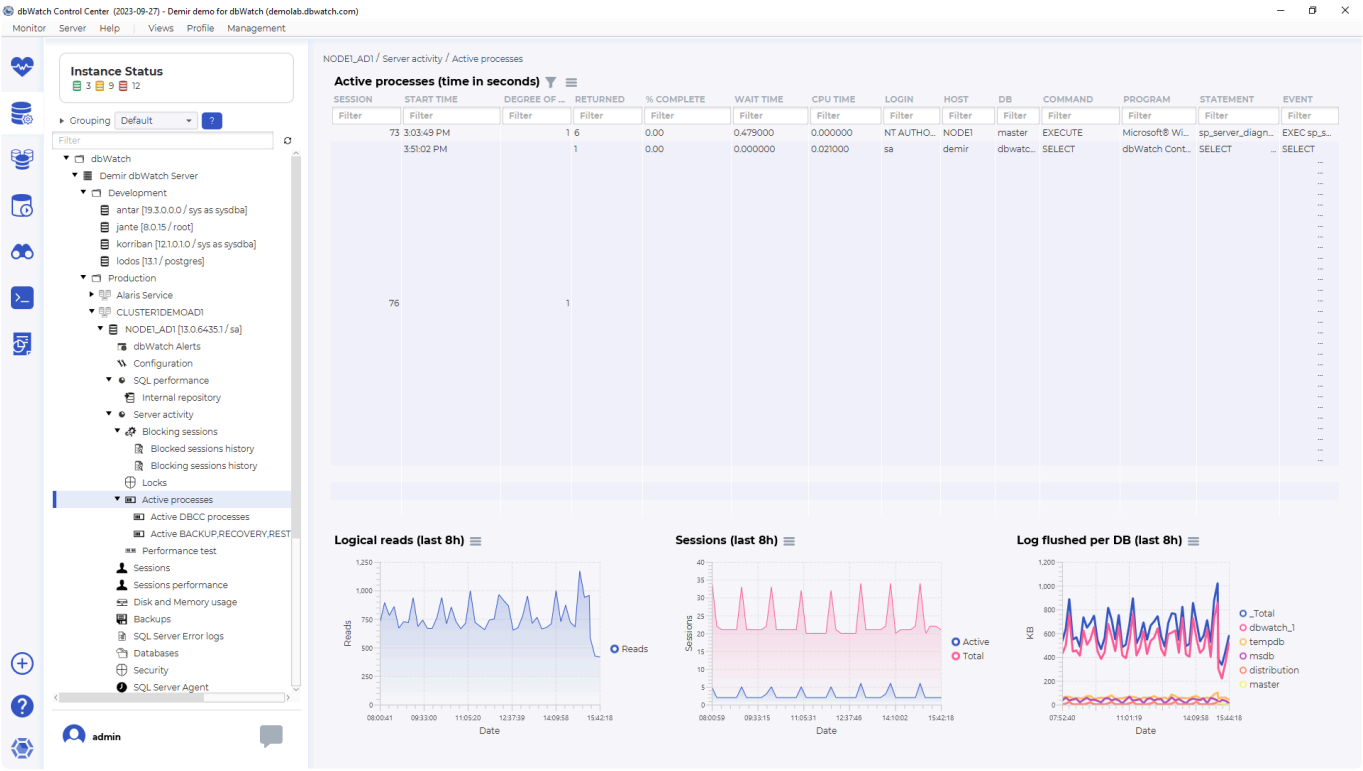
There is also a menu to get more session details.



<< Blocking sessions history / Active processes >>

Active processes

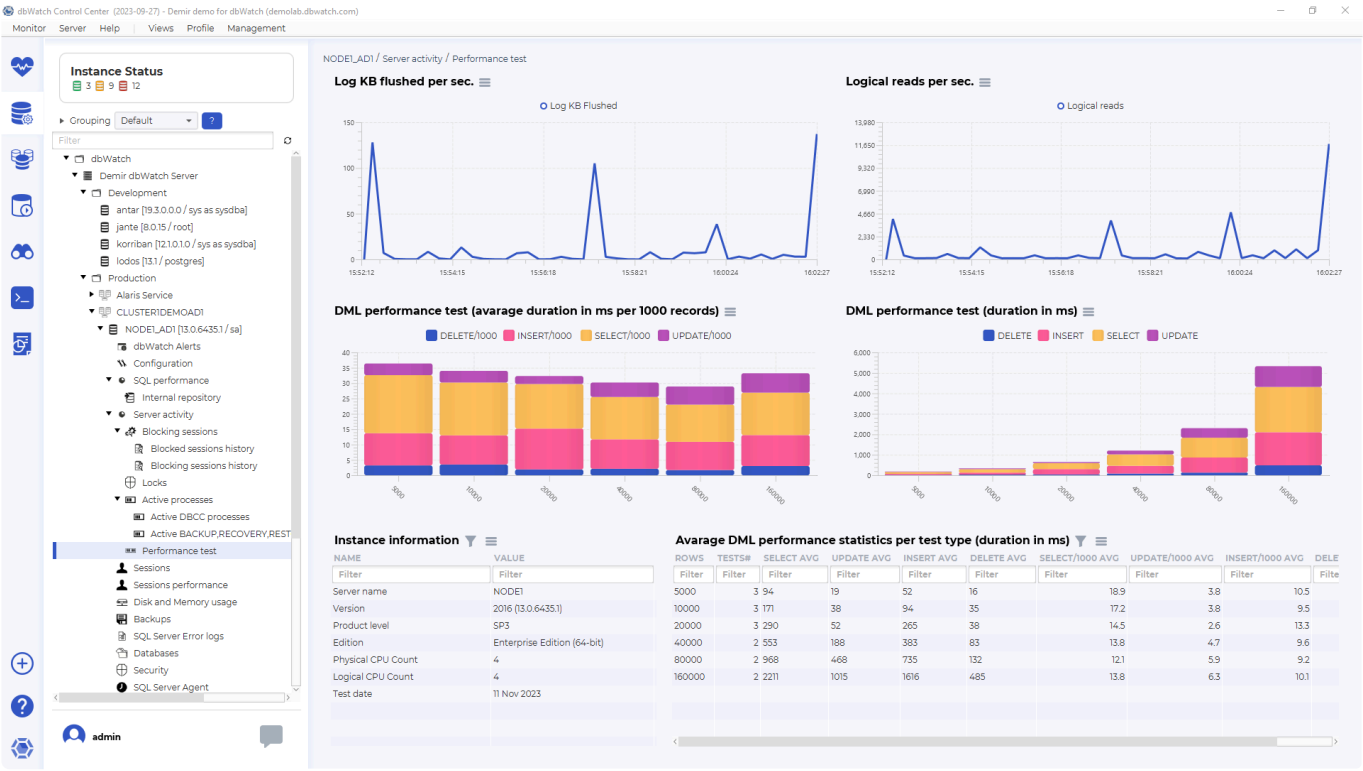
Active processes are the current running SQL statements along with statistics.



<< Locks / Performance test >>

Performance test

Performance test displays statistics gathered from the DML performance test job that runs a performance test on the database instance.

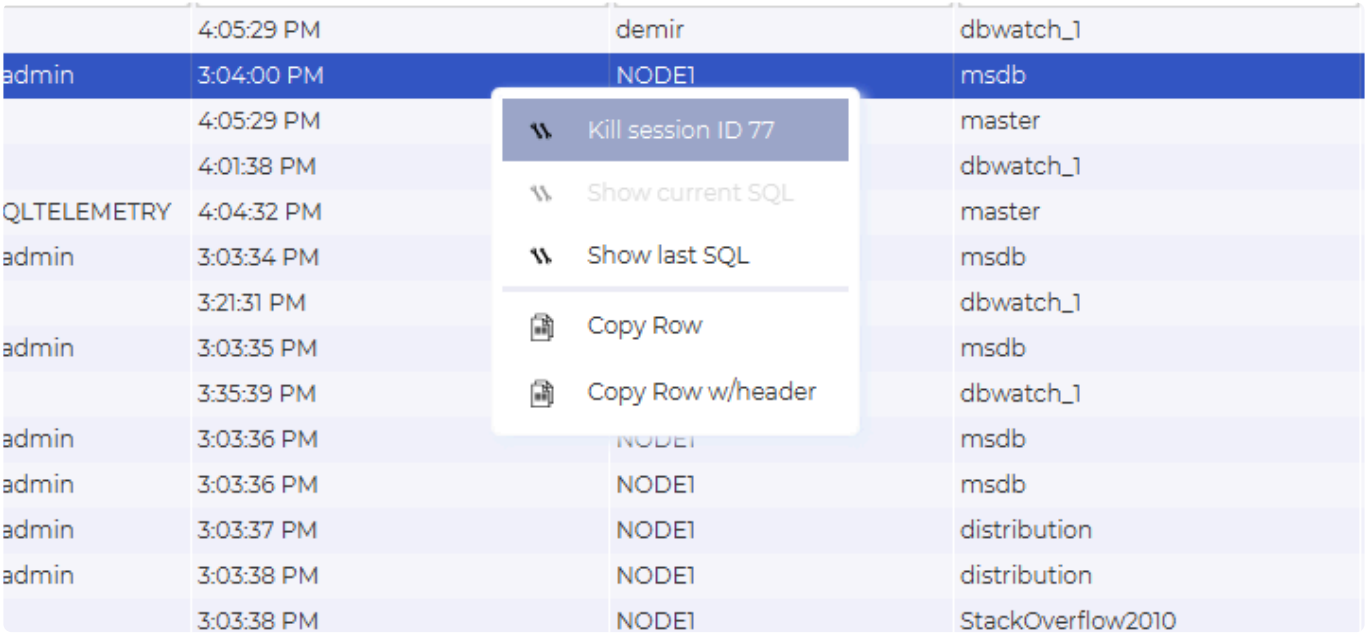


<< Active processes / Sessions >>

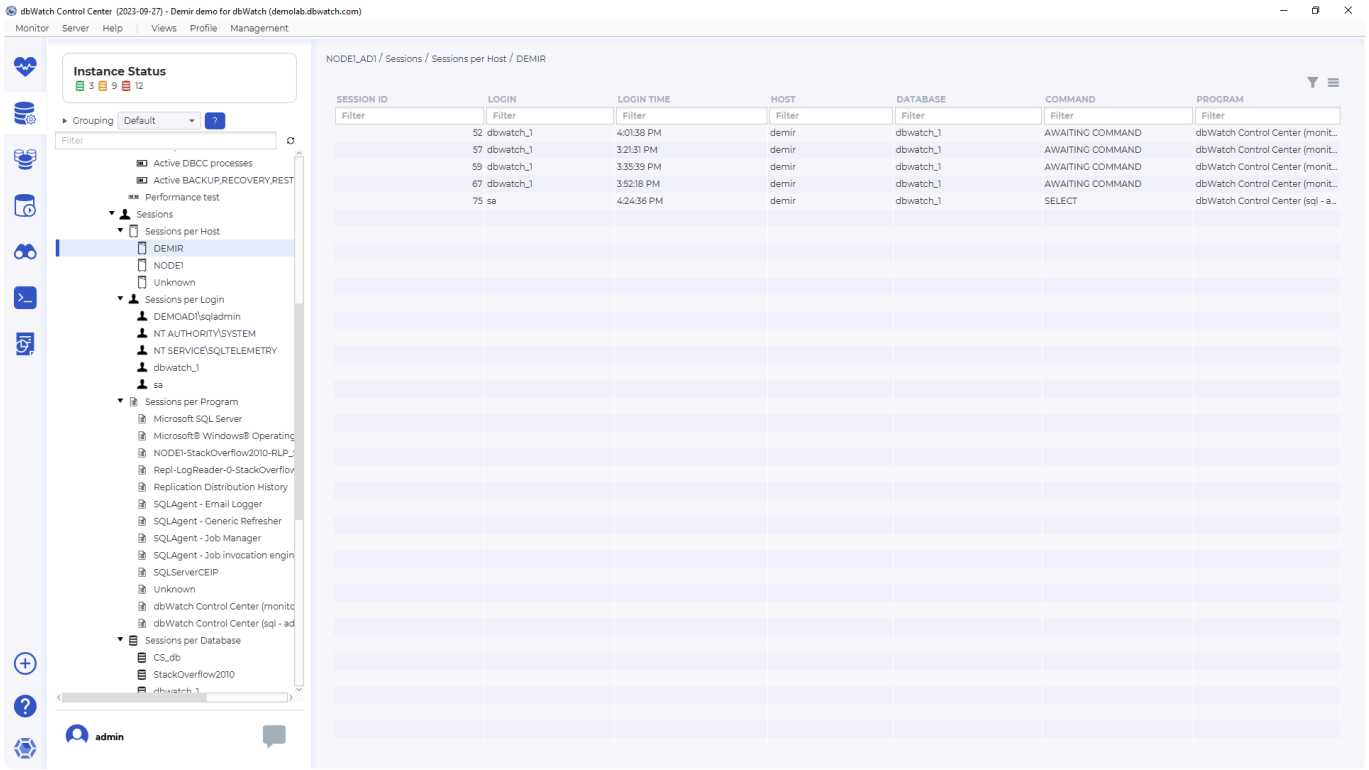
Sessions

Sessions gives an overview of the current logged in sessions on the system.

Right-click on any line in the table allows you to kill the session or see what the last SQL is.



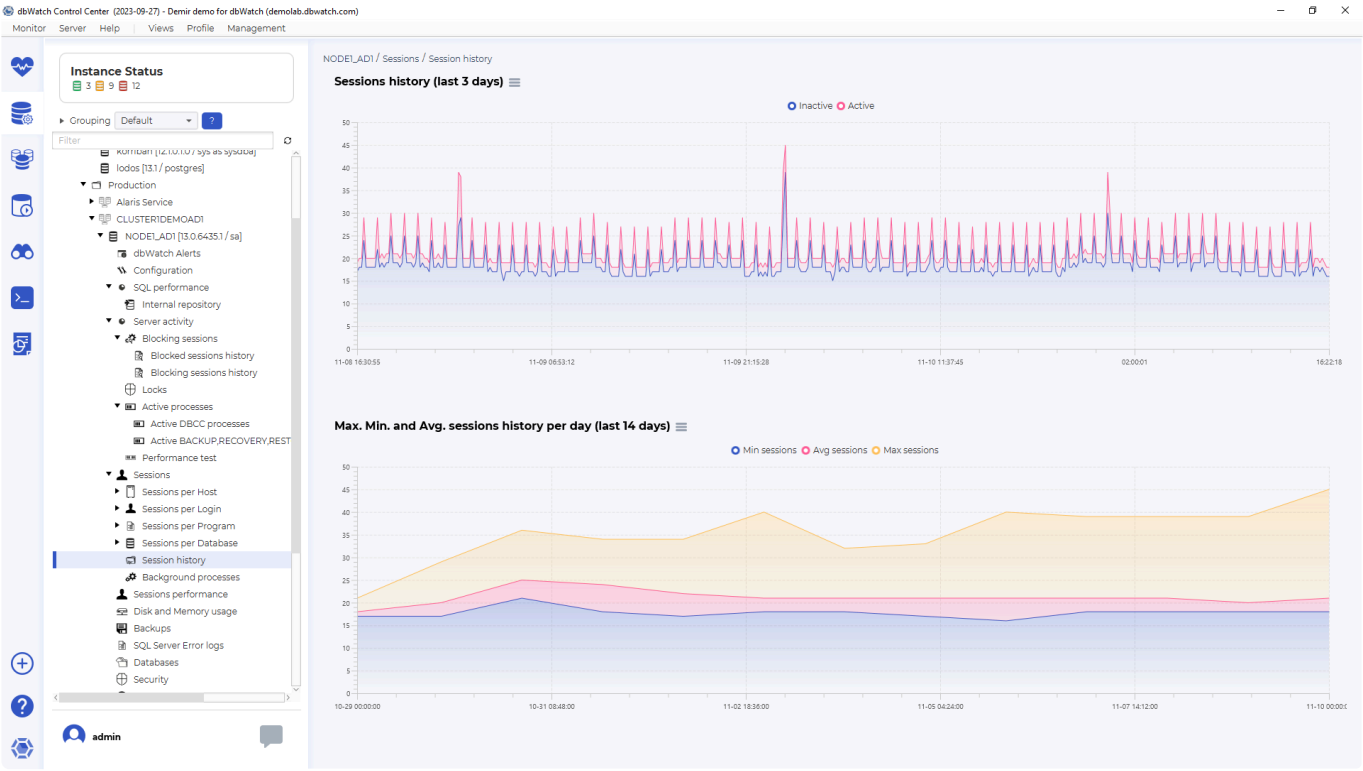
You can also drill down to sessions grouped by Host, Login, Program or Database.



<< Performance test / Sessions history >>

Sessions history

This view shows historic statistics for sessions on this instance.

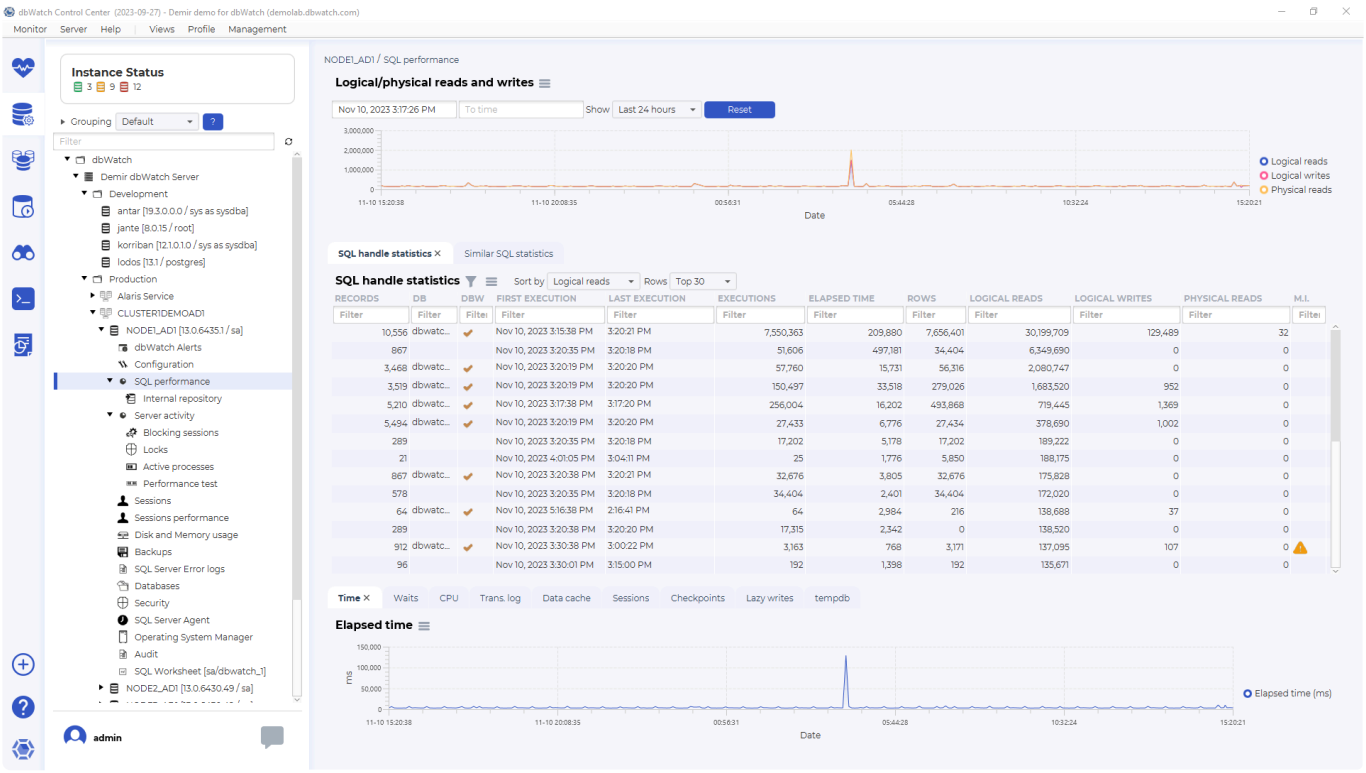


<< Sessions / Sessions performance >>

Sessions performance

The Sessions performance view is part of the SQL performance package, which is an extra cost module, that might not be available if your license has not enabled it.

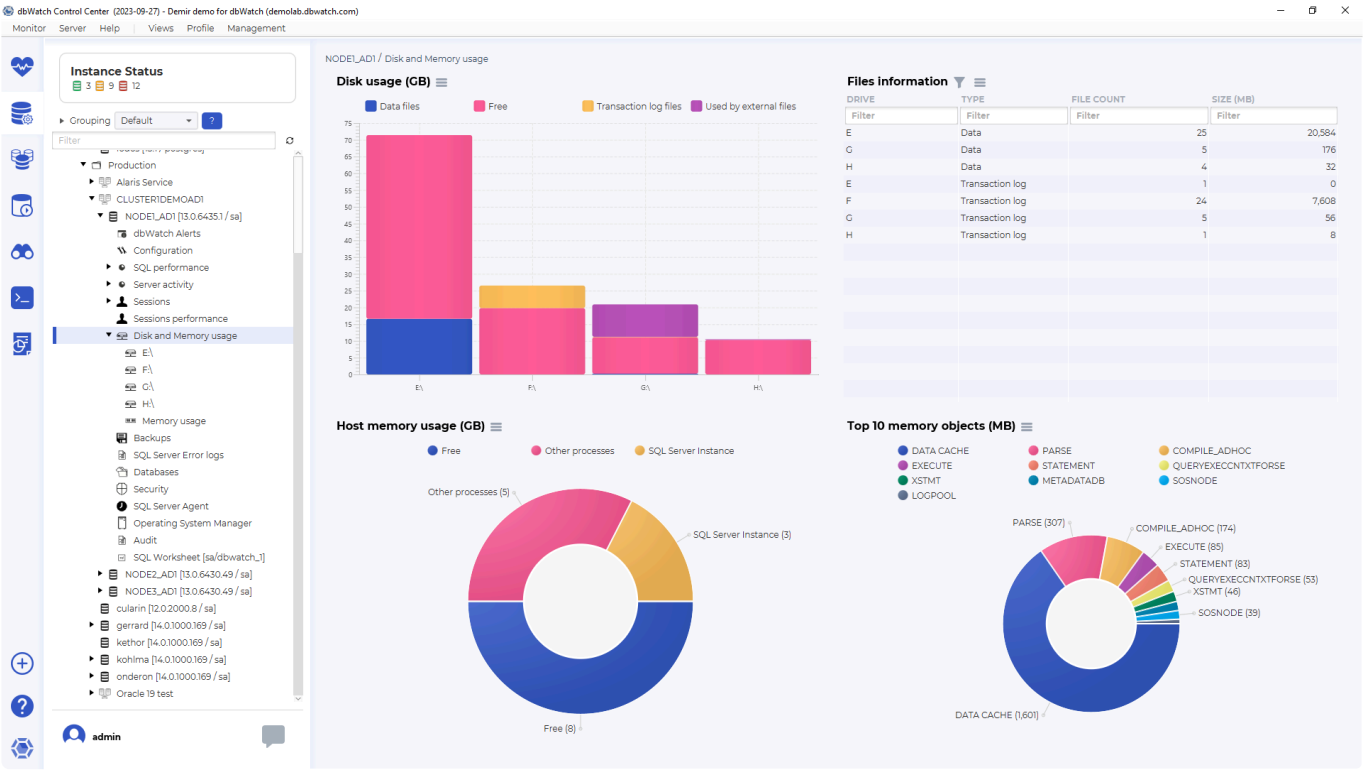
For more details look at the documentation for this module here

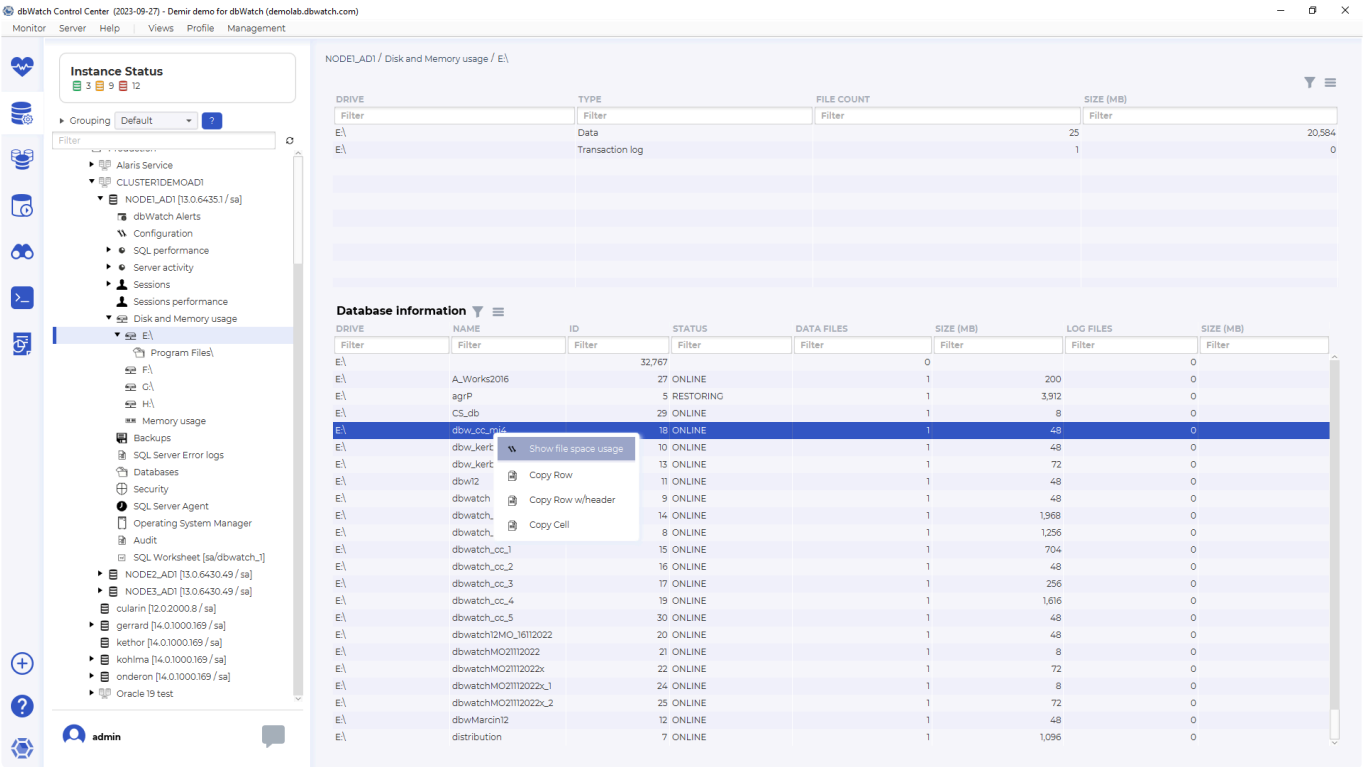


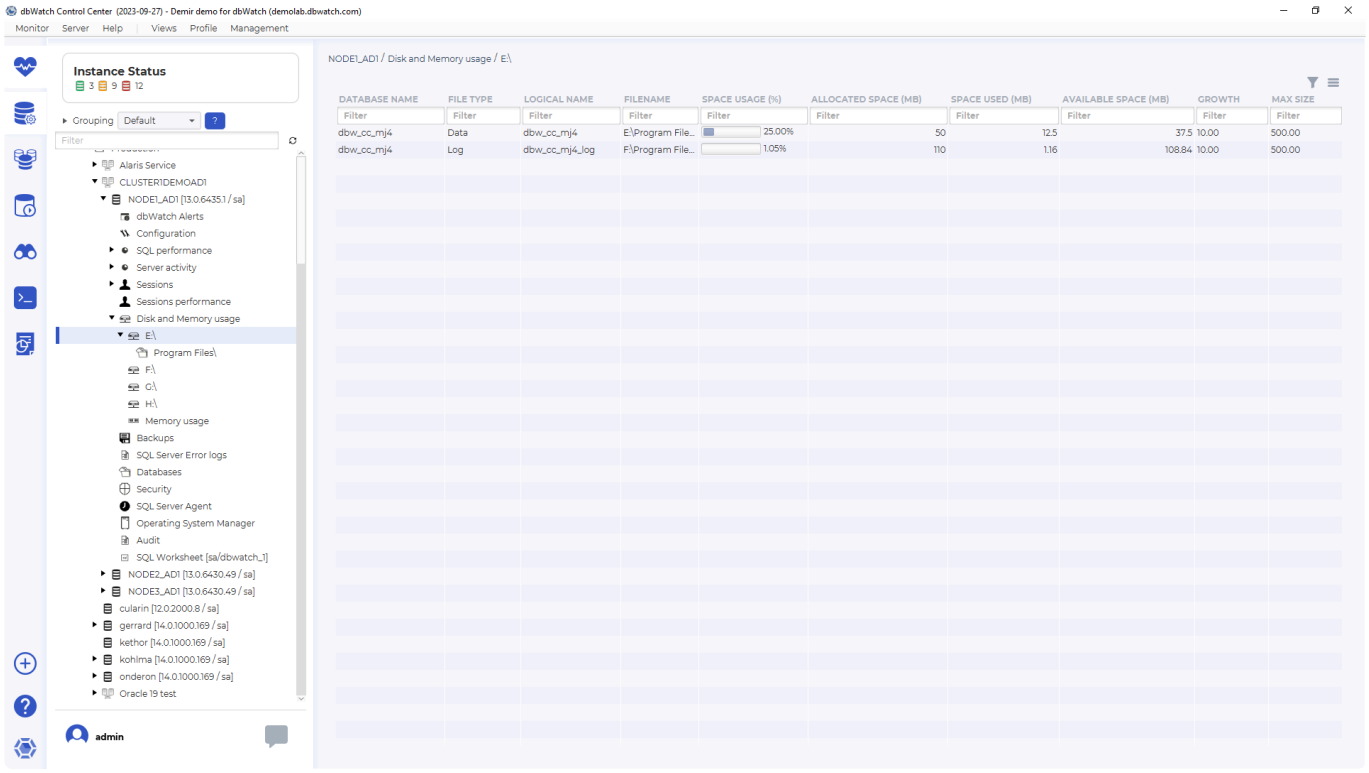
<< Sessions history / Disk and Memory usage >>

Disk and Memory usage

This allows you to get an overview and drill down into disk and memory usage on the database instance.



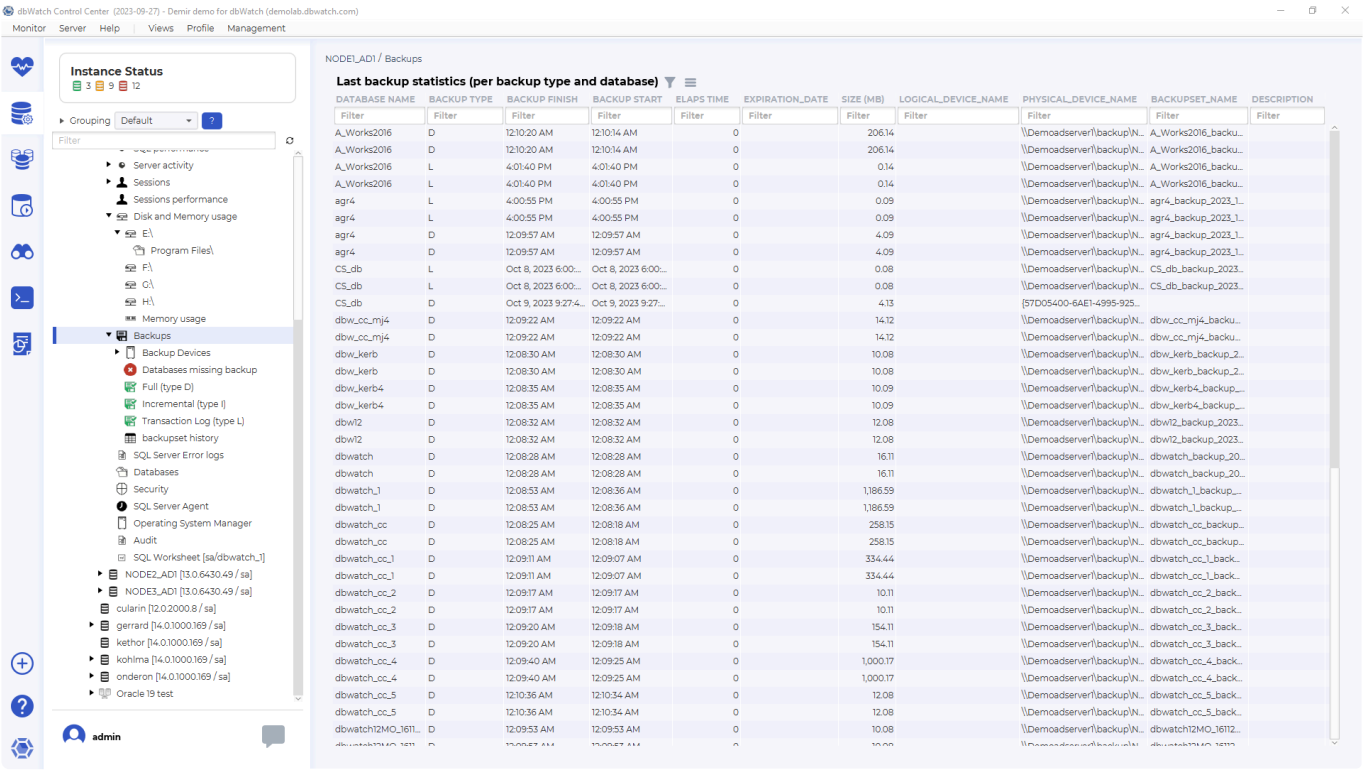




<< Sessions performance / Backups >>

Backups

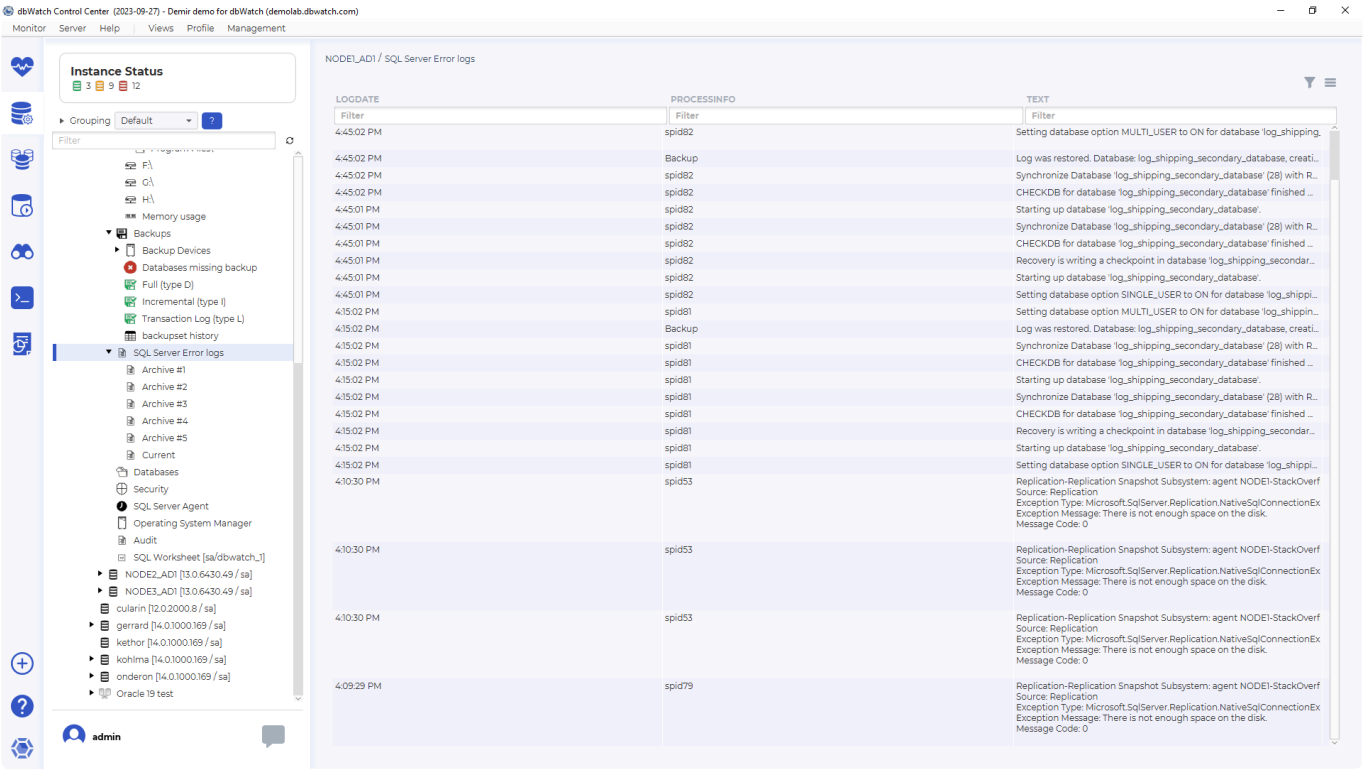
Backups is where you can drill down into any details about the backups on this SQL Server



<< Disk and Memory usage / SQL Server Error logs >>

SQL Server Error logs

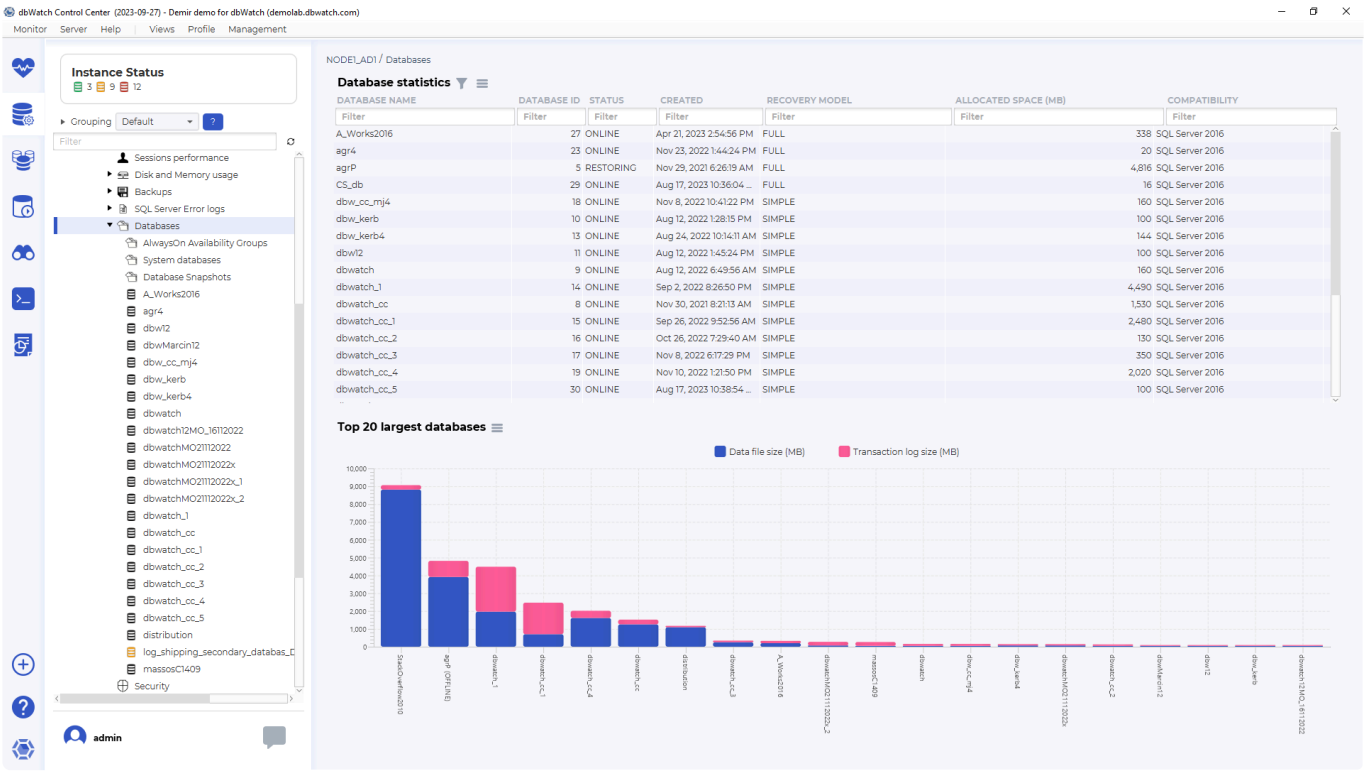
This view allows you to look at the SQL Server error logs and search and filter data.



<< Backups / Databases >>

Databases

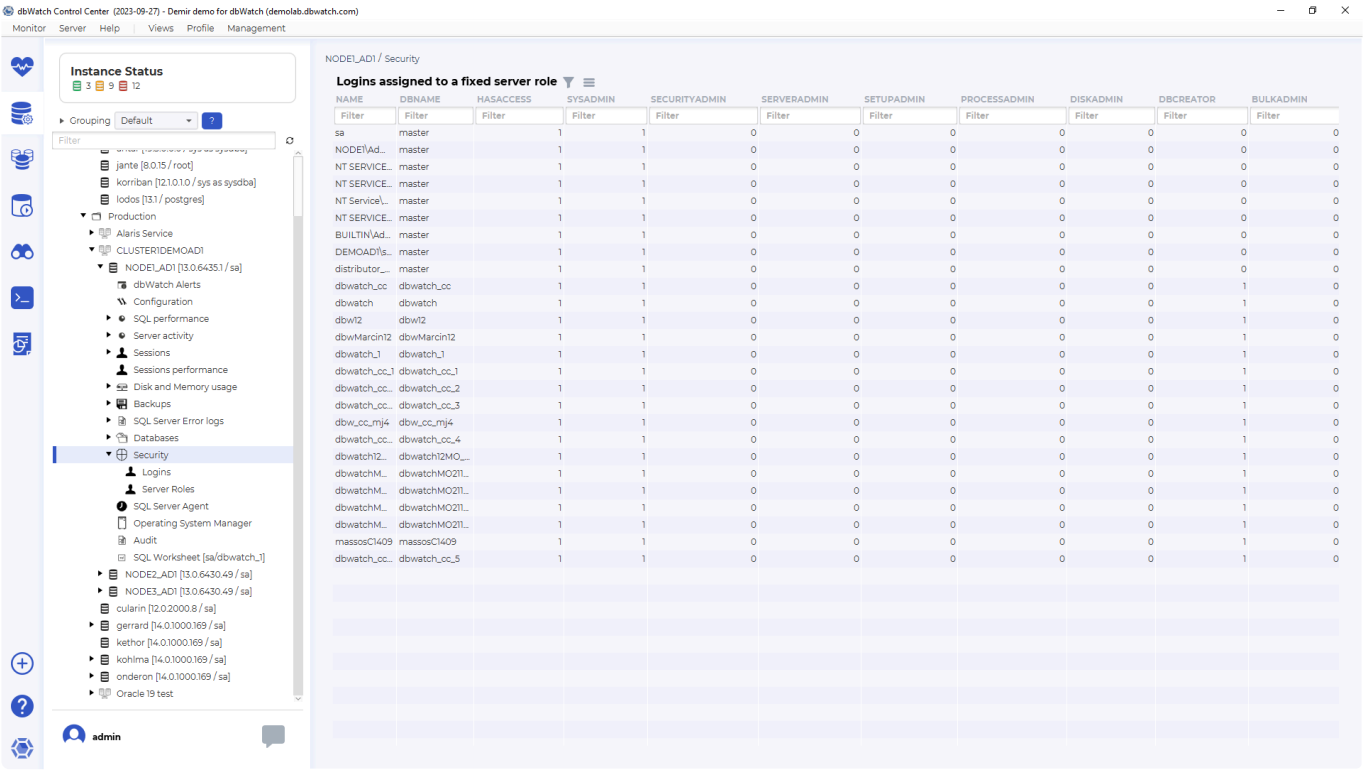
This view allows you to drill down into each database and its underlying data.



[<< SQL Server Error logs](#) / [Security >>](#)

Security

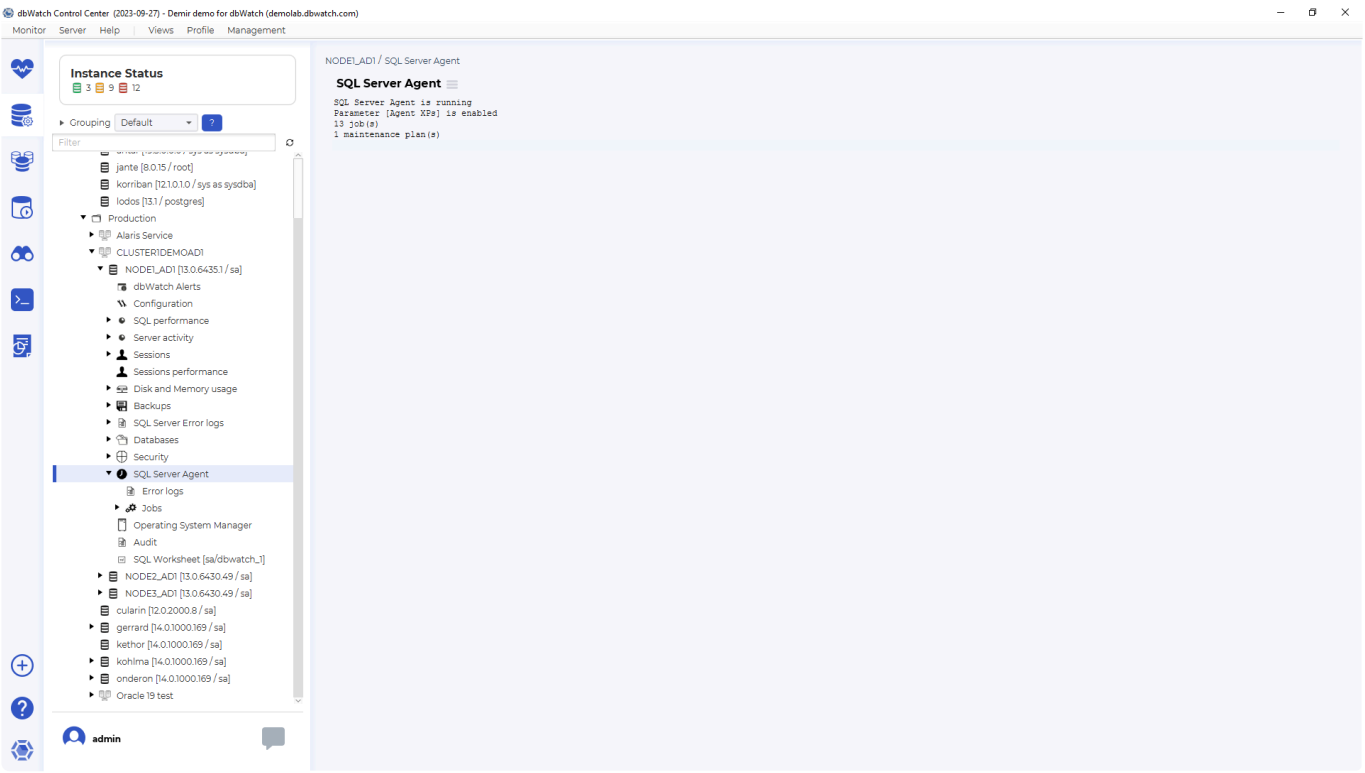
Security is where you configure users and privileges.



<< Databases / SQL Server Agent >>

SQL Server Agent

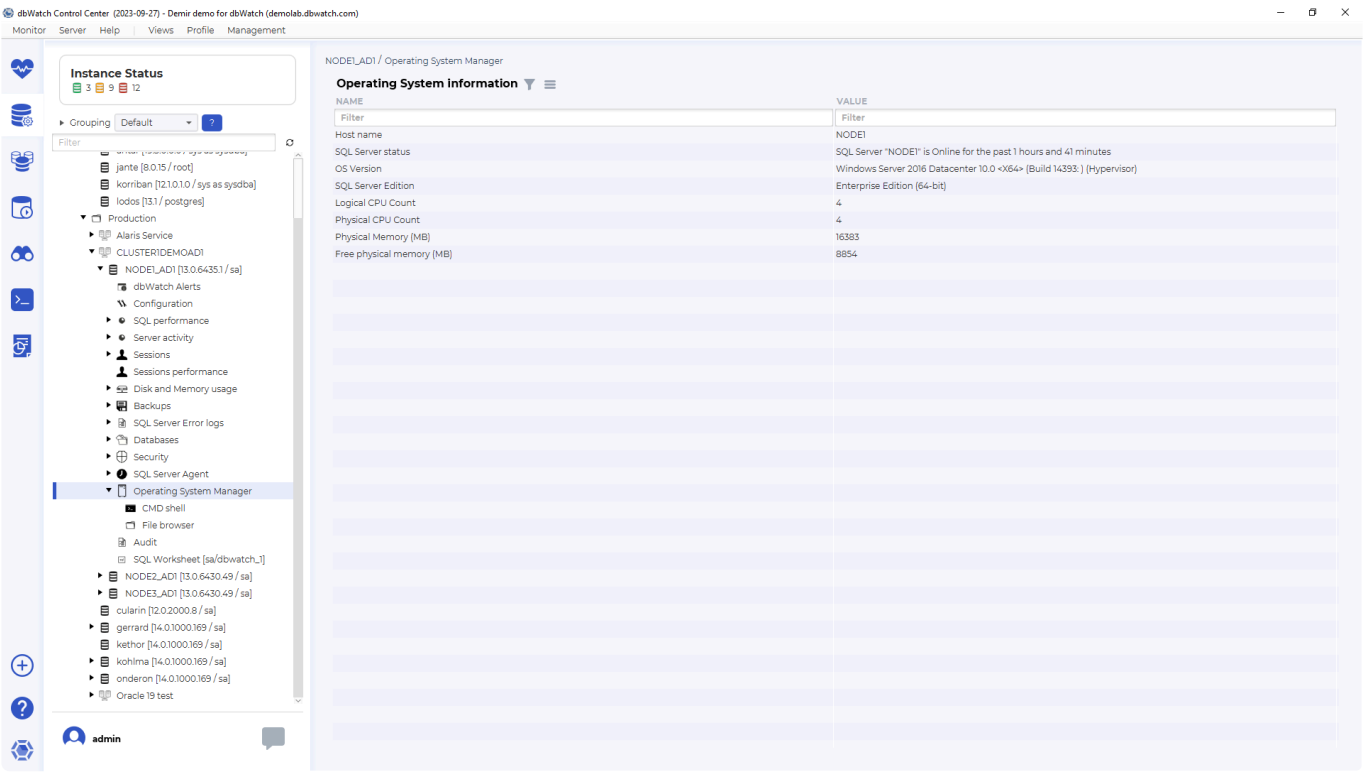
SQL Server agent allows you to check the status of the agent, look into agent jobs and its statistics, and create new agent jobs.



<< Security / Operating System Manager >>

Operating System Manager

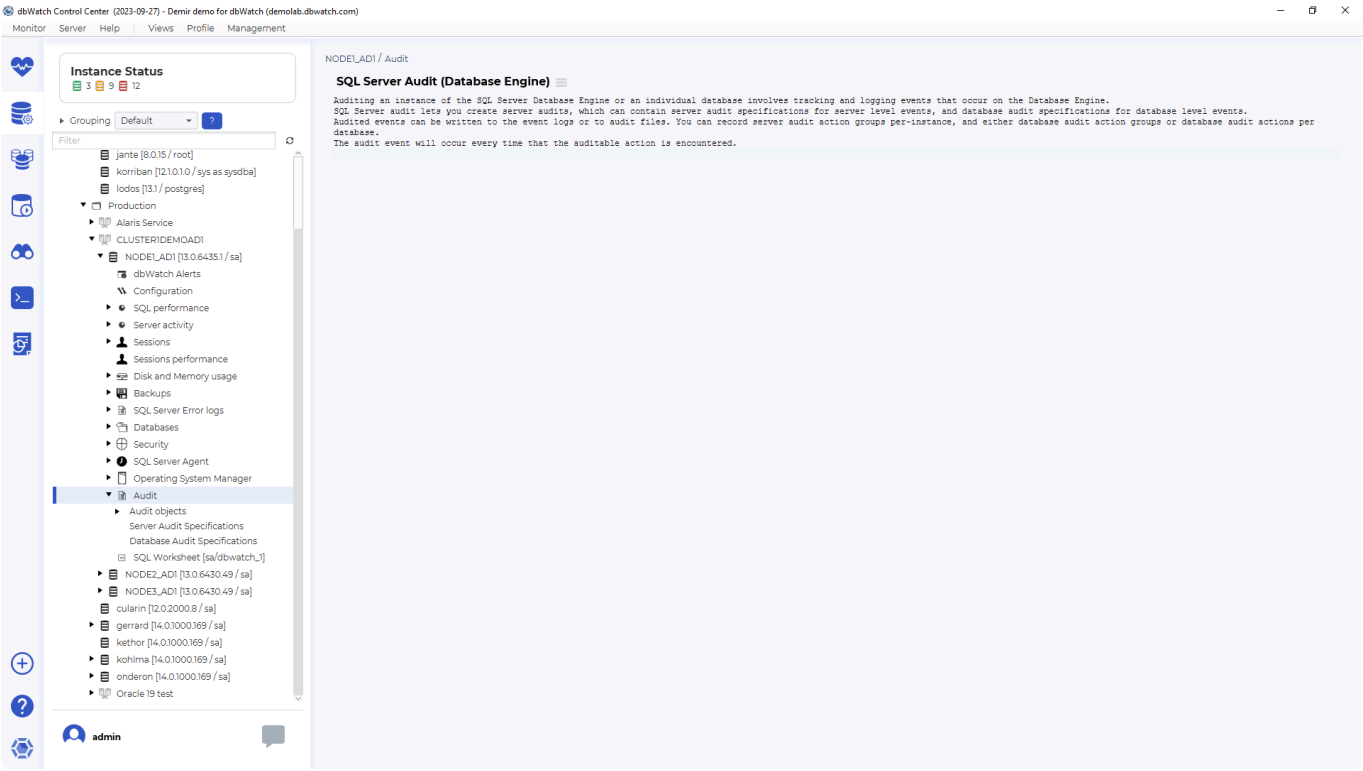
Operating System Manager allows some interaction with the OS underneath, such as a CMD shell and file browser.



<< SQL Server Agent / Audit >>

Audit

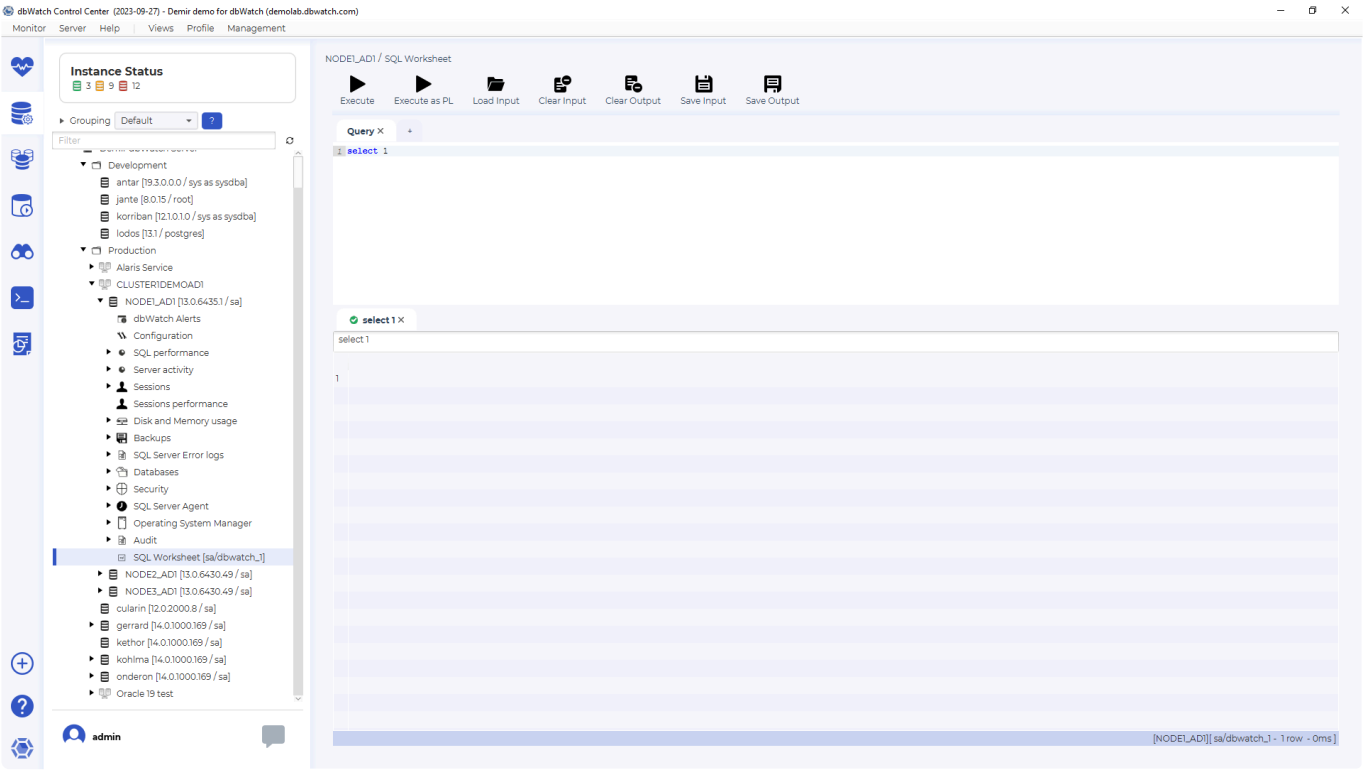
Audit is where you control what is being audited on the SQL Server



<< Operating System Manager / SQL Worksheet >>

SQL Worksheet

SQL Worksheet allows for direct SQL commands on the database instance.



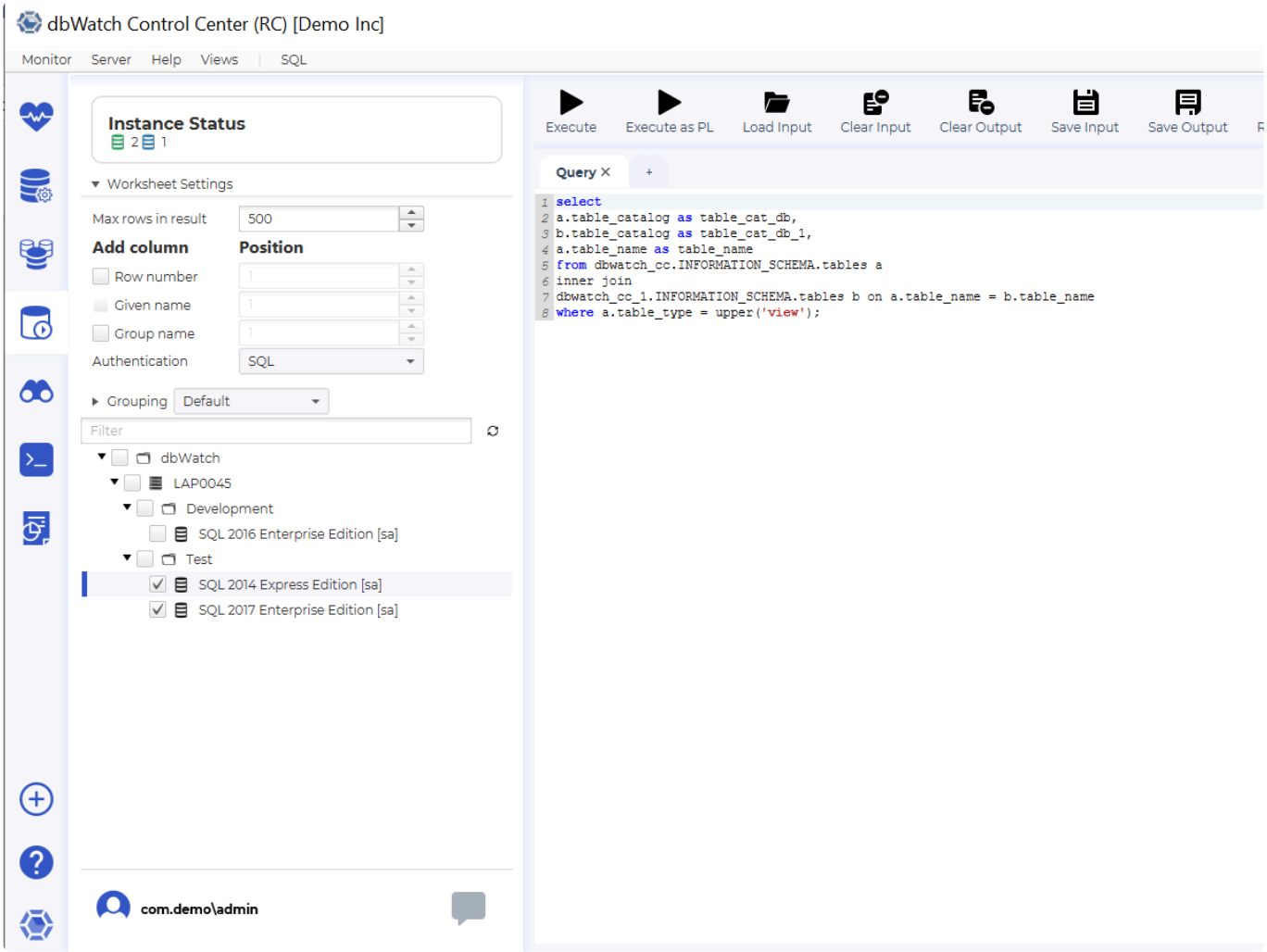
<< Audit / Worksheet >>

Worksheet

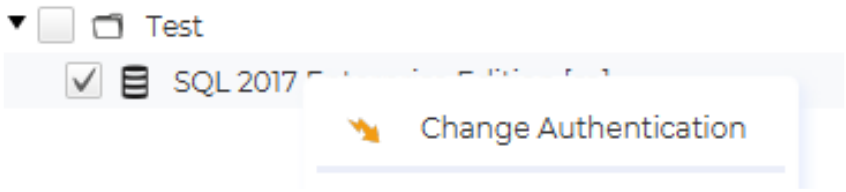
One of the truly unique features of dbWatch is its integrated SQL Worksheet, which, although simple in operation has some sophisticated applications, including cross-database queries.
The SQL Worksheet is not developed from a developer-centric perspective and therefore doesn't include many developer-oriented features such as code optimization or analysis features.

It is meant to give responsible DBA's the ability to locate any problems and correct them, without having to switch between a multitude of tools (same SQL Worksheet for all supported databases).

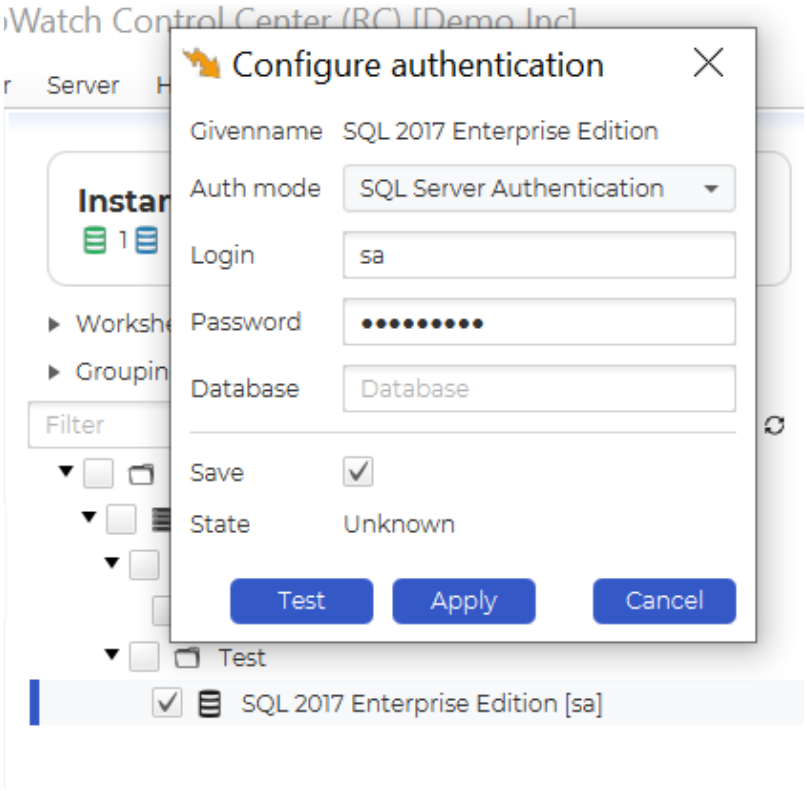
The toolbar along the top of the SQL Worksheet allows you to load and save SQL queries from file and save the output to file.
The tree on the lower left side lets you select among all the instances that are available in Control Center. On the right side of the window shows the query tab for you to write your queries.



First let us select an instance, for example the "SQL 2017 Enterprise Edition" instance. You can access the "Configure Authentication" by right clicking the instance and clicking "change authentication."

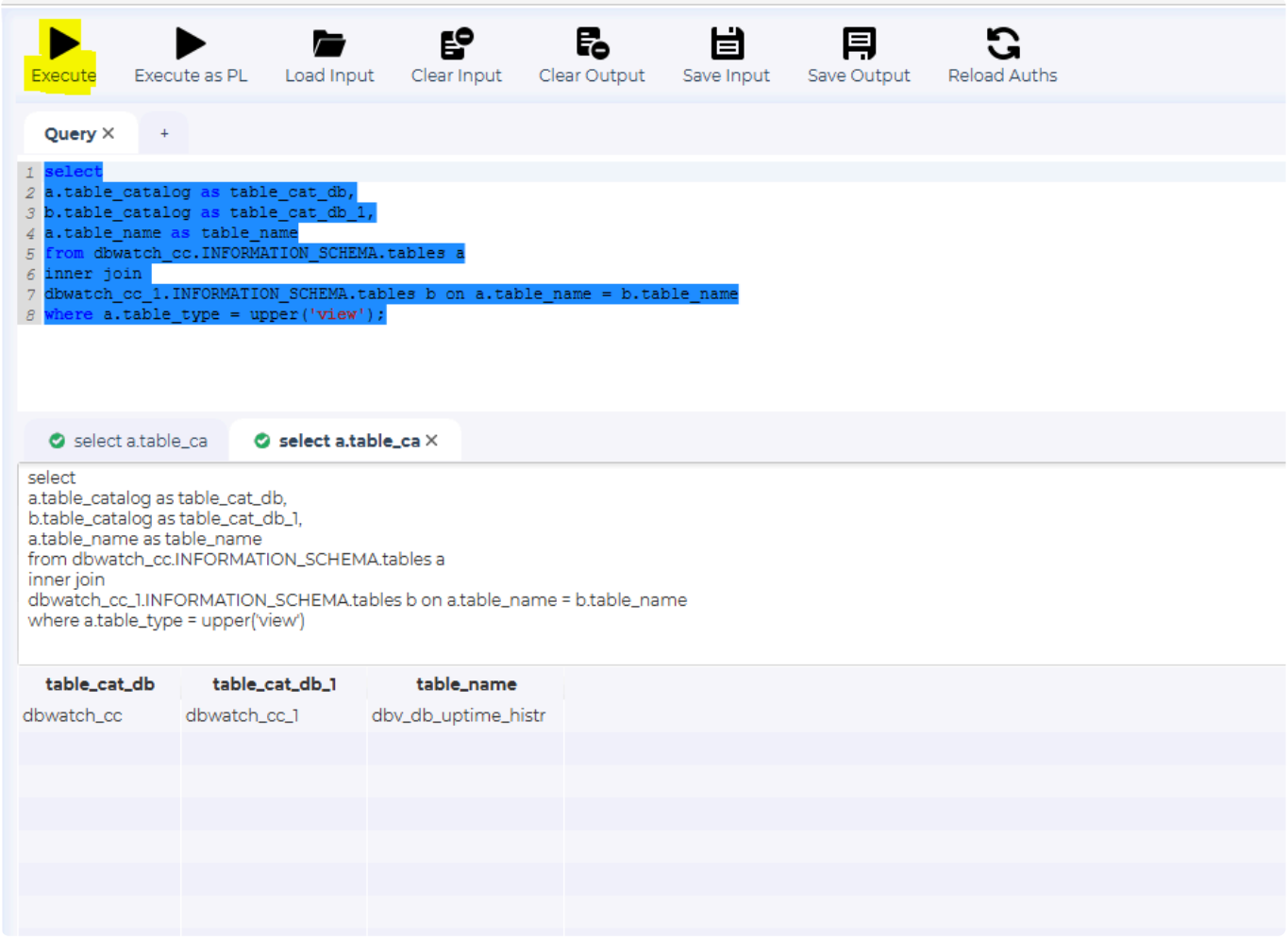


After that, input the login credentials. You can opt to specify the default database or not. For our example, we'll keep it blank and specify it in the query.



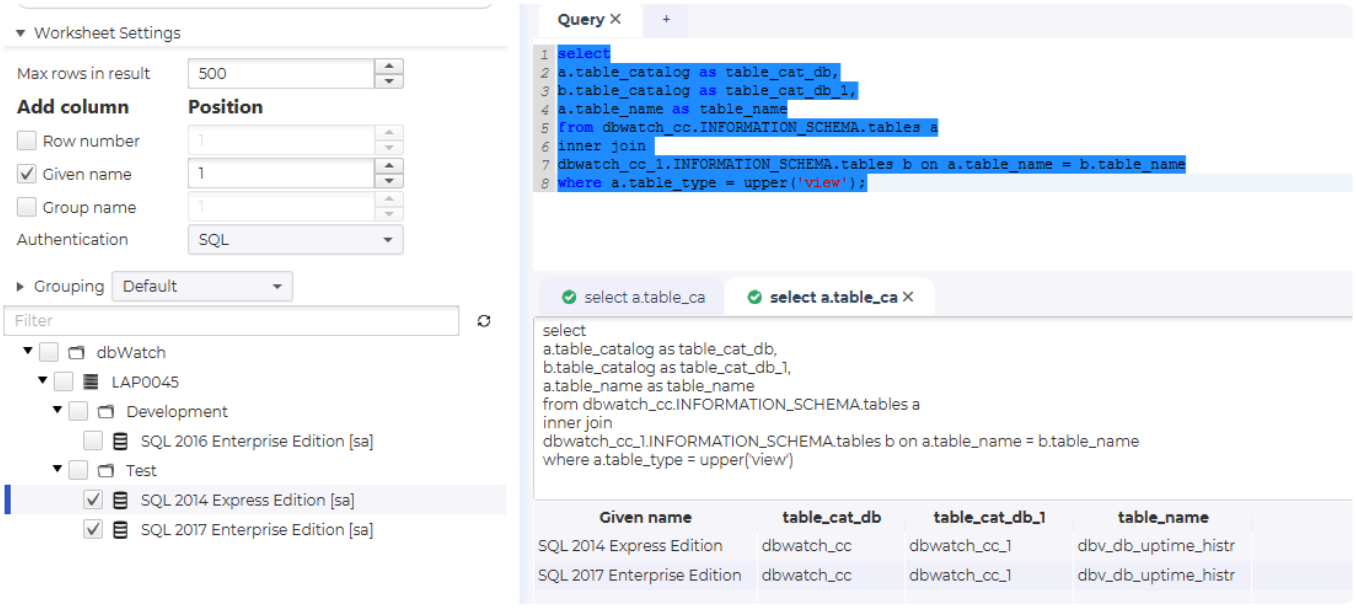
Notice that the current login is shown in the square brackets after the instance name.
Next, lets type in a simple query to execute.

For example, I would like to check what view tables were installed in dbWatch_cc_1 and dbWatch_cc_2. I'll input the query below:



So far everything is as expected. The result appears neatly in the lower window and the summary of the query is found over it.
Now let's try one of the cool features of the SQL Worksheet. When we select an additional instance in the instance tree, and hit Execute.

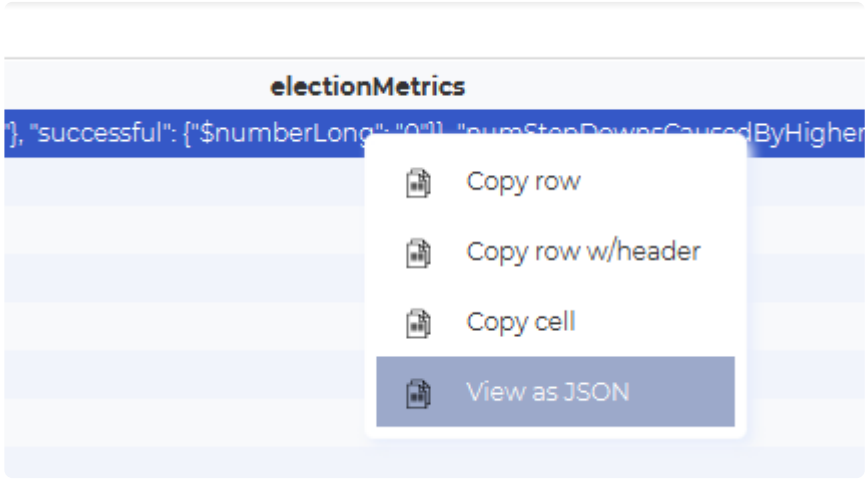
You will now see two rows in the resultset, one from each instance. But we don't know which row is from which instance. This leads us to another feature.
Control Center can add 3 columns to the resultset. A row number, the given name and the group name for the instance the row belongs to. Click on "Given name" in the view above the instance tree, and execute the statement again.



You now see that the given name appears as the first column.
The example only returns one row per instance, but there are no limits on what types of results you can merge in this way. The only requirement is that all instances must return the same number of columns. Go ahead and play around with it yourself to see more of this amazing feature.

Result formating

Some databases return results in JSON format. When this is the case it is possible to right click on a cell and select “View as JSON”.



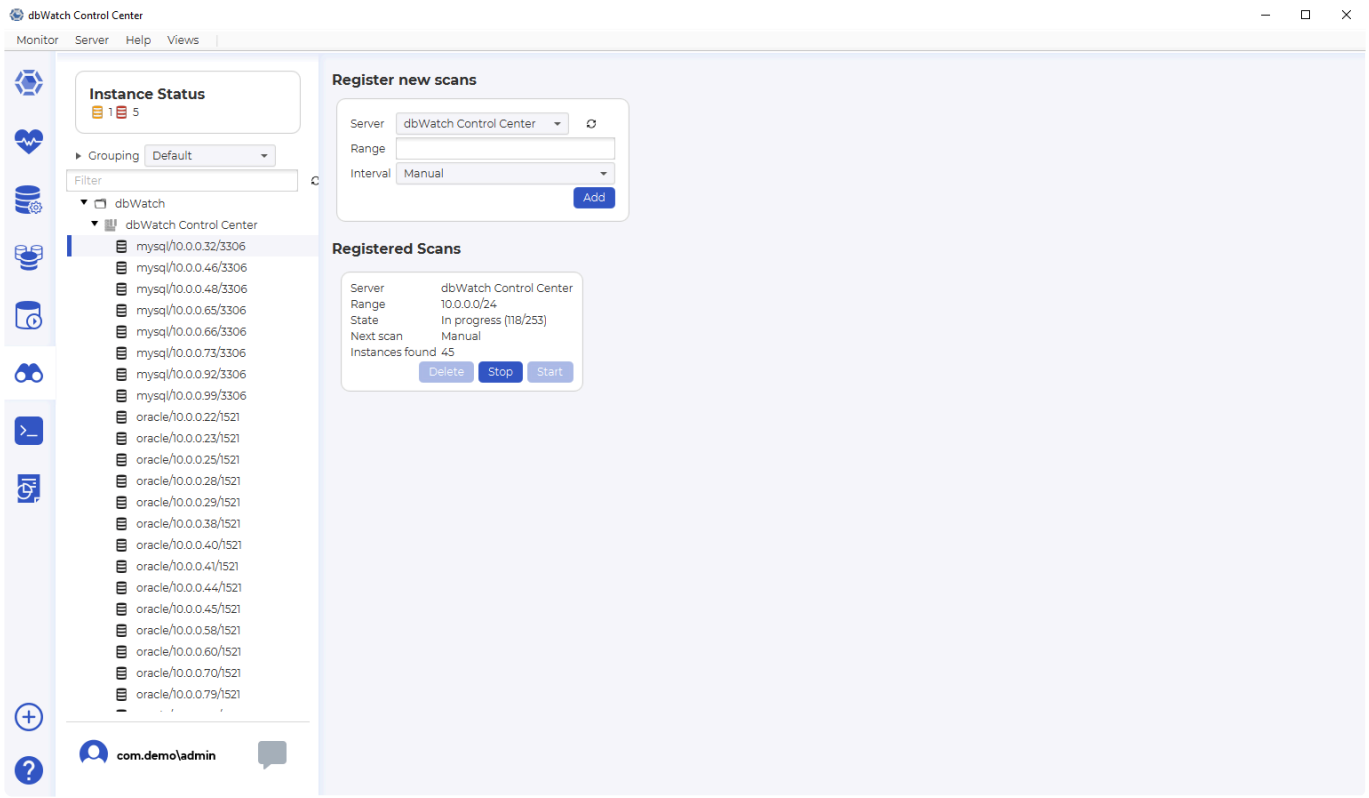
This will open a view with the cell content formatted as JSON.

<< SQL Worksheet / Autodiscover >>

Autodiscover

Discovering Database Instances

The dbWatch Autodiscover feature scans your network and locates database instances that are available. You will find the Autodiscover feature under the “Autodiscover” tab on the left-most side of the dbWatch Monitor.



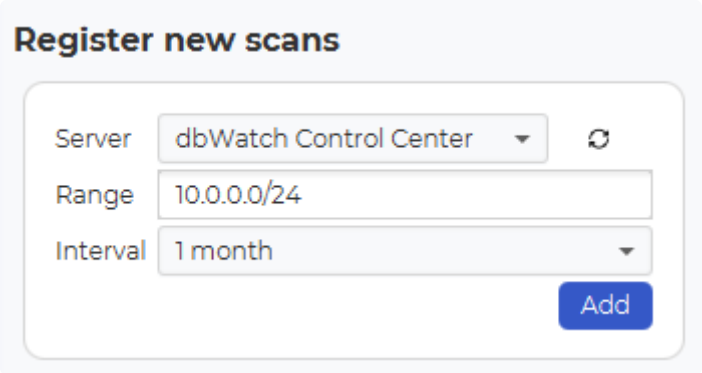
In the Autodiscover view, you have 3 main components.

Instance Tree (LEFT) : you will find all the database instances that have been discovered. You can organize this tree dynamically by manipulating the “Group by” component on the bottom of the tree.

Register new scan box (RIGHT UPPER) : you can select the server, IP range, and frequency of scanning. By clicking *ADD* , you set up the scan with the preference set up.

Registered scan box (RIGHT LOWER) : This shows the current setup and progress of scanning instances. You have the option to *STOP* an ongoing scan, *START* a scan, or *DELETE* the current set up. All boxes under the label are **Registered Scans** are all the setups added.

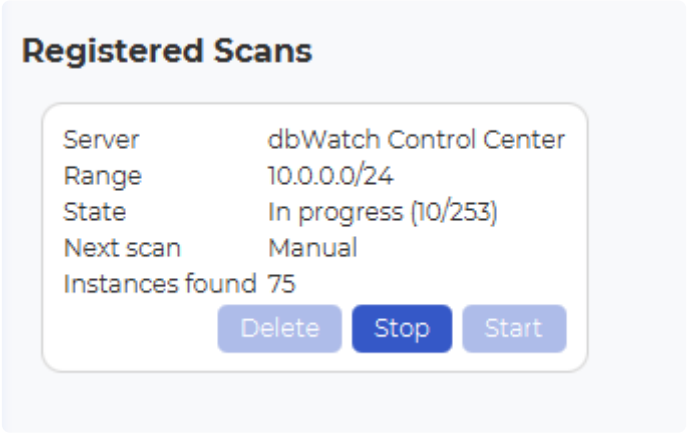
Adding a Scan



In the “Register new scan” box you have three fields.

1. Select the dbWatch Server you want to perform the scan with.
2. Select the network range (using standard CIDR notation) you want to scan. (The refresh button following the Server selection box, will fill the default range for the selected dbWatch Server)
3. Select the interval. Manual means you can start the scan manually whenever you want. You can also specify that the scan should be performed once a day, once a week or once a month.
4. Once you are satisfied, click “Add”.

After a couple of seconds, a new box representing this scan should appear.

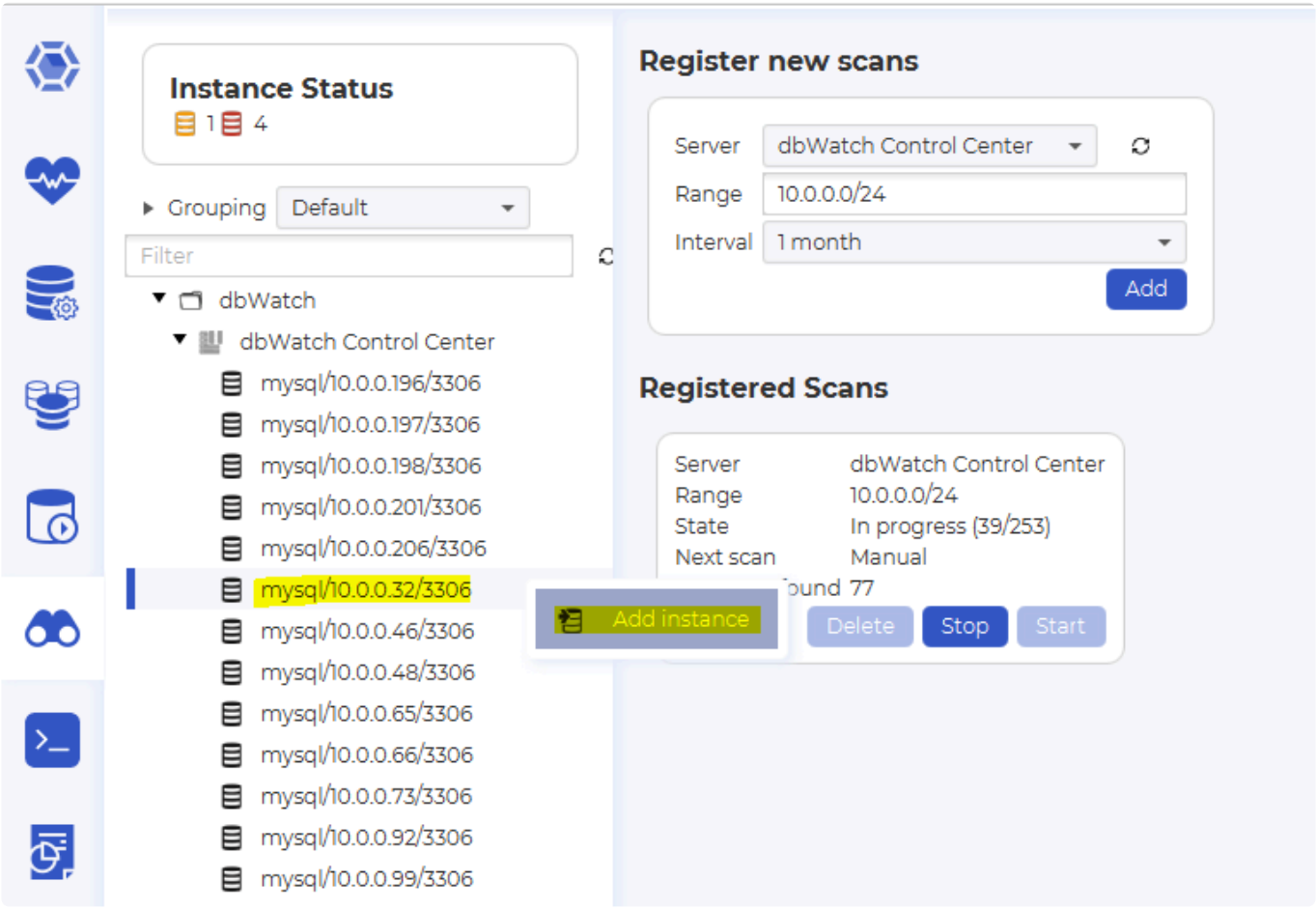


Here you see the scan progress, and you can start/stop (or delete) the scan.

Adding discovered instances

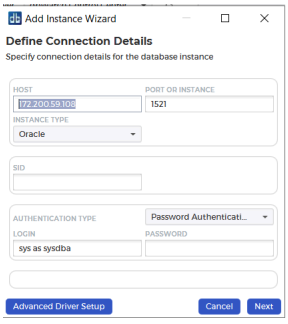
When instances are discovered, they will automatically appear in the tree.

To add an instance, simply right-click it and select “Add instance”. Adding an instance is similar to adding them manually.

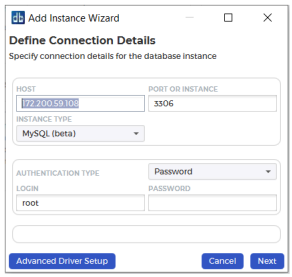


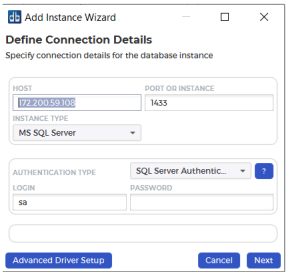
An “Add Instance wizard” will appear where their Host IP, Ports, Username, and instance type filled with. Base on what instance type you are adding, you may need to supply additional information aside from a password. For oracle instances, you need to add the correct *SID*. Otherwise, just type in the admin password.

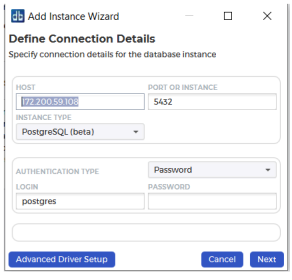
ORACLE



MYSQL / MS SQL / Postgres







Take note we are using the default admin credentials defined by Control Center. We are not using the credentials that your organization is using. In installing your instance manually, you can check the Additional Guide for Instance Installation to know more on

how you can change the setup when adding those instances.

The scan is not a full network scan, but a scan of the ports defined in the file “autodiscover_ports.txt” located in the “[dbWatch Server]/server/resources/” catalogue. For details on the file specification, see Defining Autodiscover Ports.

<< Autodiscover / Defining Autodiscover Ports >>

Defining Autodiscover Ports

By default autodiscover scans the following ports:

MS SQL Server: 1433
Oracle: 1520, 1521, 1522, 1523, 1524, 1526, 1527
Sybase: 4998, 4999, 5000, 5001, 5002
MySQL: 3304, 3305, 3306, 3307, 3308
PostgreSQL: 5429, 5430, 5431, 5432, 5433, 5434

You can override the defaults by creating a file called “autodiscover_ports.txt” and place it in the dbWatch Server config catalog (“C:/ProgramData/dbWatch/[version]/server/config” on a standard windows installation)

The file must contain an xml like syntax that defines a set of ports/port ranges for each supported DBMS type in the following way:

```
<dbms>
  <name>oracle</name>
    <ports>
      <port>1520-1527</port>
      <port>5633</port>
      <port>5635</port>
    </ports>
 </dbms>
```

As you see, a dbms tag contains:

- A name defining the dbms type (legal values: “oracle”, “sqlserver”, “sybase”, “mysql”, “postgres”, “mysql”).
- A ports tag with a set of port tags. Each port tag can define a port range (f.ex 1520-1527), or a single port (f.ex 1521). Maximum allowed value for port is 65535.

The dbms tags are contained in an enclosing port-def tag, so the file will look like this:

```
<port-def>
      <dbms>
           <name>oracle</name>
           <ports>
               ...
           </ports>
      </dbms>
      <dbms>
           <name>sqlserver</name>
           <ports>
               ...
```

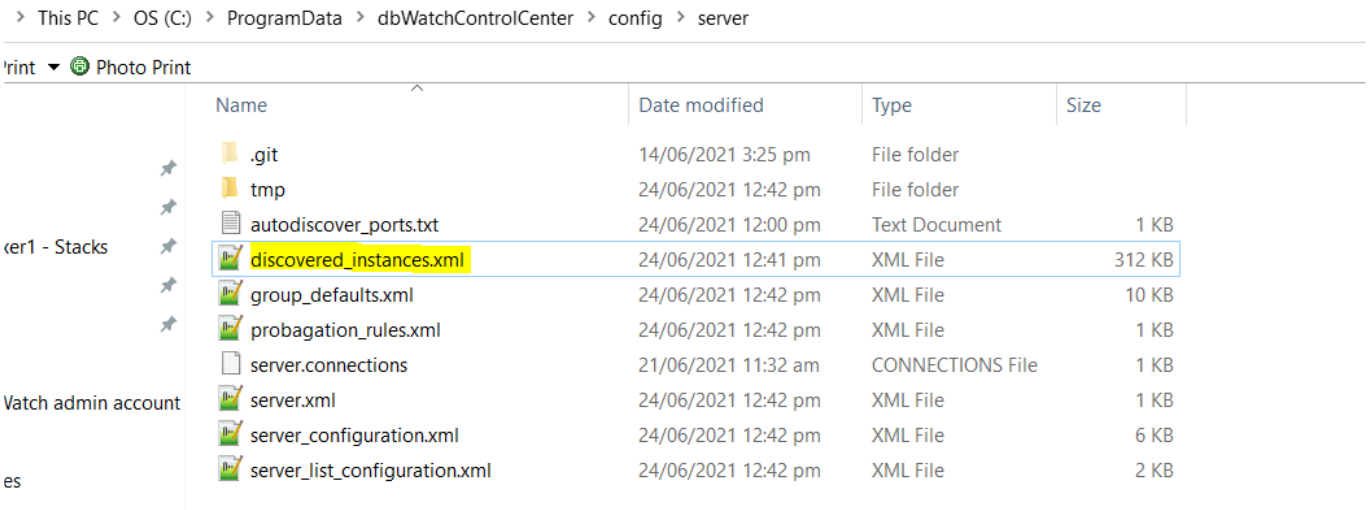
```
            </ports>
        </dbms>
        ...
</port-def>
```

Please note that the scanning function in autodiscover is designed to not be too aggressive, because that could trigger firewalls and IDS system within the company. The scanning function will scan around 10-15 ports per second spread across hosts, so a large port range will make the scanning time consuming.

[<< Discovering Database Instances] / [Auto-discover with different default port >>]

Auto-discover with different default port

We'll look deeper into how auto-discover works. First of all, you need to be familiar with what file the data is being stored in. The default directory can be found in "C:\ProgramData\dbWatchControlCenter\config\server" and it's in file "discovered_instances.xml".



> Note this can change depending on where you put your working directory. You must know where it's stored. You can check "About dbWatch" or click the dbWatch Icon to know where your working directory is.

When you open the file, you may something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<autodiscover>
        <instance>
                <dbms-type>postgres</dbms-type>
                <hostname>Server</hostname>
                <hostip>172.200.59.108</hostip>
                <port>5432</port>
                <id><![CDATA[discoveredinstance:postgres/172.200.59.108/5432@s
erver:5713907582237493967/-78510371179937116O2]]></id>
        </instance>
        ...
        <scanned>
                <ip>172.200.48.1</ip>
                <time>1624507255855</time>
        </scanned>
        ...
        <scan>
                <range><![CDATA[172.200.48.0/20]]></range>
                <interval>-1</interval>
```

```
                <last-started>1624507255481</last-started>
                <last-completed>1624509560052</last-completed>
        </scan>
</autodiscover>
```

What you can observe are those within the < instance > tag are the discovered instances while those in the < scanned > tags are the IP addresses scanned without an instance. Additionally, < scan > shows you the details of the scan. I won't go over the details but you can read on Discovering Database Instance.

Port is different from the known default port

Let's say you are using a different listening port to that of the default port which is 5432. As we can see the new listening port below, it's change to 5455:

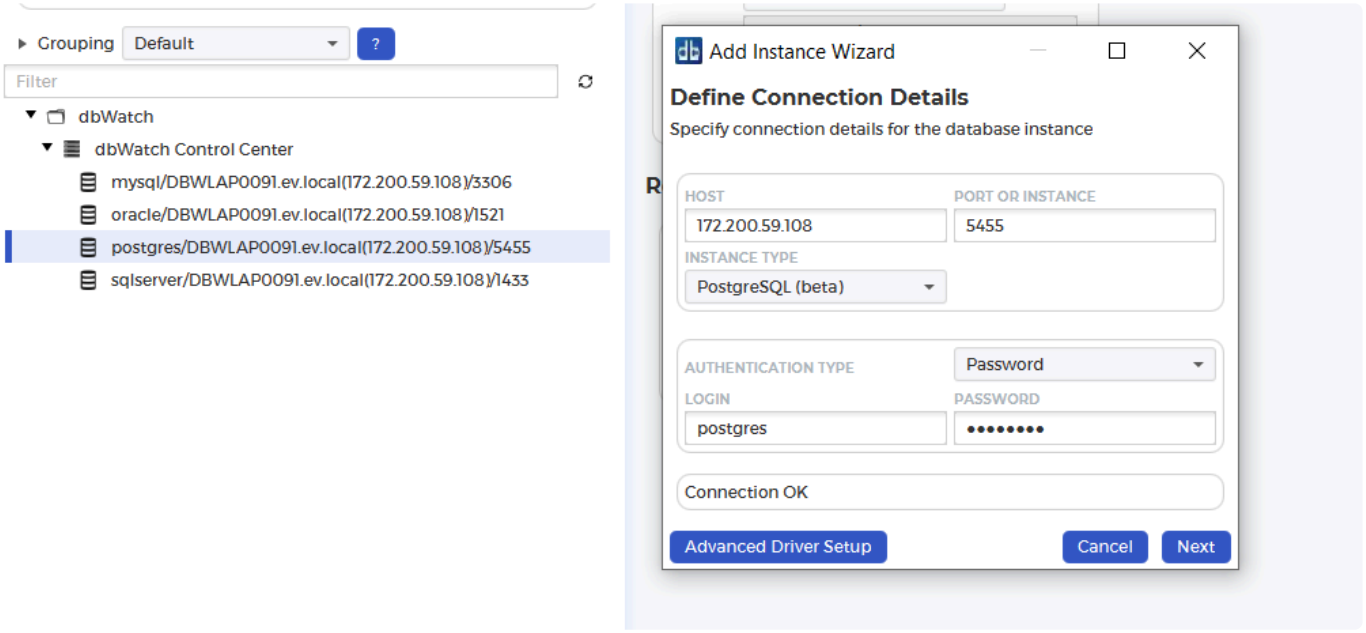
```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5455                             # (change requires restart)
max_connections = 100                   # (change requires restart)
```

How can you change the listening port to reflect those changes?

1. One thing to do is to check the "discovered_instances.xml" and alter the port value from 5432 to 5455.

```
        <instance>
                <dbms-type>postgres</dbms-type>
                <hostname>Server</hostname>
                <hostip>172.200.59.108</hostip>
                <port>5455</por
t>
                <id><![CDATA[discoveredinstance:postgres/172.200.59.108/5455@s
erver:5713907582237493967/-78510371179937116021]></id>
        </instance>
```

2. Close the monitor and restart the dbWatch server.
3. Open the dbWatch monitor and you should be able to see the changes:



<< Defining Autodiscover Ports / FDL console >>

FDL console

See FDL – Farm Data Language

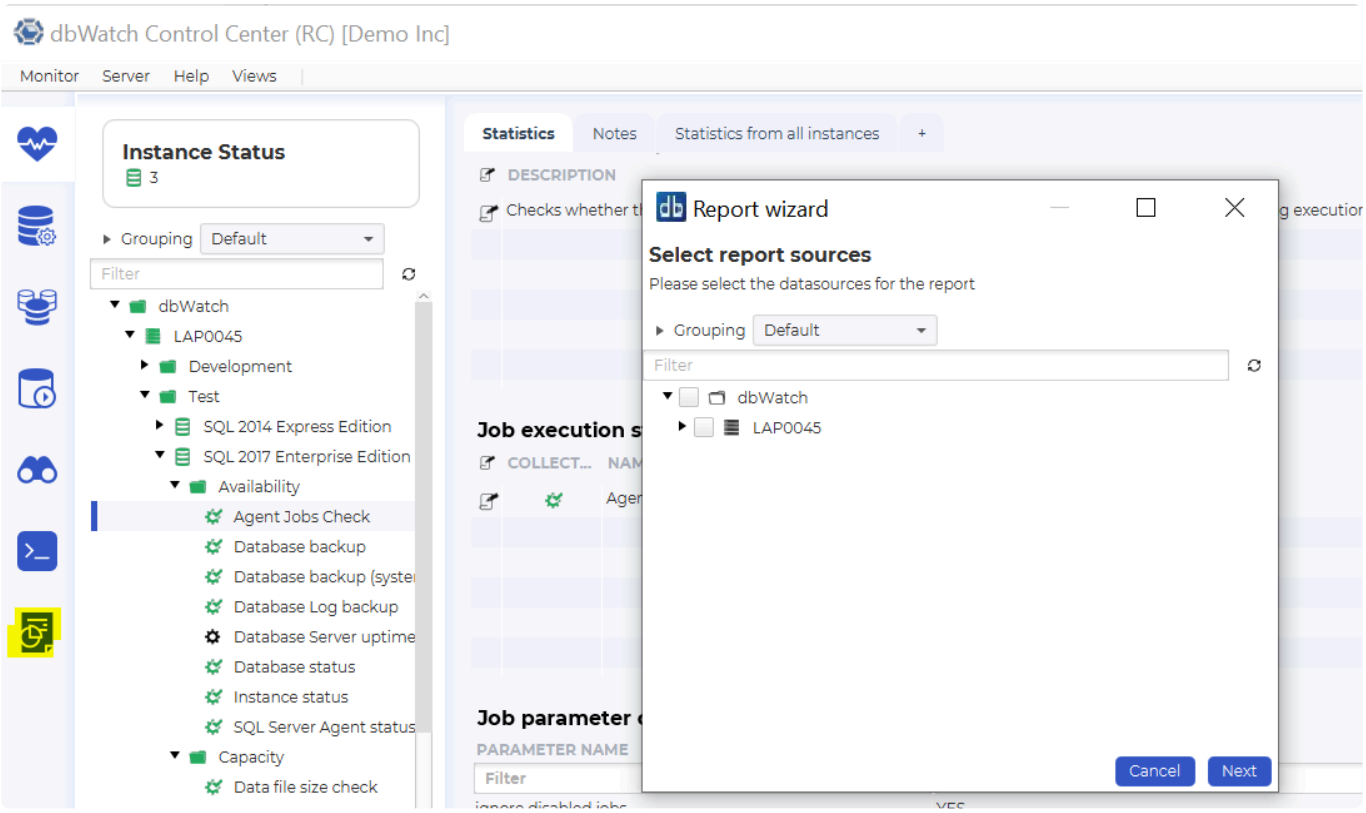
<< Auto-discover with different default port / Reporting >>

Reporting

Control Center comes bundled with a series of report templates that you can use to generate standard reports on various aspects of your database environment.

It also allows you to edit the templates or create your own for fully customizable reporting which will be included in future developments. However, you can generate reports on the fly.

You can access the Report Module by clicking the bottommost of the icon.



From here you can choose to

- Generate Report

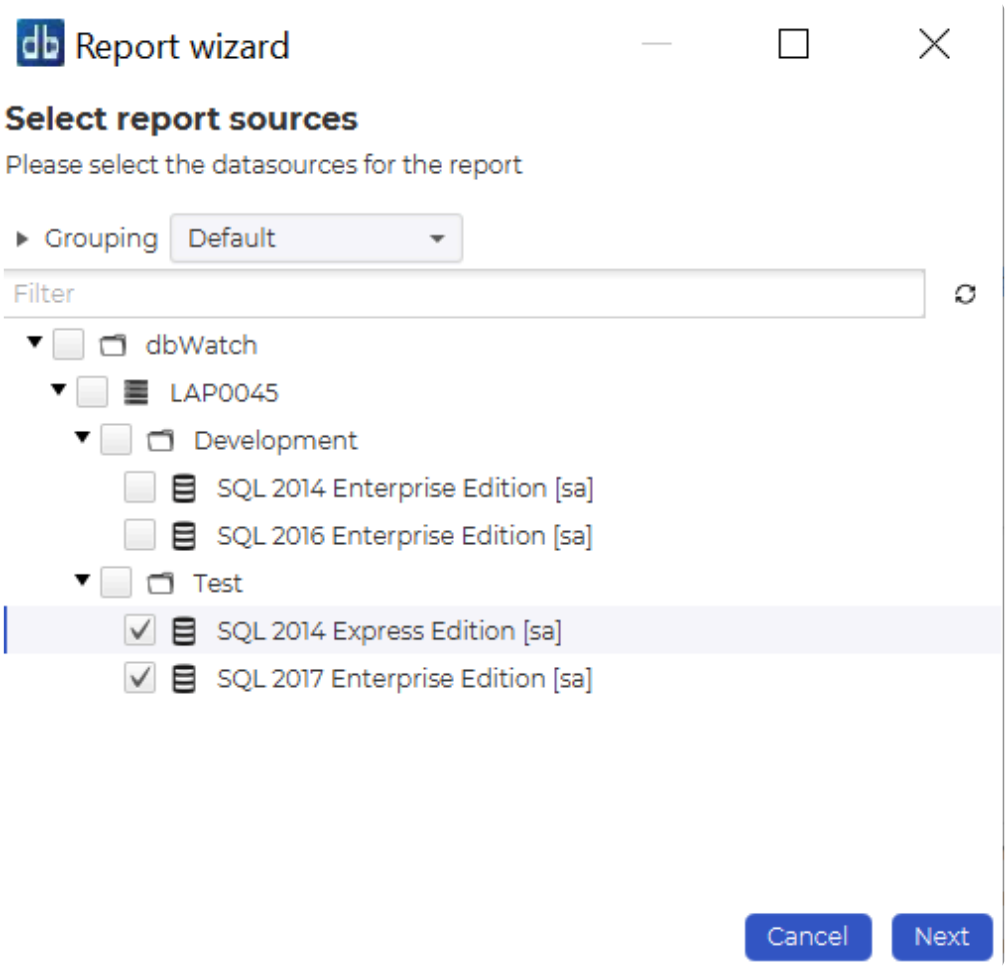
It is also possible to edit reports and create new custom reports. Information about that is available in the customization section.

<< FDL console / Generate Reports >>

Generate Reports

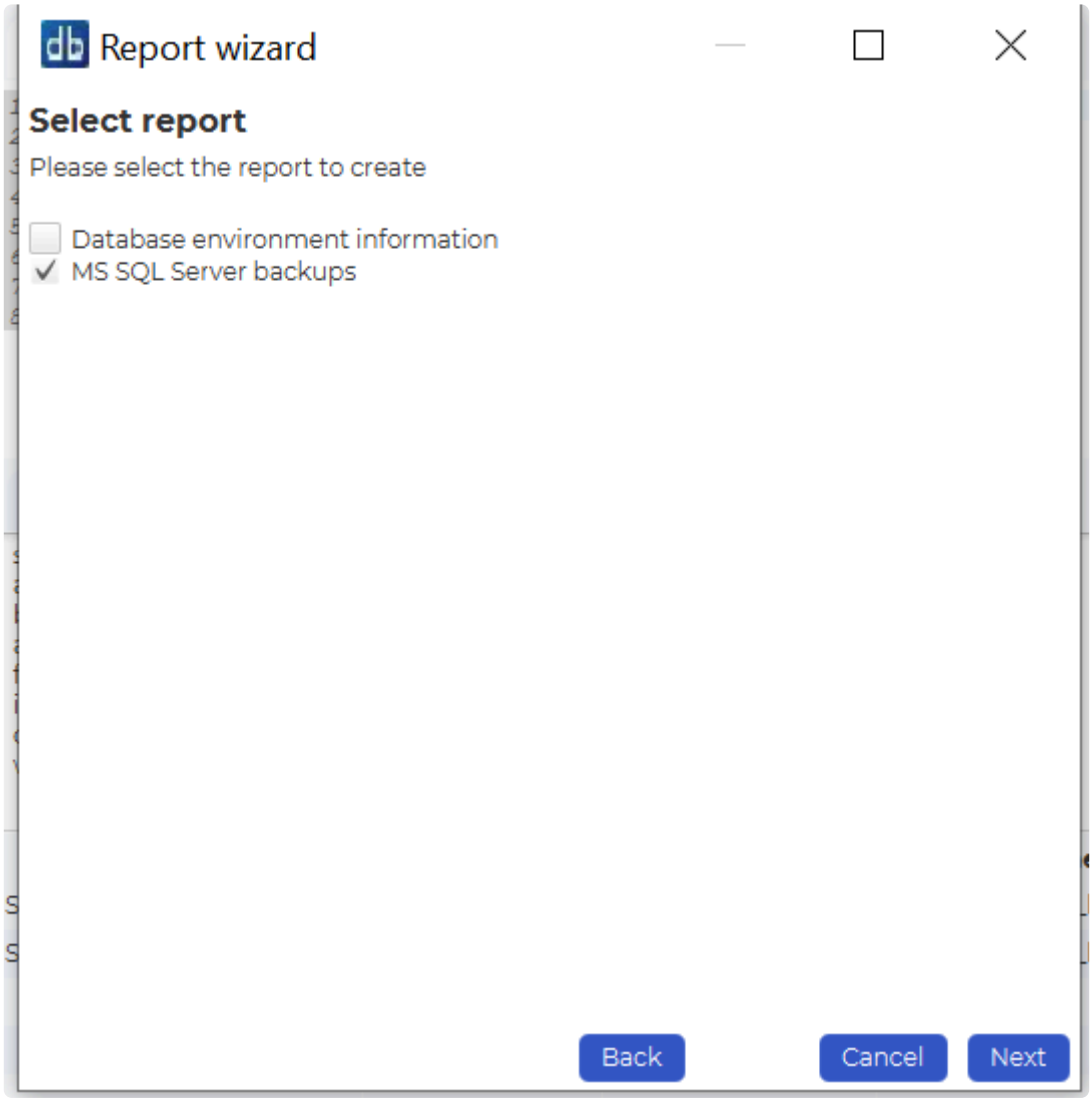
When you click on Report Module, the report wizard will be started.

In the first step you will be asked to select the database instance you wish to generate a report for. The instance that does not have a user/login set for generating the report content, will be shown greyed out with a “No authentication” tag. To set a user/login, simply click on the instance and a dialog will appear.

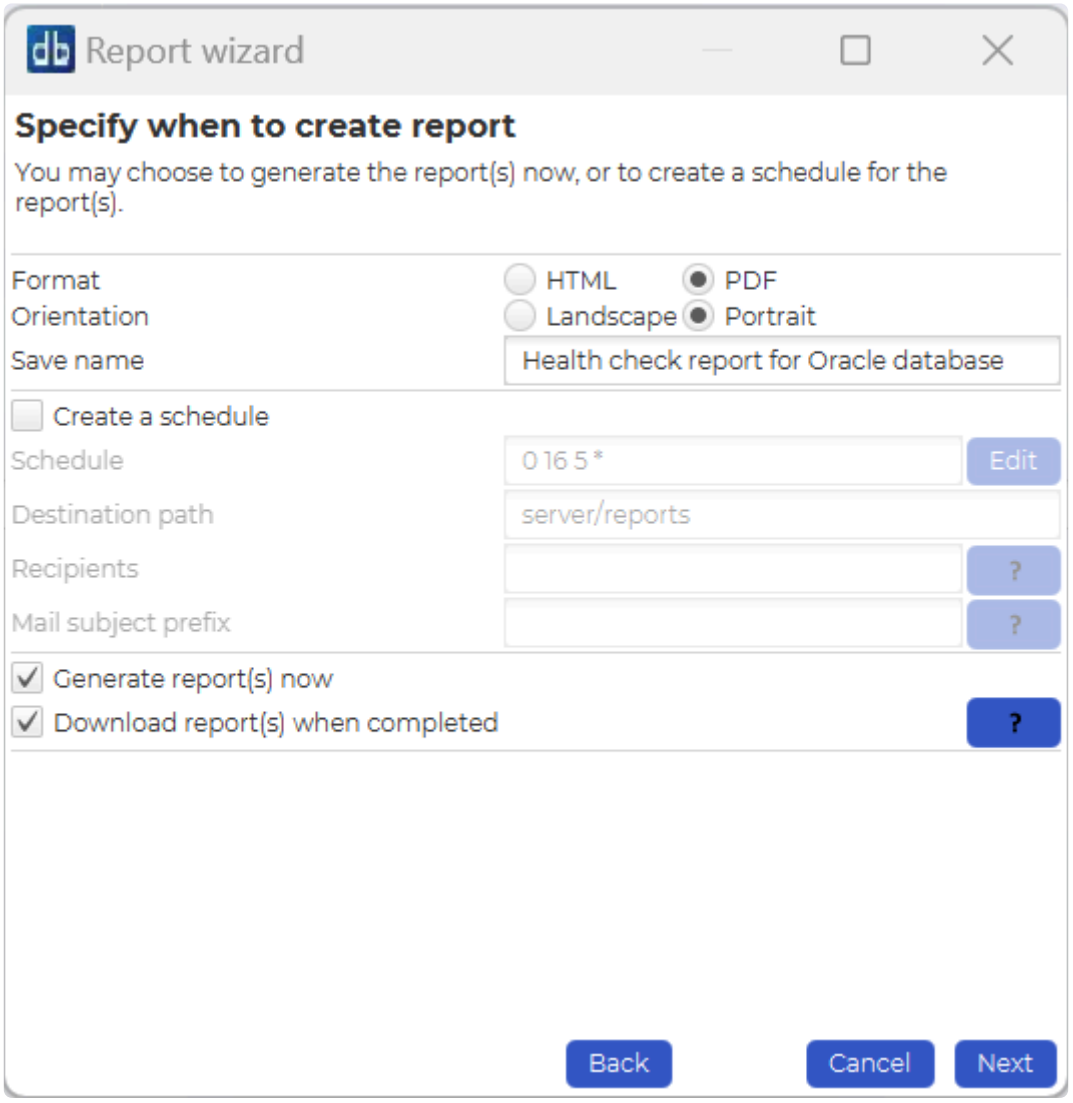


Authentication is set during the installation process. You can edit your authentication by right clicking on the instance name and clicking “Change Authentication”. Remember to hit “Apply” to save the changes.

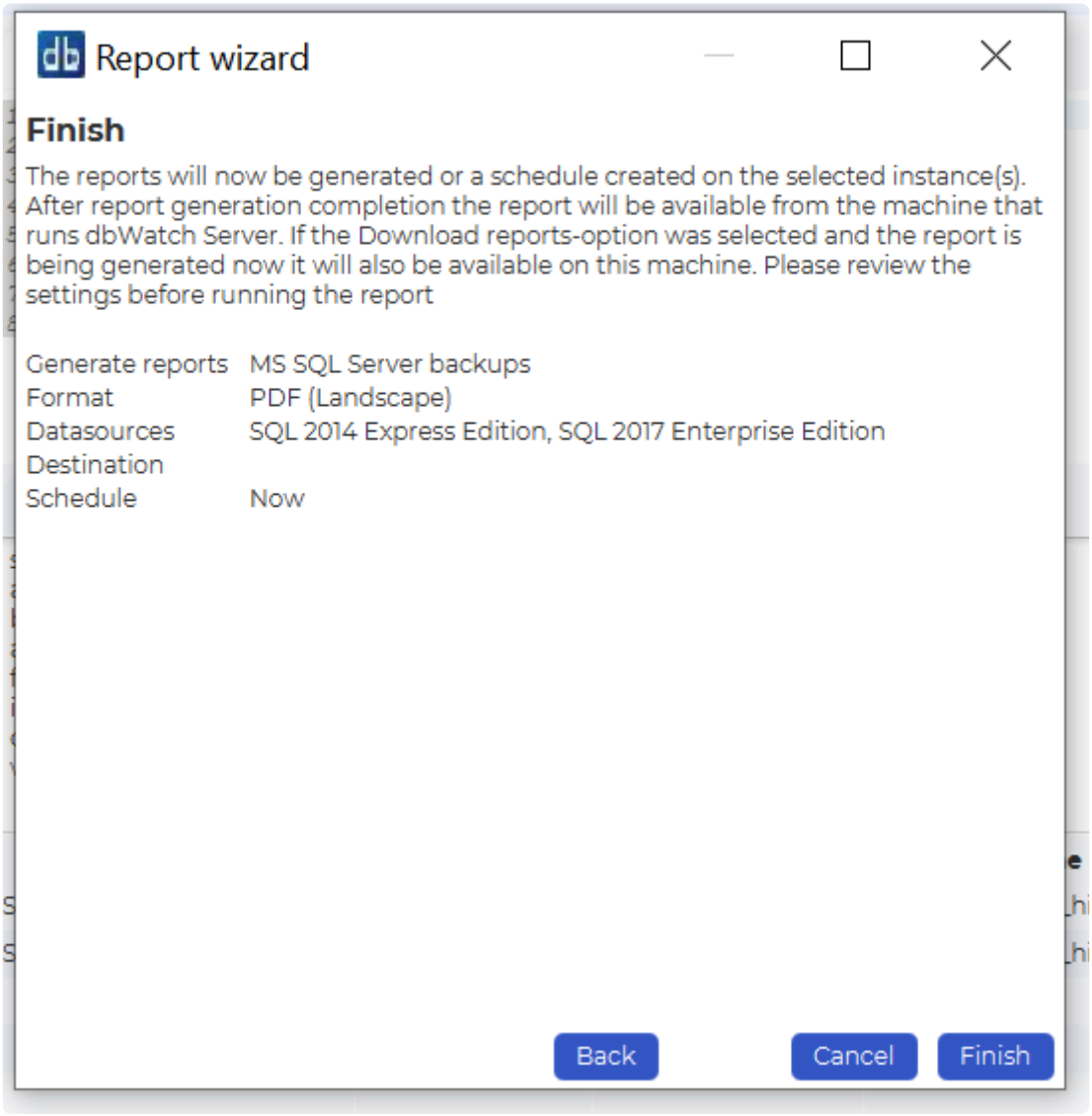
The next step will show you a list of all available report templates that are compatible with the instance you selected. Pick the report you want to generate. For this example, let’s say I want to generate a report around my MS SQL backups.



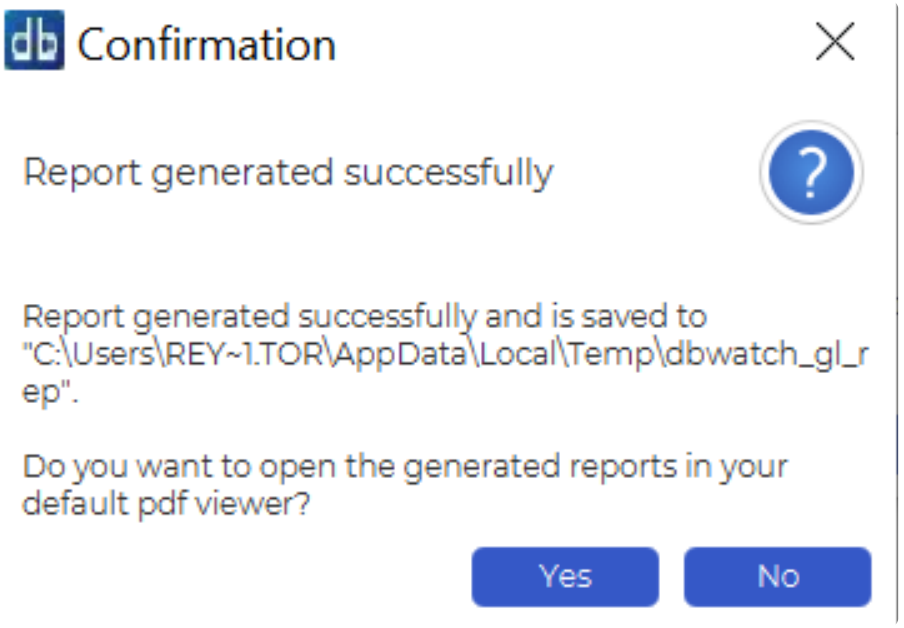
Next you can specify the format the report should be generated in (PDF or HTML), for PDF you can also specify Landscape or Portrait mode.
Then you have the choice between generating the report now, or creating a schedule that will generate the report at specific times. If you choose to generate the report now, and click the “Download reports when completed” checkbox, the report will be opened in your default viewer when it is completed.



The last step of the wizard is a summary where you can verify that you are satisfied with your choices. Click “Finish” to apply.



After generating the report, a screen will prompt notifying you to open the report.



You can click “Yes” to open the generated report.

dbWatch AS

www.dbwatch.com

September 28, 2020 4:41:33 PM

MS SQL Server backups

[<< Reporting] / [Scheduled reports >>]

Scheduled reports

Scheduled reports are defined to be generated at specified intervals, and can be automatically emailed to you when they are completed.

Create a scheduled report

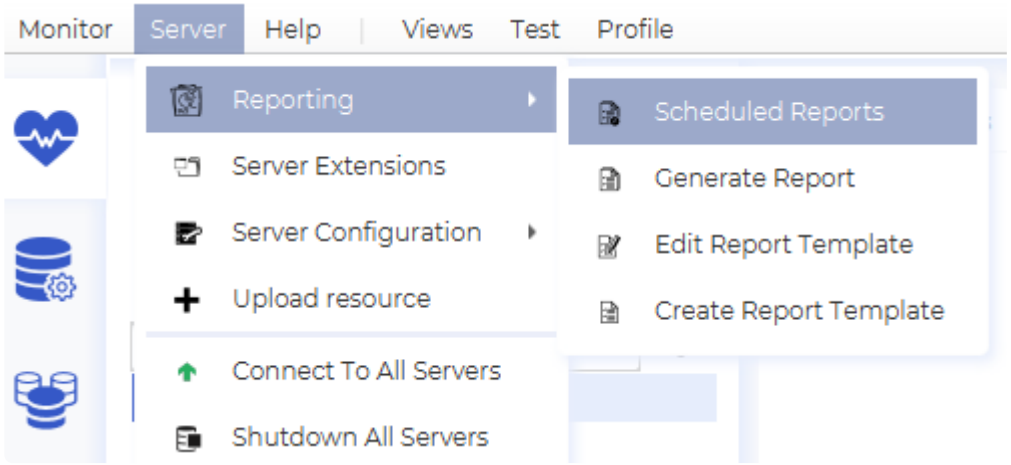
To create a scheduled report see Generate Report.

Configure a scheduled report

Once a report has been scheduled you can edit the definitions, in the **Scheduled Reports** view.

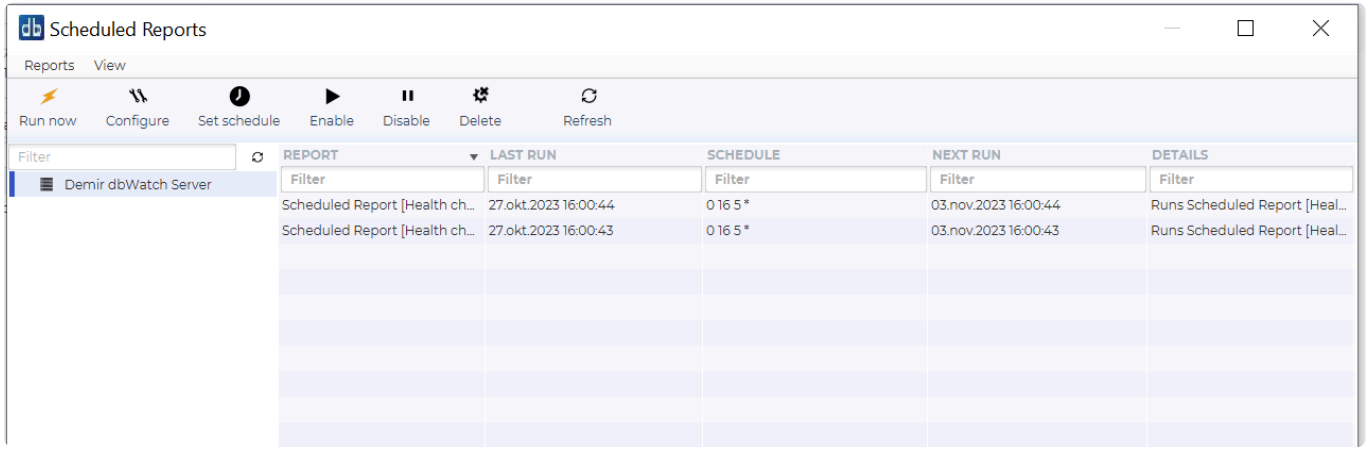
How to get here

The Scheduled Report view can be accessed by selecting **Reporting** -> **Scheduled Reports** on the **Server** menu.



Overview

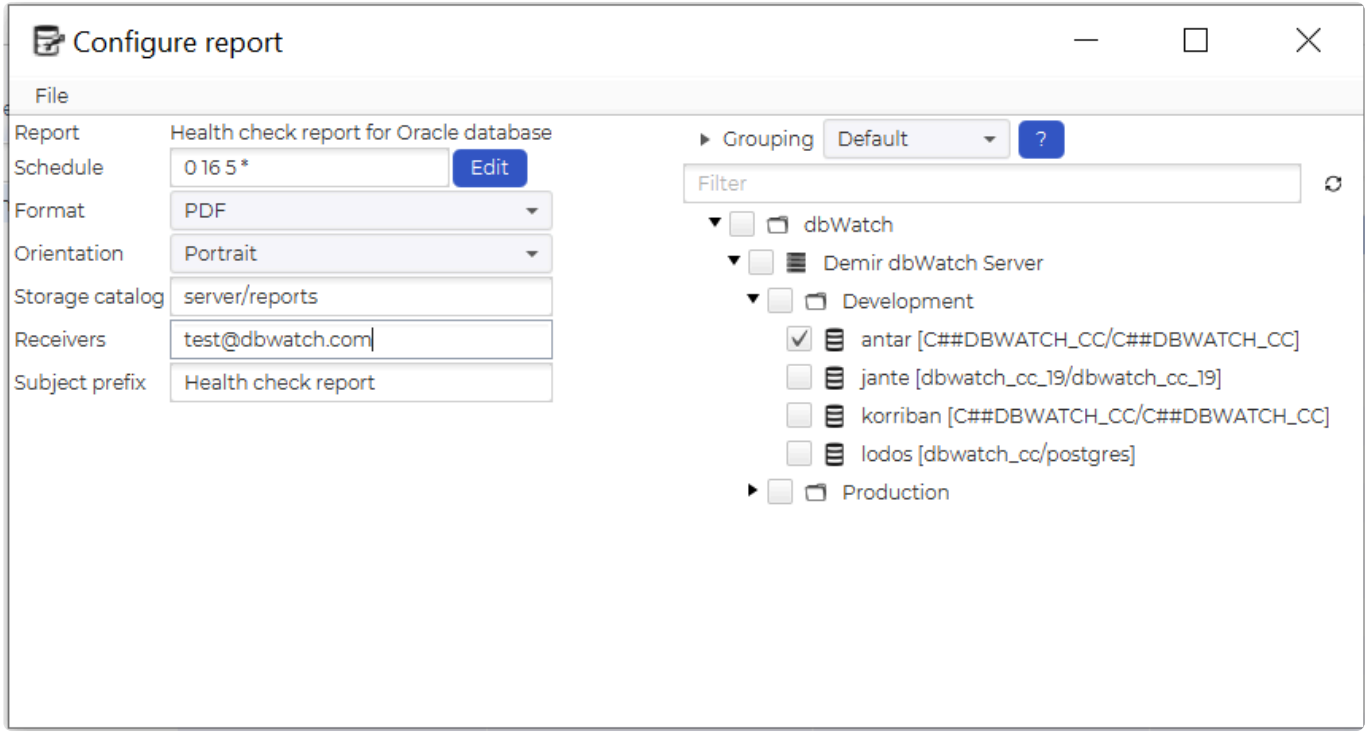
In the scheduled reports view you can see all the reports that are scheduled to be generated.



There are a set of commands that you can perform on the reports. **Run** triggers the report generation now if you don't want to wait for the next scheduled generation. **Configure** opens the configuration view (more on this in the next section). **Schedule** lets you change the intervals for report generation. **Enable/ Disable** lets you enable and disable the report generation, and **Delete** removes the report.

Configure

When you click on **Configure** you will see the configuration view below.



Here you can edit the schedule, the format (html/pdf) and the orientation (landscape/portrait) for the generated report.

The **Report folder** is the catalog where the finished report will be placed. This is a path on the dbWatch Server, it can be absolute or relative to the dbWatch installation folder.

The **Receivers** field is where you can specify the email addresses (if any) you want the report sent to when it is completed. This is a comma seperated list.

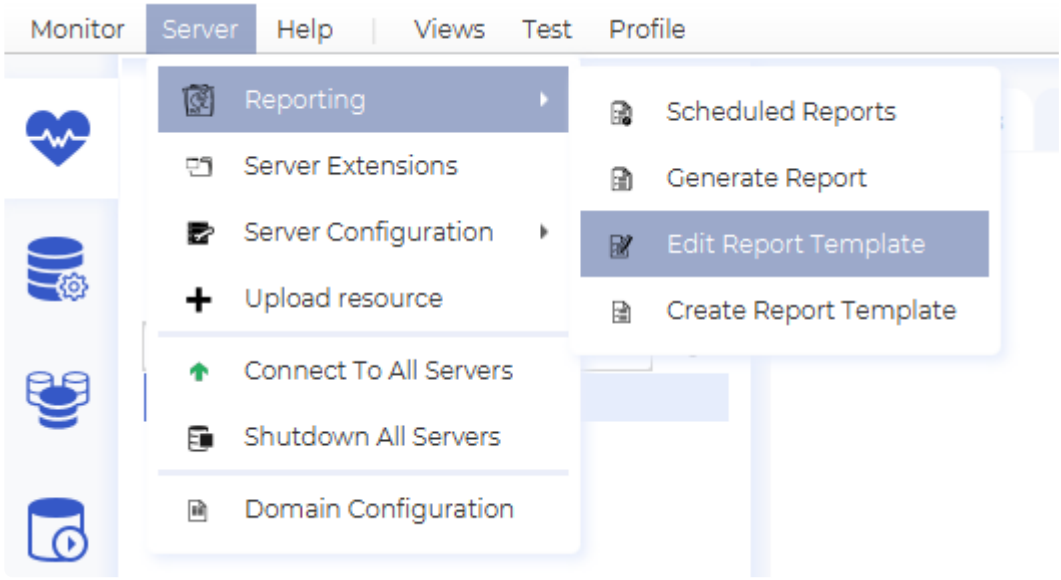
The lower part of the view is where you can specify the database instances this report should be generated for. In the square brackets behind each instance you see the username/login that will be used and the database the connection will be made to (if relavant for the specified dbms type). Right clicking on the instance lets you change this authentication information.

[<< Generate Reports] / [Edit Report Templates >>]

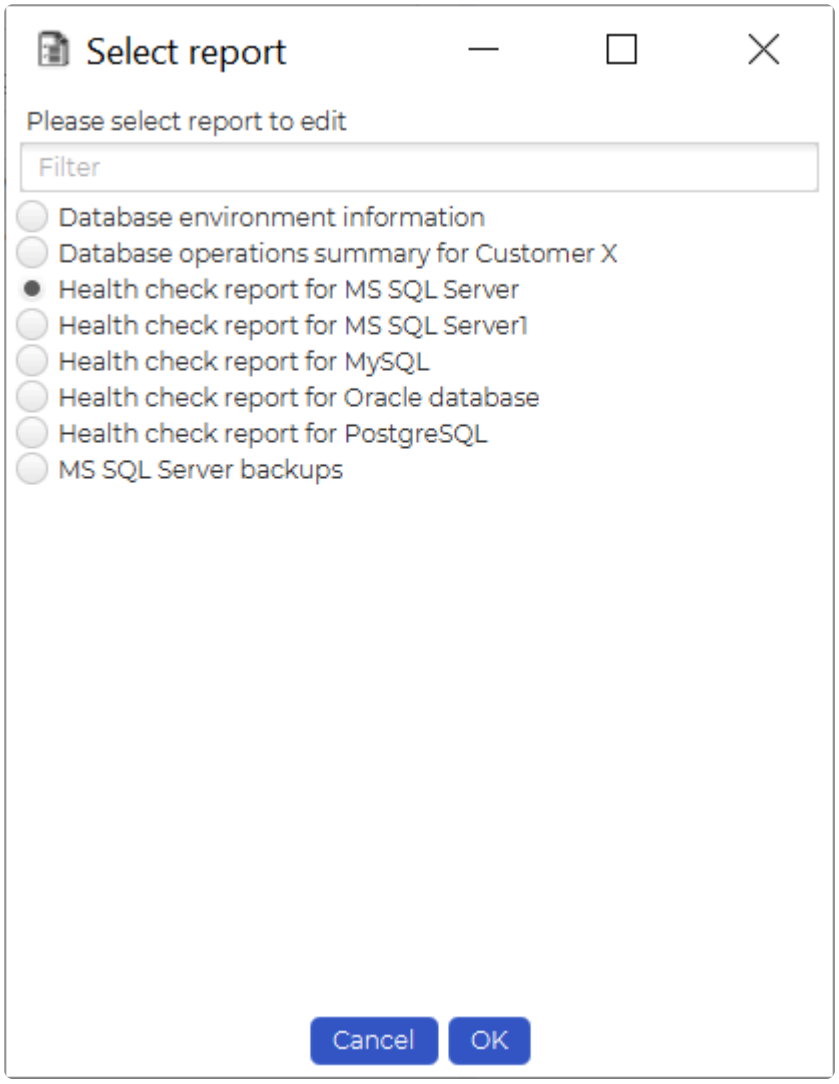
Edit Report Templates

Report Selection

When you click on **Server** -> “Reporting” on the toolbar and then select “Edit Report Template”, you will see a list of all available report templates.

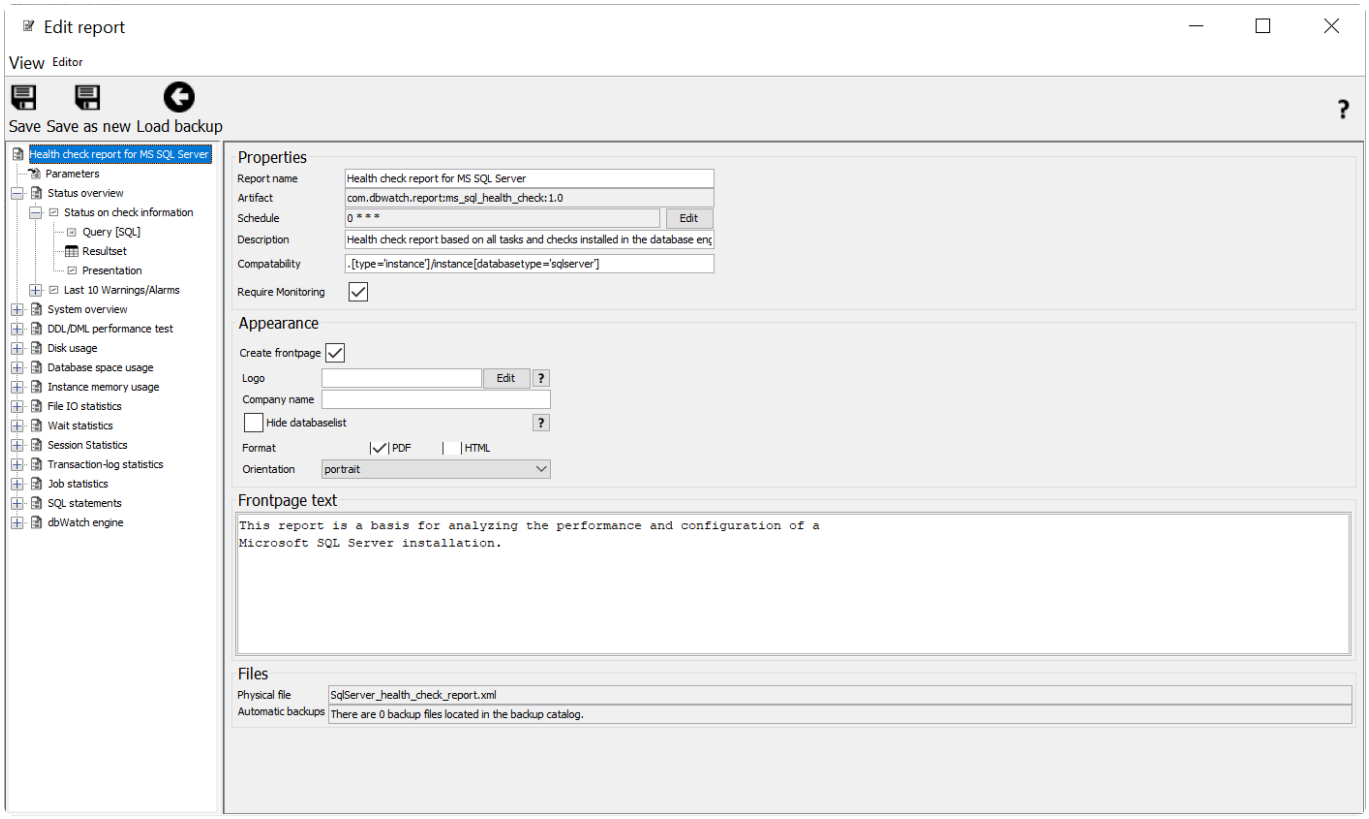


Select the report template you wish to edit and click “OK”.



Report properties

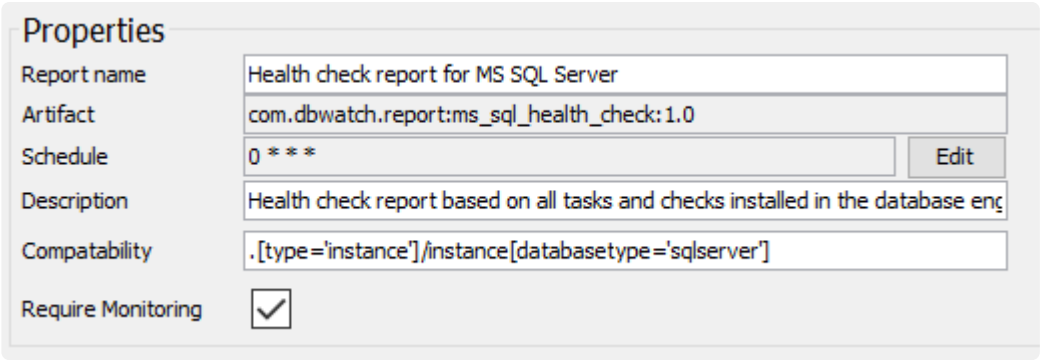
You will now see the main view of Report Editor. On the left side you see a tree structure giving you access to the metadata/properties and the various chapters and sections in the report.



When the top node in the tree is selected you see 3 views on the left side, **Properties**, **Appearance** and **Frontpage text** as well as, the physical location of the report template file and where the automatic backups are placed.

Report Properties

In the Properties section, you can specify the report name, the default schedule, a short description, and the compatability for the report.



Report Appearance

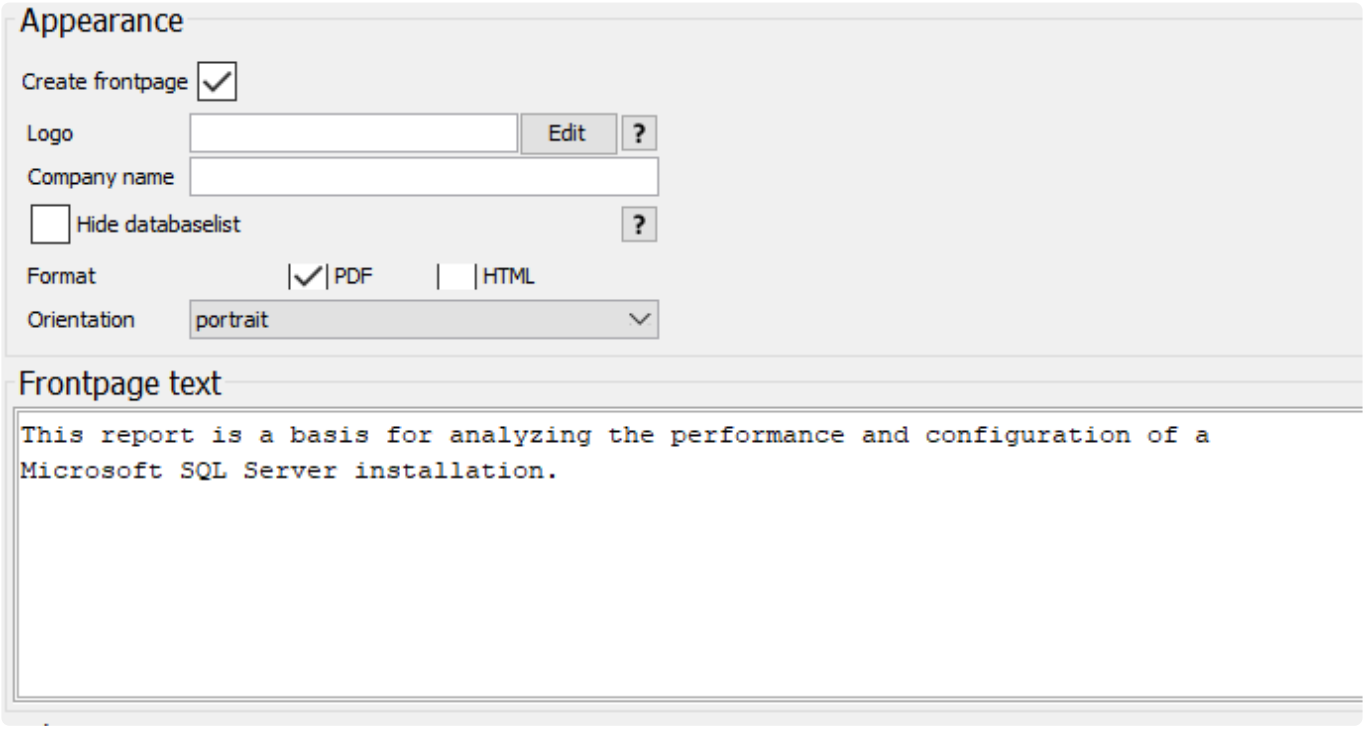
In the Appearance section you can specify a custom logo and company name for the report (which will appear on the frontpage of the report).

The “Hide databaselist” checkbox allows you to specify that the list of which database instances this report is generated for should not be shown. (This is normally page two of the report).

Format lets you choose the between PDF and HTML as the default format for this report.

And finally “Orientation” lets you specify if the report is in portrait or landscape mode by default.

Frontpage text is the text that will appear on the frontpage of the report.

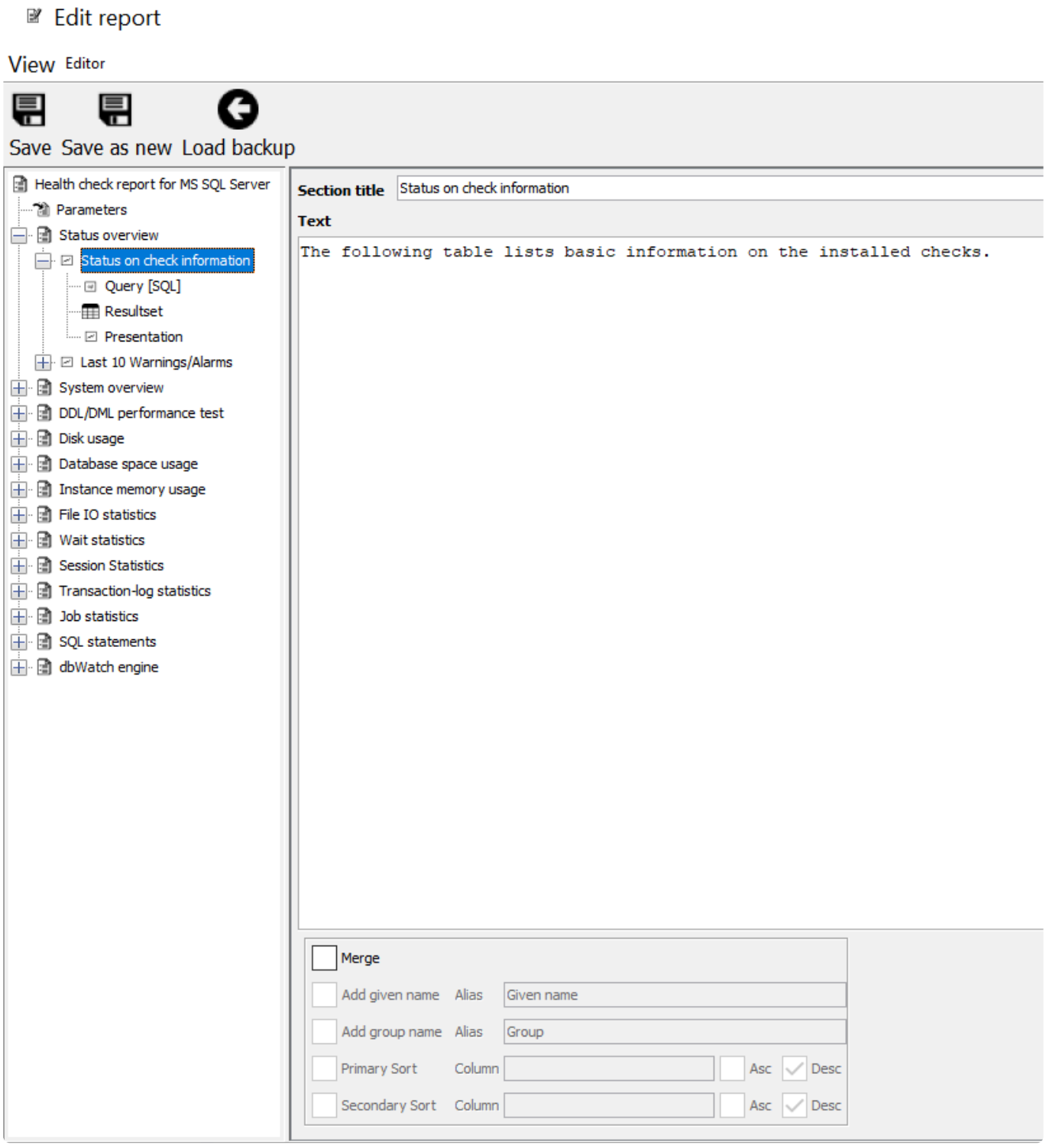


Chapter

A report can be divided into several chapters, each containing one or more sections.

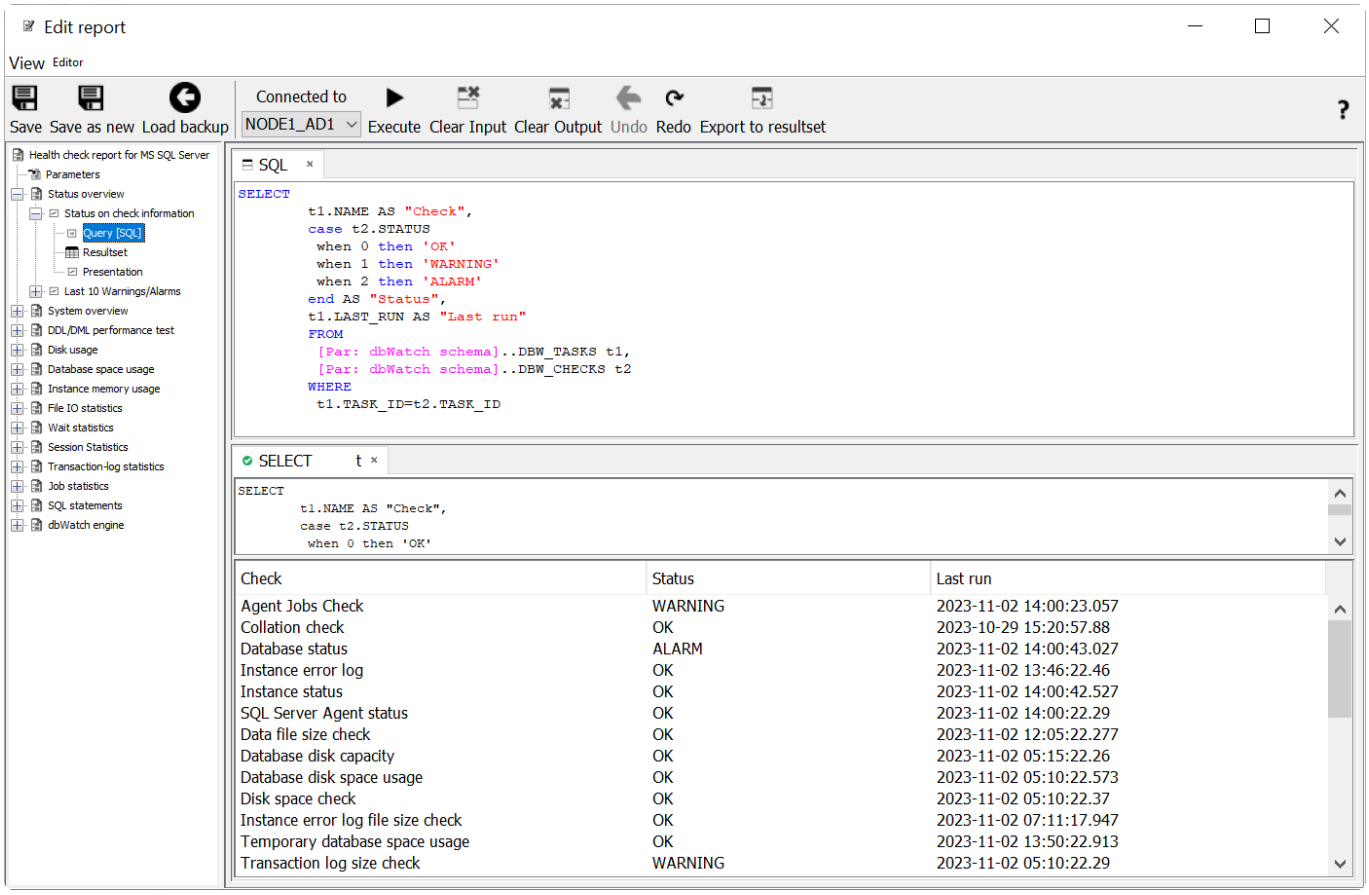
Section

When you click on a section, you can edit basic information like the title and a text that will appear at the start of the section. There is also a panel that lets you specify that this section should be a “Merged” section. (More on this later).

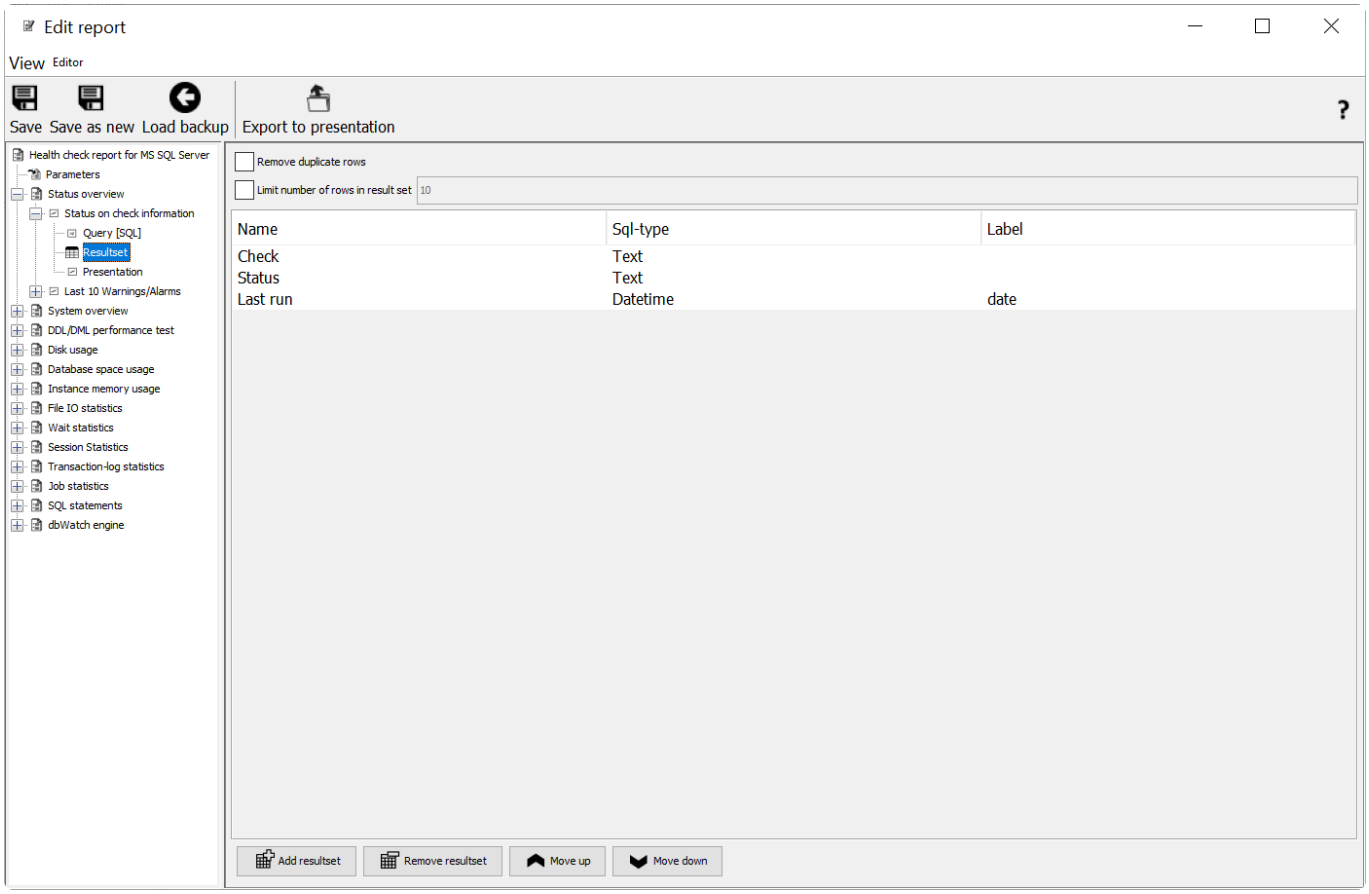


Each section defines an sql query that retrieves the data used to create a presentation.

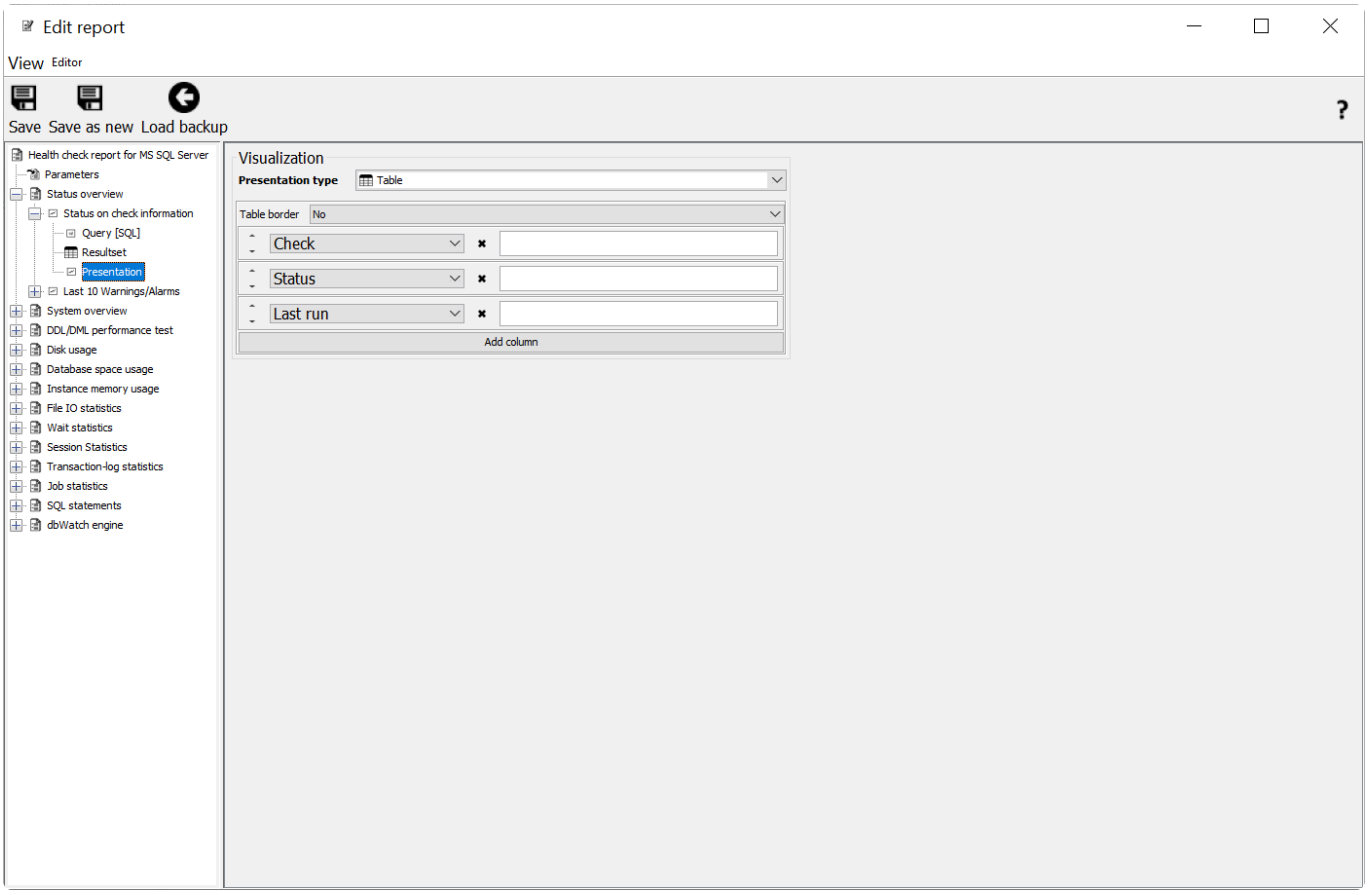
The task editor allows you to execute the query to verify that it has the correct syntax and that it returns the data you expected.



When you click on **Resultset** you see a definition of the colums and datatypes the query produces. If this is empty, you can click on “Export to resultset” in the query view after you have generated a sample resultset by executing the query, and the resultset definition will be automatically genereated.



When you click on **Presentation** you can define how you want the result presented. The available presentation types are: **Table**, **List**, **Category Chart – Series**, **Category Chart**, **Pie Chart** and **Dual Axis Chart**



<< Scheduled reports / Customization >>

Customization

dbWatch Control Center is designed to be customizable and much of the contents are developed inside the product

In this section, we will go over how you can customize different parts of the product. To proceed, click on of the topics below:

1. Customize dbWatch Views or dashboards using FDL
2. Customize Reports in dbWatch Control Center
3. Customize or create new engine jobs
4. Extending the FDL language with properties, both dynamic and static
5. Creating and customizing web dashboards
6. The management module is a set of XML scripts that can be extended and altered. There is currently no graphical interface for editing this, but new XML files can be added using resource import.

dbWatch can help our customers in customizing their views. Just email our support team and we'll set up a call to clarify what you need. Similarly, customizing views can be done by the end user.

To import new files into the product look into the importing resources section

Underlying most of the product is data retrieved using FDL (Farm Data Language) queries. FDL queries on a graph structure of properties that allow us to create cross-database and cross-platform queries. It is possible for users to add and extend the property system, with metadata and dynamic properties derived from SQL, javascript, or other sources.

Monitoring can be customized by creating monitoring jobs. There are currently two types of jobs.
Engine jobs, which are installed into the database instance, consist of a part of native database program code and accompanying tables.
No-engine jobs, which are based on the property system and consist of one or more queries (SQL or FDL) with program code in javascript.

There is currently only a graphical interface for editing engine jobs.

You can check out our blogs for a more in-depth look at how to customize your dbWatch views:

- Create a customized database performance view in dbWatch
- Database monitoring: Customizing your views using Farm Data Language to make your life easier

<< Edit Report Templates / Importing resources >>

Importing resources

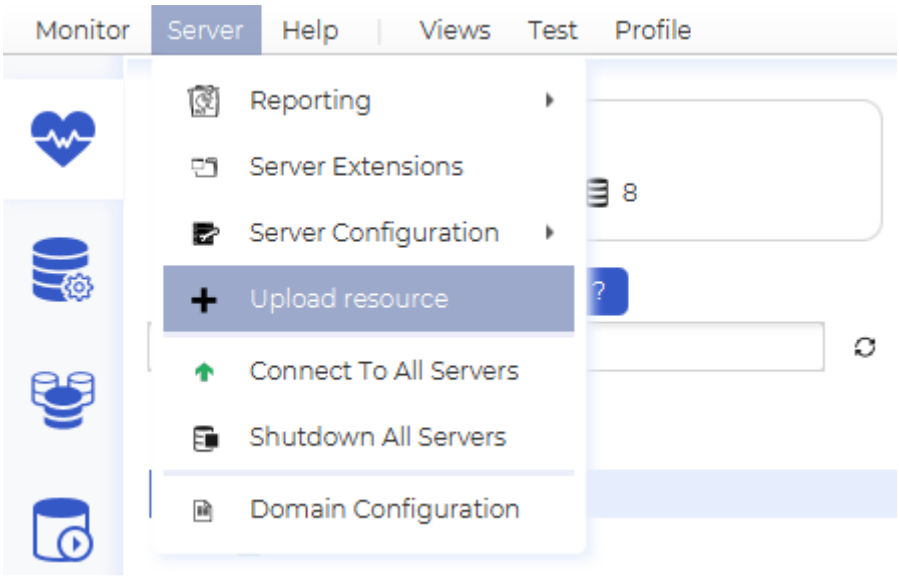
Much of the functionallity in dbWatch is defined in xml files, including report specifications, task specifictions and management specifications.

That means that functionallity can be modified by supplying new xml files to dbWatch.

There are two ways of doing this.

Through the dbWatch Monitor.

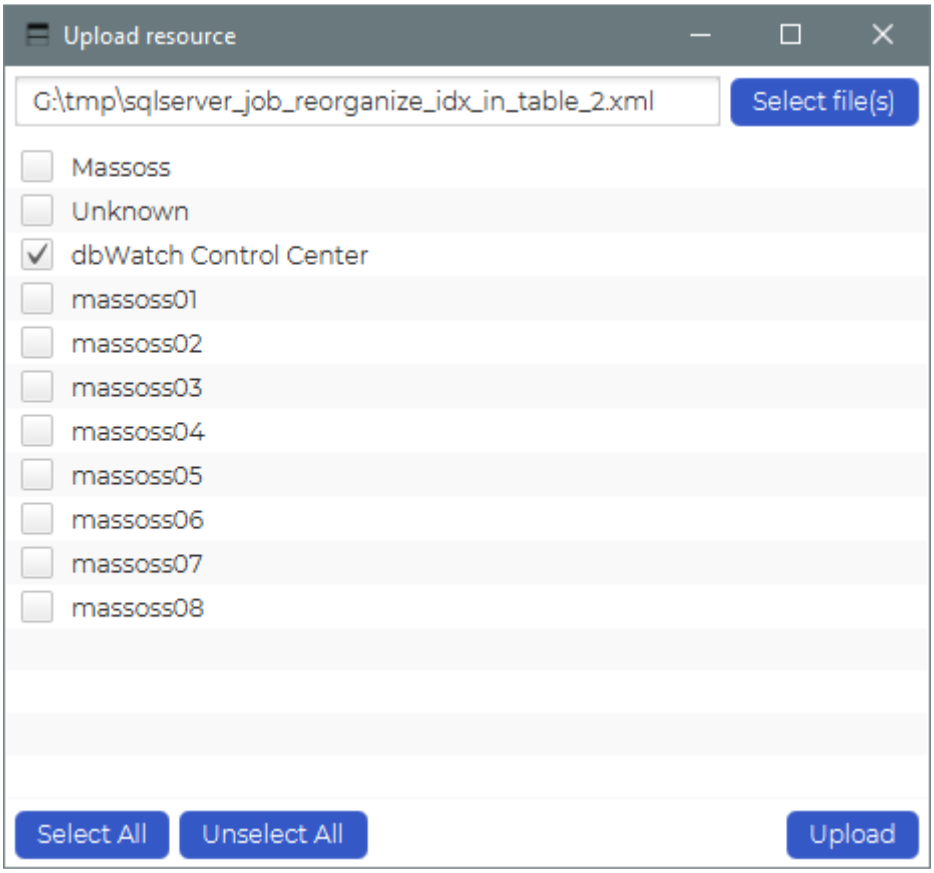
On the Server menu, you can click on “Upload resource”.



This brings up a dialog where you can select a file to upload, and what dbWatch Servers to upload it to.

> ! Only CSV or XML files can be uploaded.

In our example we have a dbWatch Server registered.



To upload a file, simply click “Select file” (or type/paste the complete path manually). Select the desired Servers, and click “Upload”.

This is the preferred way to import single files to many dbWatch Servers.

Through the file system.

Find the desired dbWatch Server workarea on the file system. (On a standard windows installation this should be “C:/ProgramData/dbWatchControlCenter”.)

Then locate the import area catalog located here: “[dbWatch workarea]/data/resources/import_area”

Copy the xml/csv file(s) into this catalog, and the dbWatch Server will automatically import them.

This is the preferred way for bulk imports.

<< Customization / Monitoring >>

Monitoring

Custom monitoring and editing monitoring jobs

dbWatch has two types of monitoring jobs that can detect issues in the database instances. They are referred to as Engine jobs and “No-engine jobs”.

Engine jobs are jobs that run inside the database engine, and consists of at least one procedure in the native database programing language, and usually consists of additional tables for historical data. There is a built-in editor for engine jobs.

No-engine jobs as jobs that run on the dbWatch Server, and typically run a SQL statement and process the result of this statement in javascript. There is no built-in editor for this type of job, but its possible to edit in a normal editor and upload the file into dbWatch. Examples and more information here.

Editor for engine jobs

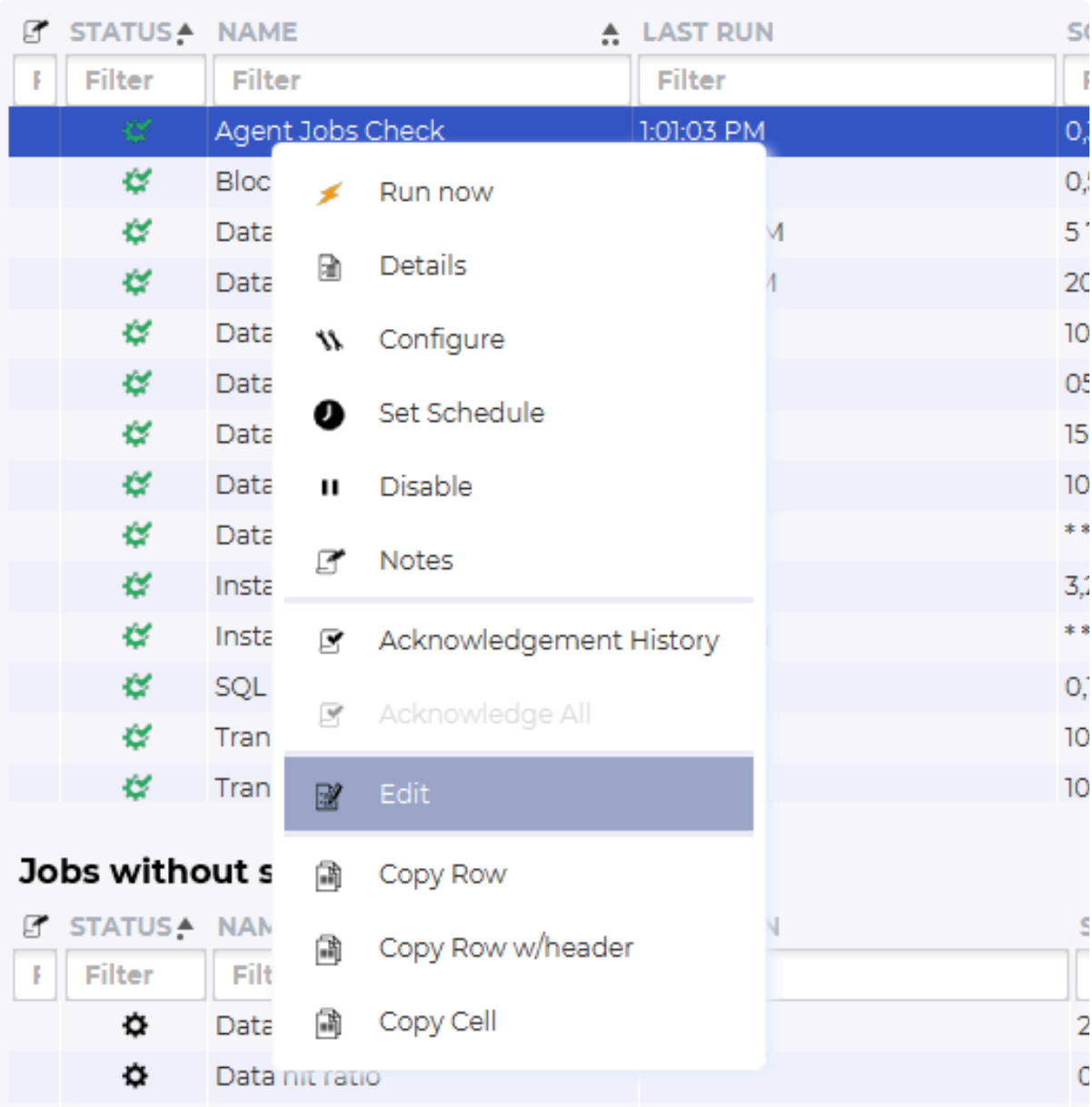
How to get here

Example jobs

We have some example jobs to make the development easier here.

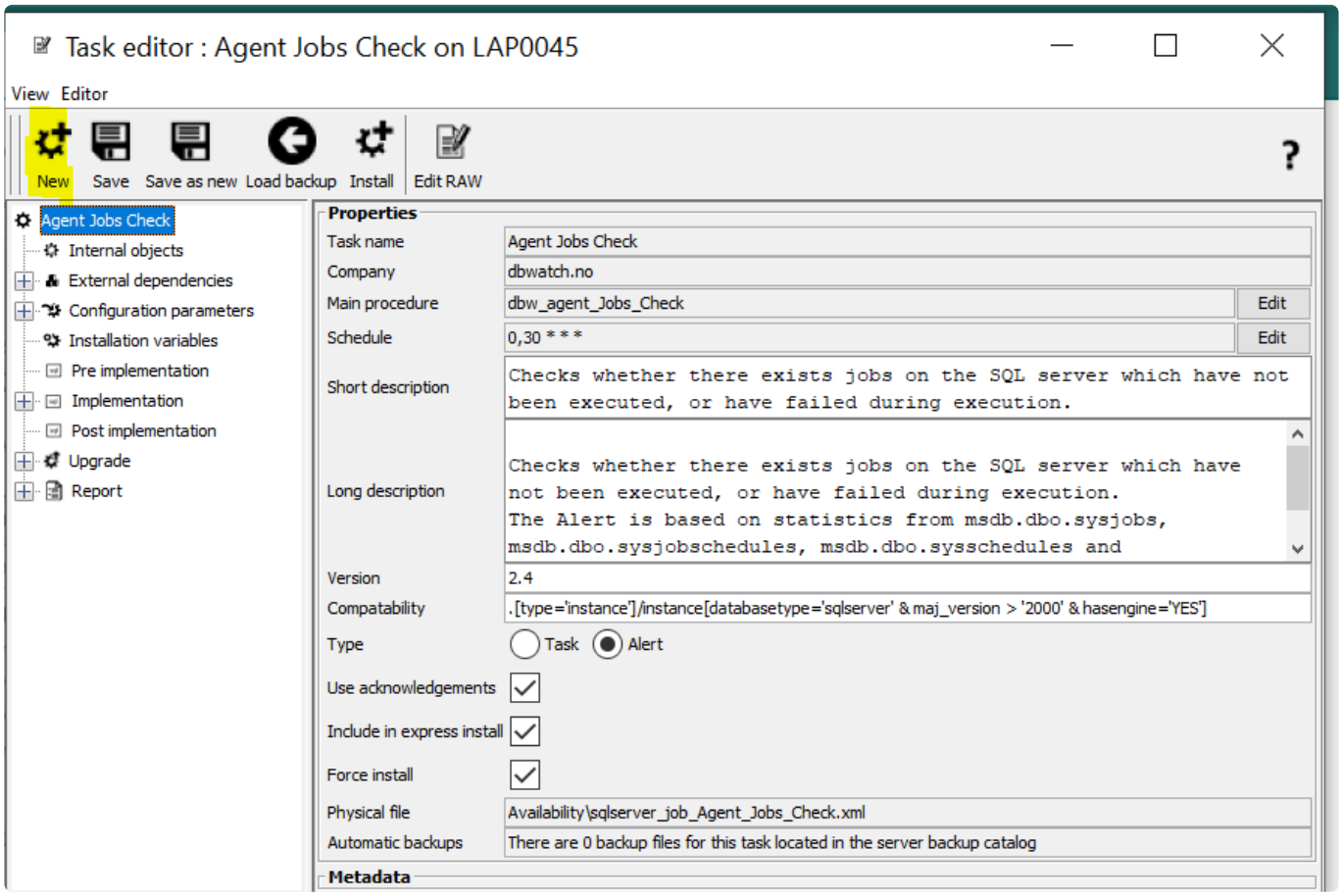
Edit Existing

To edit an existing Task or Alert open the Task/Alert Editor by right clicking on any task or alert and select **Edit task** from the popup menu.

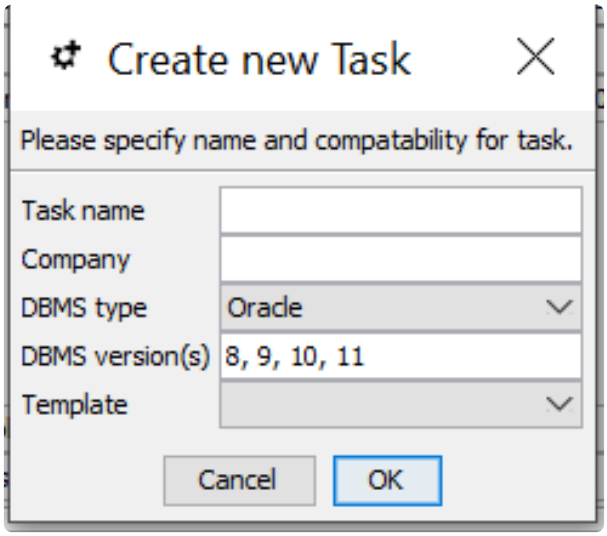


Create new

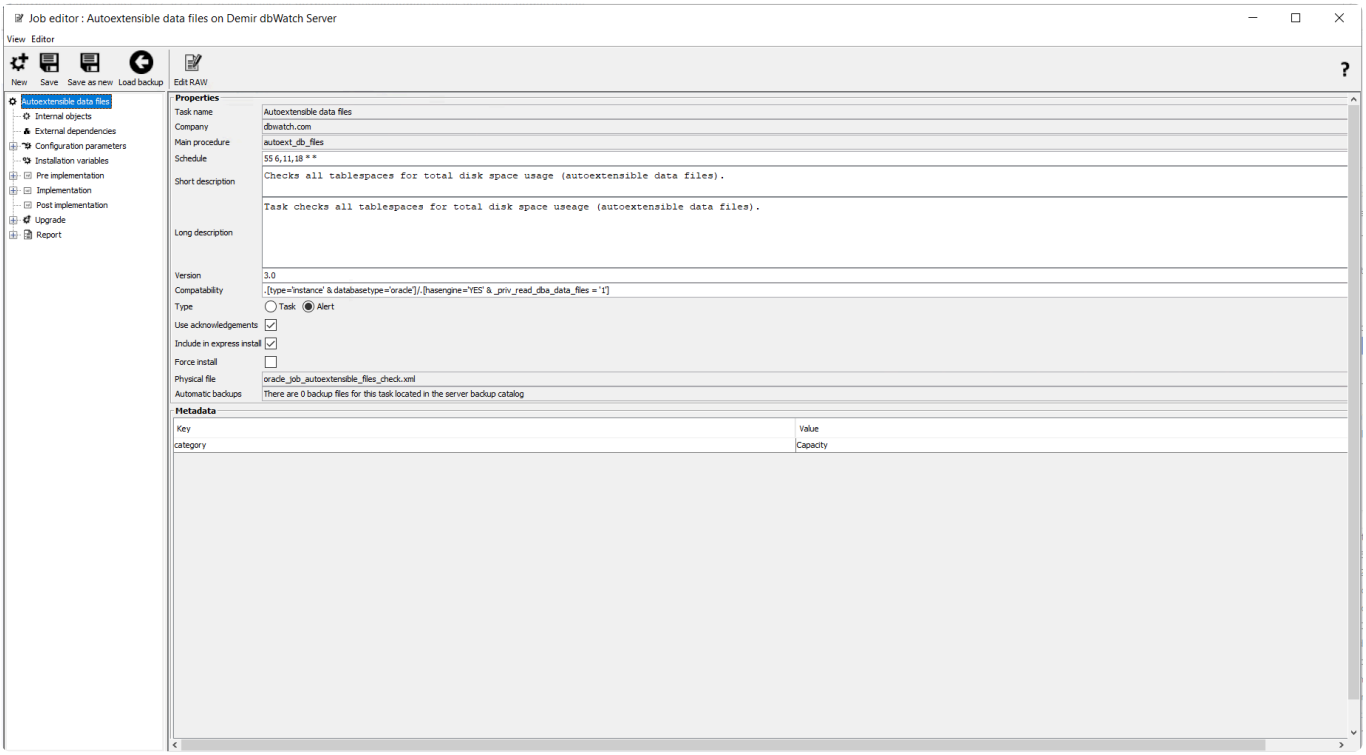
To create a new Task or Alert from scratch, open an existing job in the editor and click “New”.



You will then be prompted with a dialog that lets you fill in the basic details of the new Task or Alert.



Developing your own engine jobs



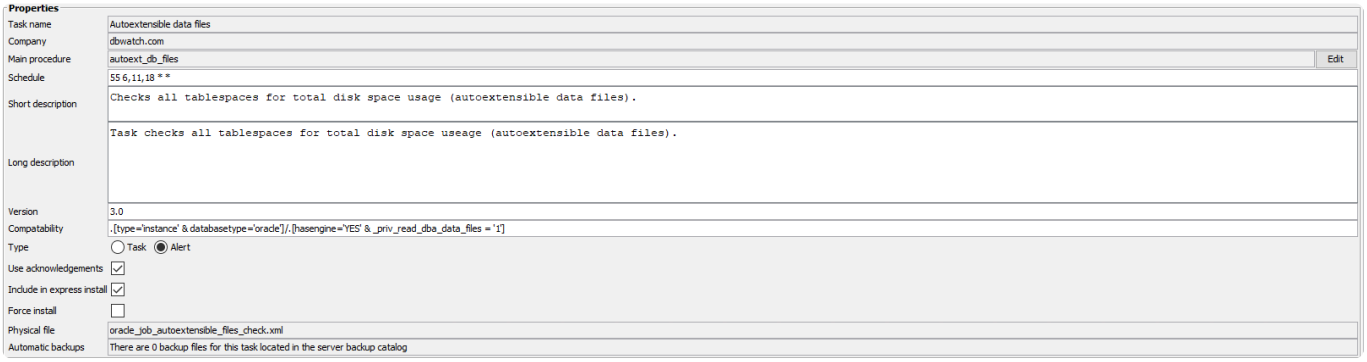
The editor has a tree structure and the top level has two areas:
Properties this is where the general settings for the job are configured.
Metadata this is metadata the job will set on its own object once installed.

The underlying nodes are:
Internal objects this is a mapping of objects this job creates.
External dependencies are dependencies outside this job, to other tables, functions procedures, or views.
Configuration parameters are the parameters that a user can use to adjust thresholds. We can access the parameter values set here from inside the program code.
Installation variables are used for input variables prompted on job installation.
Pre implementation is used for objects or grants that needs to be implemented, often by another user, before implementation steps.
Implementation is the main step where tables, views, functions and procedures for a job is located.
Post implementation is used for steps after the implementation, such as recompilation of procedures
Upgrade is sets of implementation steps to upgrade from one or multiple versions to the current version
Report is the report available when you click on “Details” in the job menu.

In most jobs the developer only needs to focus on the sections:
Properties
Configuration parameters
Implementation
and optionally:
Report

<< Monitoring / Properties >>

Properties

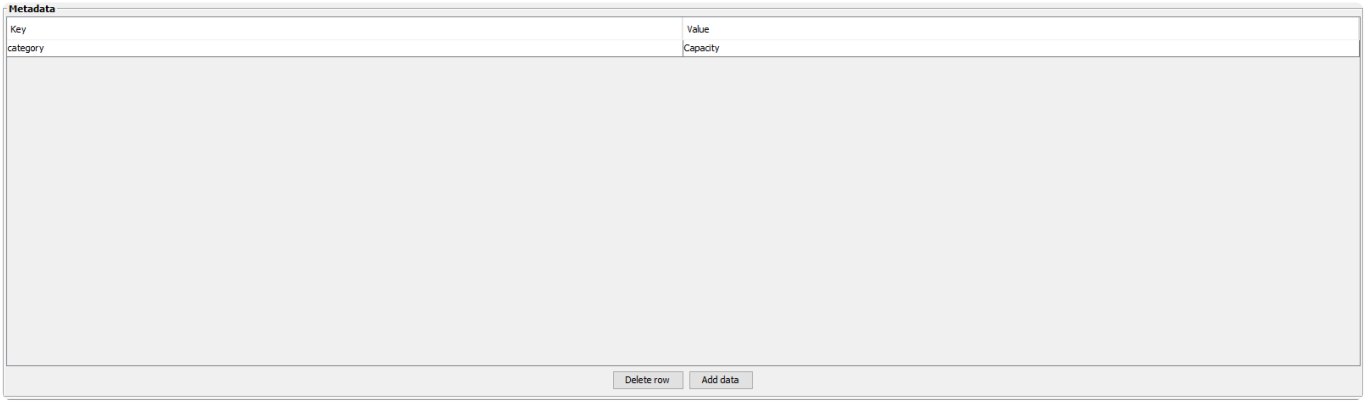


The properties section defines the main settings for the job. Once a job is created the “Task name”, “Company” and “Compatibility” is specified.
“Task name” is the visible job name when you install a job or when its visible in monitoring.
“Company” is the domain name of the company that creates this job.

“Main procedure” is the procedure or function that gets triggered by Control Center, when it runs the job.
“Schedule” is the default schedule this job.
“Short description” is a one-line description of what the job monitors.
“Long description” is a more verbose description of what the job monitors.
“Version” is current version of the job.
“Compatibility” is what database platforms, versions or type this job support. The compatibility is specified as an FDL query with filters, where only compatible instances are left once filters are applied.
“Type” is set to task or alert. Task is jobs without status and alert is jobs with status.
“Use acknowledgments” indicates if this job will use the acknowledgment system
“Include in express install” indicates if this job is pre-selected to be installed when you add a new instance.
“Force install” indicates that objects in the internal objects list will be dropped if it already exists on install.
“Physical file” is the physical file name of the job. All jobs on Control Center is wrapped in XML. You can open the actual XML file with the “Edit RAW” button in the editor, but making changes in RAW edit requires expert knowledge. There are however backups.
“Automatic backups” counter of the number of automatic backups available. You can reset back to an older backup with the “Load backup” button.

<< Editor for engine jobs / Metadata >>

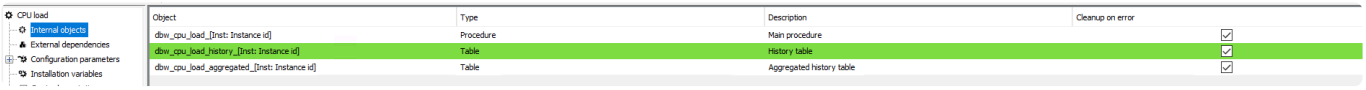
Metadata



Metadata is used to set metadata properties on the job once installed. Normally this is where the job grouping is set.

<< Properties / Internal objects >>

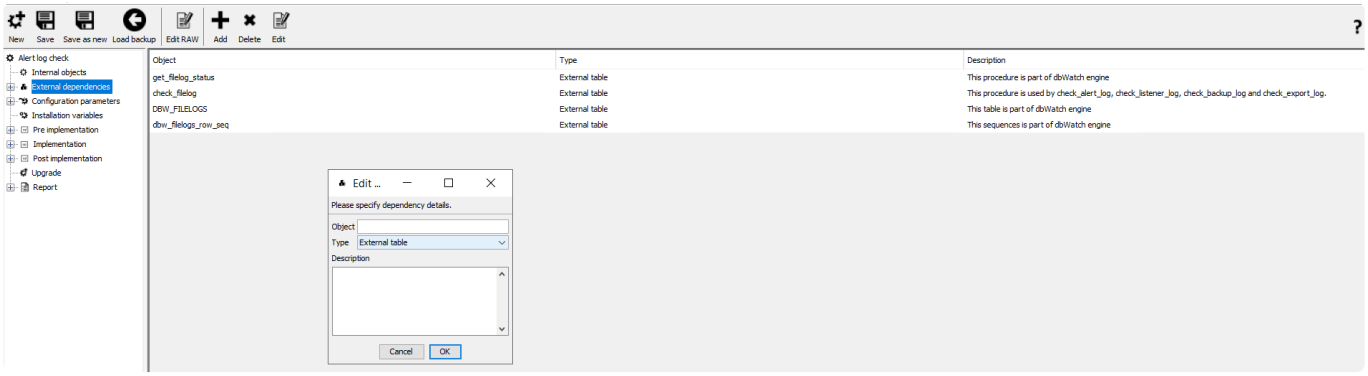
Internal objects



Internal objects are generated once you add objects in the implementation section. You can however adjust if the objects are dropped if the job installation fails, or stay present after any cleanup attempt.

<< Metadata / External dependencies >>

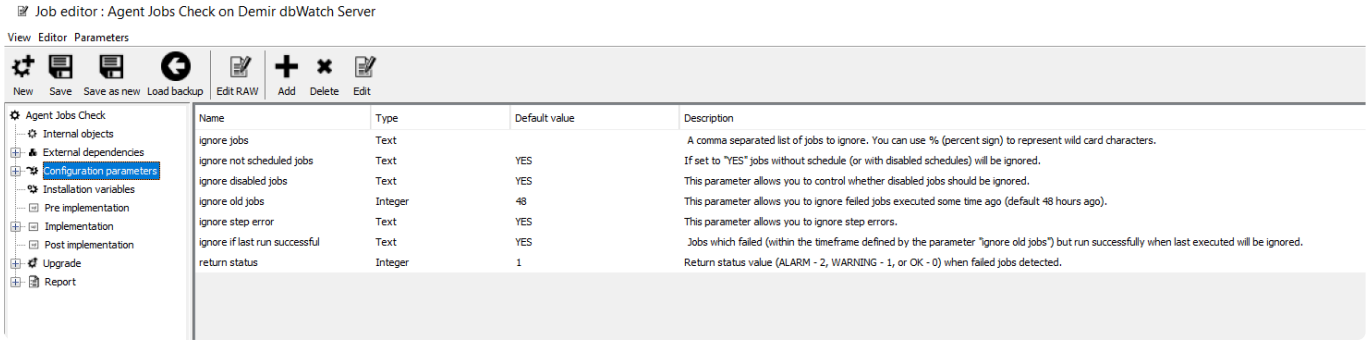
External dependencies



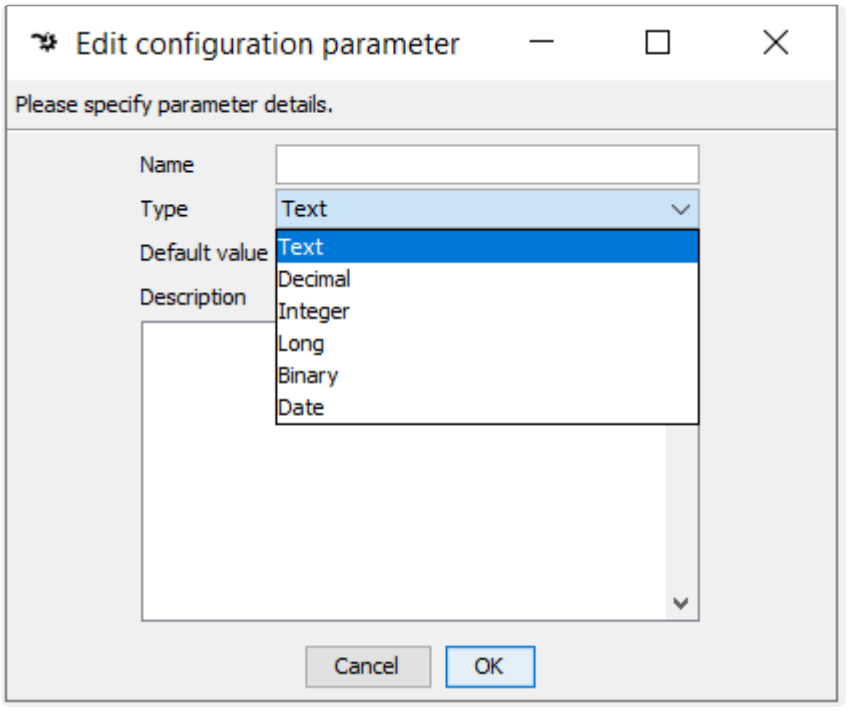
External dependencies are used to add dependencies to external tables, views, functions, and procedures. External in this setting is external to, as not part of, this job. It can be external objects, such as tables and procedures that the procedures in this job depend on, to enable them to compile properly. It is not required but used in a few routines that require specific parts of the framework routines to function.

[<< Internal objects](#) / [Configuration parameters >>](#)

Configuration parameters



Configuration parameters are the parameters a user can use to control the behavior of a job. This can be thresholds for alerting, items to ignore, and other types of adjustments that make it easy to create a good fit between a standardized job and a custom database instance setup.
The parameters are written to the dbw_parameters table in the dbwatch_cc (by default) schema/ database and the procedure can retrieve this data upon execution and use it in the internal logic.

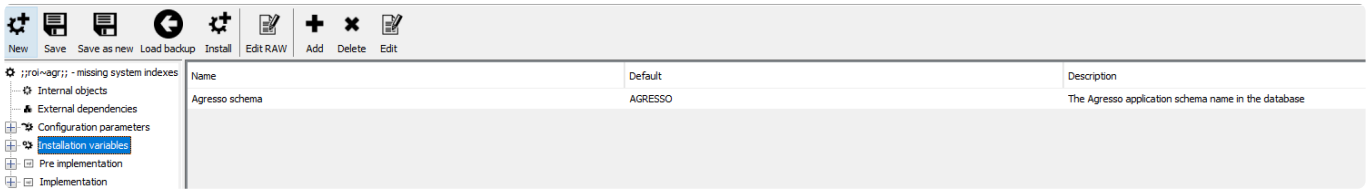


Parameters can be in the following formats:
Text, Decimal, Integer, Long, Binary, and Date.

All parameters need a name and a type. “Default value” and “Description” are recommended optional values.

[<< External dependencies] / [Installation variables >>]

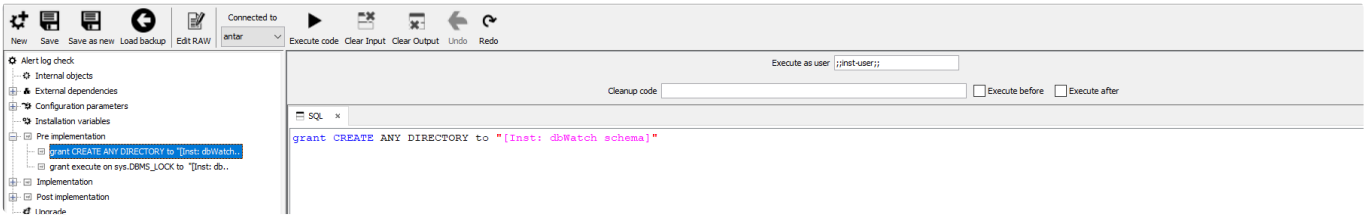
Installation variables



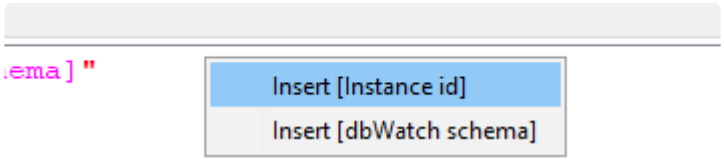
Installation variables are used if you want user input during job installation. It is mostly used for application-specific monitoring where the application schema or database can not be determined correctly by the job after installation and human input are necessary.

[<< Configuration parameters](#) / [Pre implementation >>](#)

Pre implementation



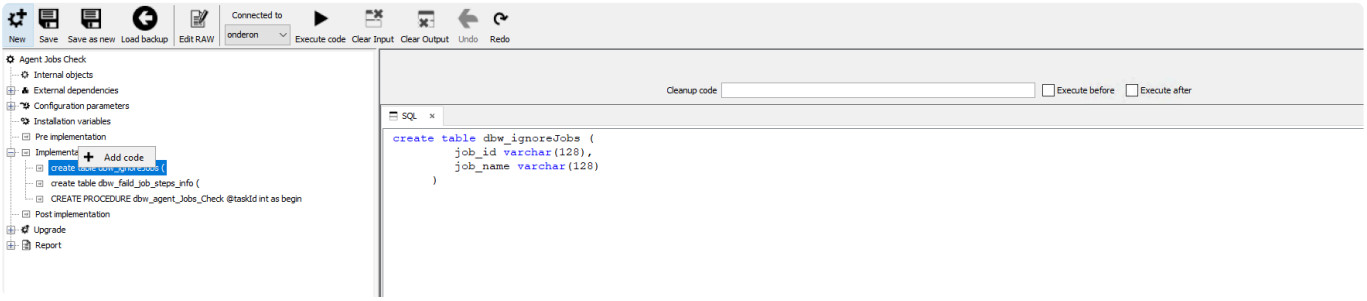
“Pre implementation” is used to issue grants or create objects as another user, before the implementation step. Its optional, and in this example used to issue grants that is needed by the implementation procedure.
“Execute as user” is the user used to execute the command. The variable “;;inst-user;;” is substituted to the privileged user used when this instance was added to dbWatch Control Center.
This example grant also uses “[Inst: dbWatch schema]” which is substituted on execution with the schema/login name of the dbWatch Control Center user, by default dbwatch_cc.
You can also add a cleanup code. This can be for dropping objects etc. It can be executed before or after the main statement.



The substitutions are available when you right-click in the editor.

[<< Installation variables] / [Implementation >>]

Implementation



The implementation steps are the most important section. It consists of code blocks that creates the contents of the job. Typically at least one table and a procedure.
You can right-click on the “Implementation” tree element to add new code blocks. You can use drag-and-drop to position the code elements as they are executed in the order they appear in the interface.

Code elements like create table and view is quite straight forward. The main procedure has some elements that is mandatory and different for each platform, and is covered in a sperate subsection.
The cleanup code is mostly used as part of development, as you can execute code blocks on a spesific database instance. You can choose the database instance in the “Connected to” pulldown menu. “Execute code” will execute this code block on that database instance.
In the case of a create table statement, it can help development to add a drop table statement in the cleanup code, and hook the execute before checkbox. This will allow you to recreate the table until you are happy with the layout. It is not common to keep the cleanup code in these code blocks after development, as this type of functionality is handled by the “Force install” or “Cleanup” checkboxes in normal use.

<< Pre implementation / Main procedure SQL Server >>

Main procedure SQL Server

We will use the “Test alert” as example code, as its only purpose is to generate alarms and warnings, but uses many underlying functions. We go through it in steps, and include the full routine at the end.
For historical reasons some code will refer to terms like “task” or “check”. Task is jobs without status, and check is jobs with status.
As checks are an expansion of a task, both tasks and checks are found in the dbw_tasks table, but only checks are in the dbw_checks table. The dbw_checks table only keeps additional data not stored in dbw_tasks.

```
create  procedure dbw_test_alert @taskID INT
```

The main procedure gets the taskid as input when it gets executed. This is used to lookup in the underlying framework to get hold of parameters and updating status.

```
as
begin
set nocount on
declare @status int
```

It is typical to declare an int value for the return status. If this job wants to update status, this argument needs to be part of the call of the dbw_updateCheckValues procedure. 0 is OK, 1 is Warning and 2 is Alarm.

```
declare @now_date datetime
declare @exec_info VARCHAR(1000)
```

At the end of this procedure we call “dbw_updateCheckValues `taskID`, status, @exec_info”. This routine writes data in the underlying framework. This data is then again displayed in the software, and used for alerts.
We declare the exec_info variable. The data in this variable will end up as the details field on the job in the monitoring module.

We continue with general code for this procedure..

```
declare @last_run_alarm datetime
declare @between_alarm int
declare @send_alarm int
declare @alarm_message varchar(1000)
declare @alarm_disable int
declare @last_run_warning datetime
declare @between_warning int
declare @send_warning int
declare @warning_message varchar(1000)
```

```
declare @warning_disable int
declare @ok_message varchar(1000)
```

These are just other declarations for the function of this job.

```
BEGIN TRY
```

This is to run the procedure with exception handling

```
set @status = 0
set @now_date = getdate()
set @send_alarm = 0
set @alarm_disable = 0
set @send_warning = 0
set @warning_disable = 0
```

These is some variables used in the function of this job.

```
select @between_warning = convert(int, VALUE) FROM DBW_PARAMETERS WHERE UPPE
R(NAME)=UPPER('Time between warnings') and TASK_ID=@taskID
select @between_alarm = convert(int, VALUE) FROM DBW_PARAMETERS WHERE UPPER(NA
ME)=UPPER('Time between alarms') and TASK_ID=@taskID
select @alarm_message = VALUE FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Ala
rm message') and TASK_ID=@taskID
select @warning_message= VALUE FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Wa
rning message') and TASK_ID=@taskID
select @ok_message = VALUE FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Ok mes
sage') and TASK_ID=@taskID
select @warning_disable = convert(int,CASE VALUE WHEN 'NO' THEN '0' WHEN 'YE
S' THEN '1' ELSE '0' END)
FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Disable warnings') and TASK_ID=@t
askID
select @alarm_disable =convert(int, CASE VALUE WHEN 'NO' THEN '0' WHEN 'YES' T
HEN '1' ELSE '0' END)
FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Disable alarms') and TASK_ID=@tas
kID
```

This is how we get data from parameters. The configurable parameters in the job, are stored in the framework in the “DBW_PARAMETERS” table. We retrieve values from this by using the parameter name and the task_id we got at execution time. “DBW_TASKS” table provides the mapping between task_id and the name of the job.

```
select @last_run_alarm = max(histr_date) from dbw_test_alert_history where typ
e_message like @alarm_message
```

```
select @last_run_warning = max(histr_date) from dbw_test_alert_history where t
ype_message like @warning_message
if @last_run_alarm is null set @last_run_alarm = getdate() - 9999
if @last_run_warning is null set @last_run_warning = getdate() - 9999
                        if @now_date > @last_run_warning + @between_warning/2
4./60.
                        begin
                        if @send_alarm = 0
                        begin
                                if @warning_disable = 0
                                begin
                      set @send_warning = 1
                                end
                        end
         end
                        if @now_date > @last_run_alarm + @between_alarm/24./6
0.
         begin
                        if @send_warning = 0
                        begin
                                if @alarm_disable = 0
                                begin
                      set @send_alarm = 1
                                end
                        end
         end

        if @send_warning > 0
        begin
                insert into dbw_test_alert_history (histr_date, type_message)
values (@now_date, @warning_message)
                set @status = 1
                set @exec_info = @warning_message
        end
        if @send_alarm > 0
        begin
                insert into dbw_test_alert_history (histr_date, type_message)
values (@now_date, @alarm_message)
                set @status = 2
                set @exec_info = @alarm_message
        end

        if @send_warning = 0
        begin
                if @send_alarm = 0
                begin
                        set @status = 0
```

```
                        set @exec_info = @ok_message
                end
        end


                        delete from dbw_test_alert_history where histr_date <
getdate() -30
```

This is the main function section of the job.

```
exec dbw_updateCheckValues  @taskID, @status, @exec_info
```

This call is essential to update the result of the job. dbw_updateCheckValues takes the task_id(which we get as input when this procedure is executed), an int, here status (which can be 0 for OK, 1 for Warning, and 2 for Alarm), and a string, here exec_info where we add short detailed information of the status of the job. Keep this short, under 1000 characters.

```
END TRY
BEGIN CATCH
set @exec_info = 'Exception, ' + ERROR_MESSAGE()
exec dbw_updateCheckValues  @taskID, 0 , @exec_info
END CATCH
end
```

It is recommended, for stability, to add exception handling to the job, in case something goes wrong. Here we try to add some information about the exception, as part of the details, so it will be visible in the graphical interface.

That is the layout of a main procedure on MS SQL Server. All the jobs provided are open, and its possible to read the code to get inpiration to new jobs.

Complete code

```
create  procedure dbw_test_alert @taskID INT
as
begin
set nocount on
declare @status int
declare @now_date datetime
declare @exec_info VARCHAR(1000)
declare @last_run_alarm datetime
declare @between_alarm int
declare @send_alarm int
declare @alarm_message varchar(1000)
declare @alarm_disable int
```

```
declare @last_run_warning datetime
declare @between_warning int
declare @send_warning int
declare @warning_message varchar(1000)
declare @warning_disable int
declare @ok_message varchar(1000)

BEGIN TRY

set @status = 0
set @now_date = getdate()
set @send_alarm = 0
set @alarm_disable = 0
set @send_warning = 0
set @warning_disable = 0

select @between_warning = convert(int, VALUE) FROM DBW_PARAMETERS WHERE UPPE
R(NAME)=UPPER('Time between warnings') and TASK_ID=@taskID
select @between_alarm = convert(int, VALUE) FROM DBW_PARAMETERS WHERE UPPER(NA
ME)=UPPER('Time between alarms') and TASK_ID=@taskID

select @alarm_message = VALUE FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Ala
rm message') and TASK_ID=@taskID
select @warning_message= VALUE FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Wa
rning message') and TASK_ID=@taskID
select @ok_message = VALUE FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Ok mes
sage') and TASK_ID=@taskID

select @warning_disable = convert(int,CASE VALUE WHEN 'NO' THEN '0' WHEN 'YE
S' THEN '1' ELSE '0' END)
FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Disable warnings') and TASK_ID=@t
askID
select @alarm_disable =convert(int, CASE VALUE WHEN 'NO' THEN '0' WHEN 'YES' T
HEN '1' ELSE '0' END)
FROM DBW_PARAMETERS WHERE UPPER(NAME)=UPPER('Disable alarms') and TASK_ID=@tas
kID

select @last_run_alarm = max(histr_date) from dbw_test_alert_history where typ
e_message like @alarm_message
select @last_run_warning = max(histr_date) from dbw_test_alert_history where t
ype_message like @warning_message


if @last_run_alarm is null set @last_run_alarm = getdate() - 9999
if @last_run_warning is null set @last_run_warning = getdate() - 9999

                        if @now_date > @last_run_warning + @between_warning/2
```

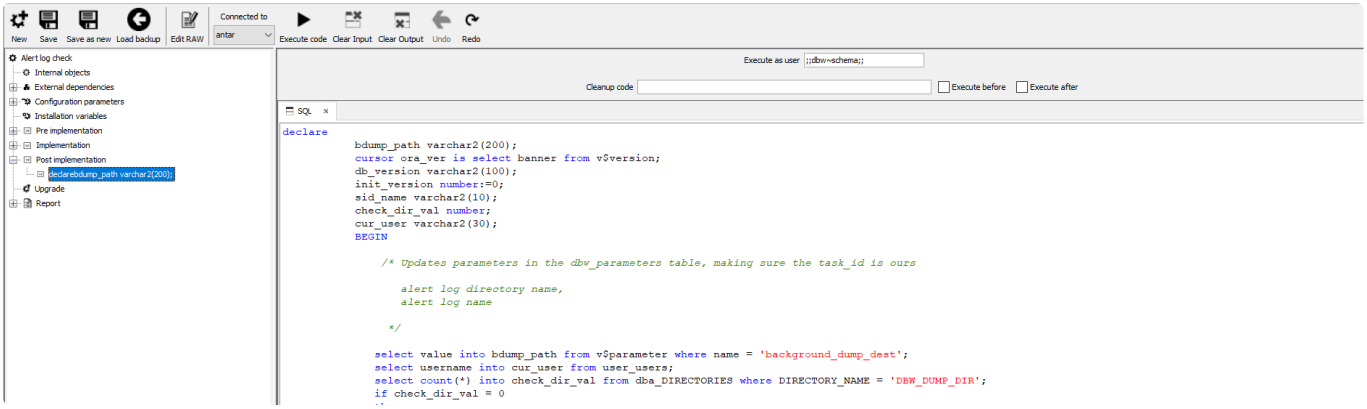
```
4./60.
                        begin
                        if @send_alarm = 0
                        begin
                                if @warning_disable = 0
                                begin
                    set @send_warning = 1
                                end
                        end
         end
                        if @now_date > @last_run_alarm + @between_alarm/24./6
0.
         begin
                        if @send_warning = 0
                        begin
                                if @alarm_disable = 0
                                begin
                    set @send_alarm = 1
                                end
                        end
         end

        if @send_warning > 0
        begin
                insert into dbw_test_alert_history (histr_date, type_message)
values (@now_date, @warning_message)
                set @status = 1
                set @exec_info = @warning_message
        end
        if @send_alarm > 0
        begin
                insert into dbw_test_alert_history (histr_date, type_message)
values (@now_date, @alarm_message)
                set @status = 2
                set @exec_info = @alarm_message
        end

        if @send_warning = 0
        begin
                if @send_alarm = 0
                begin
                        set @status = 0
                        set @exec_info = @ok_message
                end
        end
```

```
        delete from dbw_test_alert_history where histr_date < getdate() -30

exec dbw_updateCheckValues  @taskID, @status, @exec_info

END TRY
BEGIN CATCH
set @exec_info = 'Exception, ' + ERROR_MESSAGE()
exec dbw_updateCheckValues  @taskID, 0 , @exec_info
END CATCH
end
```

End of procedure

[<< Implementation](#) / [Post implementation >>](#)

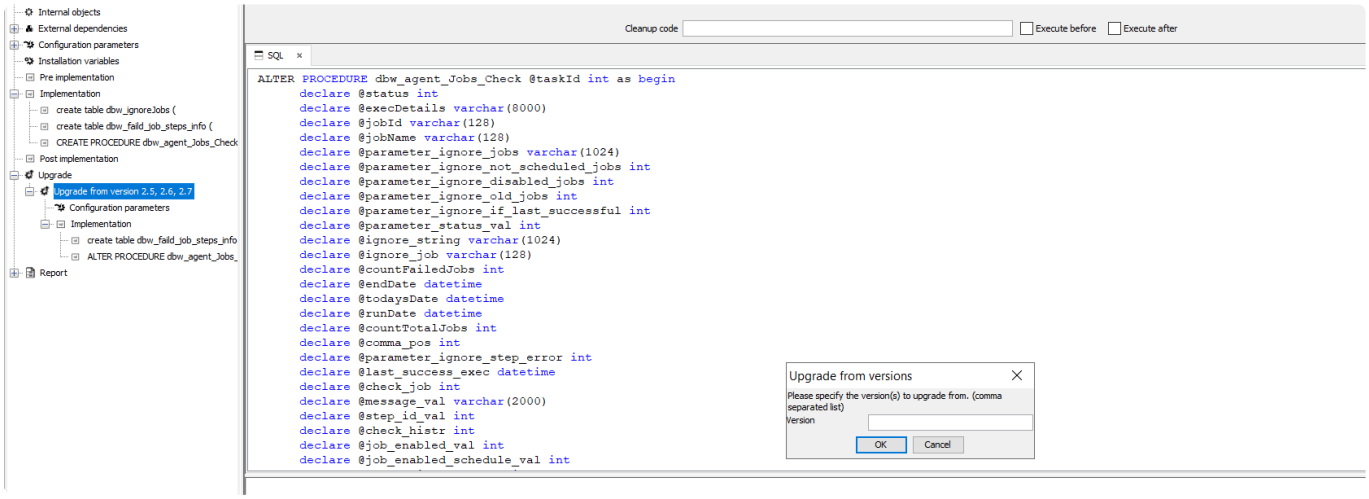
Post implementation



“Post implementation “is used to do actions after implementation steps. It can be recompiling code, fixing issues in data, or adding data to tables. In this example, it is a procedure that goes through the parameters configured and makes sure they are all set correct. Its not mandatory and is used in a fraction of the jobs.

<< Main procedure SQL Server / Upgrade >>

Upgrade



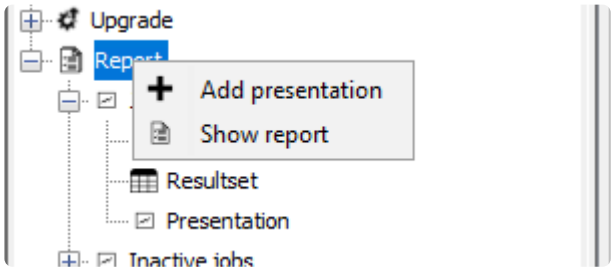
When you add upgrade/implementation steps you get prompted for “Version”, and that is a comma-separated list of versions that this implementation can upgrade from. It is always to the current version. You can use the configuration parameters to create new parameters. This is for parameters that are part of the current version but were not part of the version(s) you list in “Version”.

The implementation part of the upgrade follows the same syntax and setup of the main Implementation, but if tables were already created in older versions of the job, they are typically omitted in the upgrade implementation.

<< Post implementation / Report >>

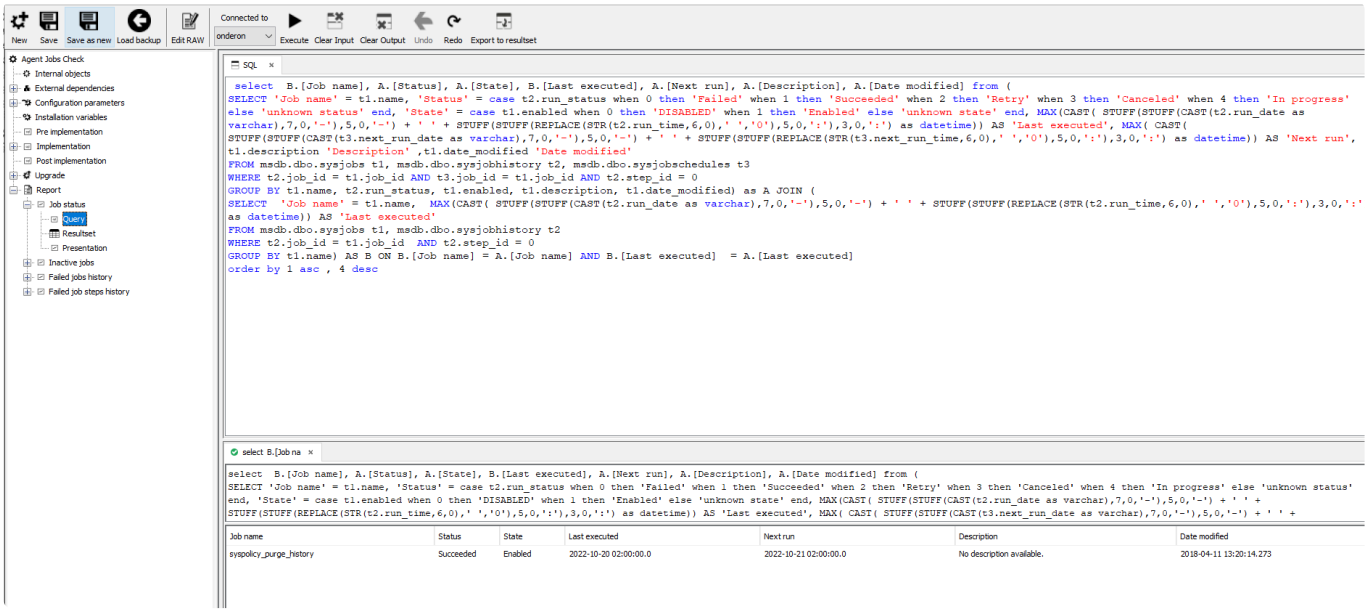
Report

The report section of a job is a lightweight version of the reporting engine in Control Center. It supports SQL queries that can be used in multiple output formats, such as graphs and tables.

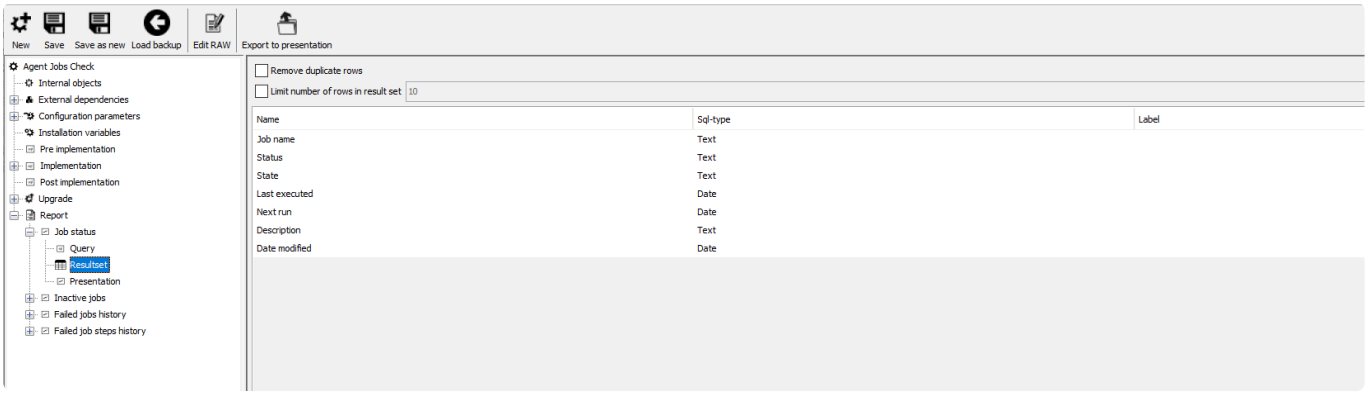


You can right-click on the “Report” item, add new sections to the presentation with the “Add presentation” button. Its also possible to run the report with the “Show report” button, which opens the report output in an other window.

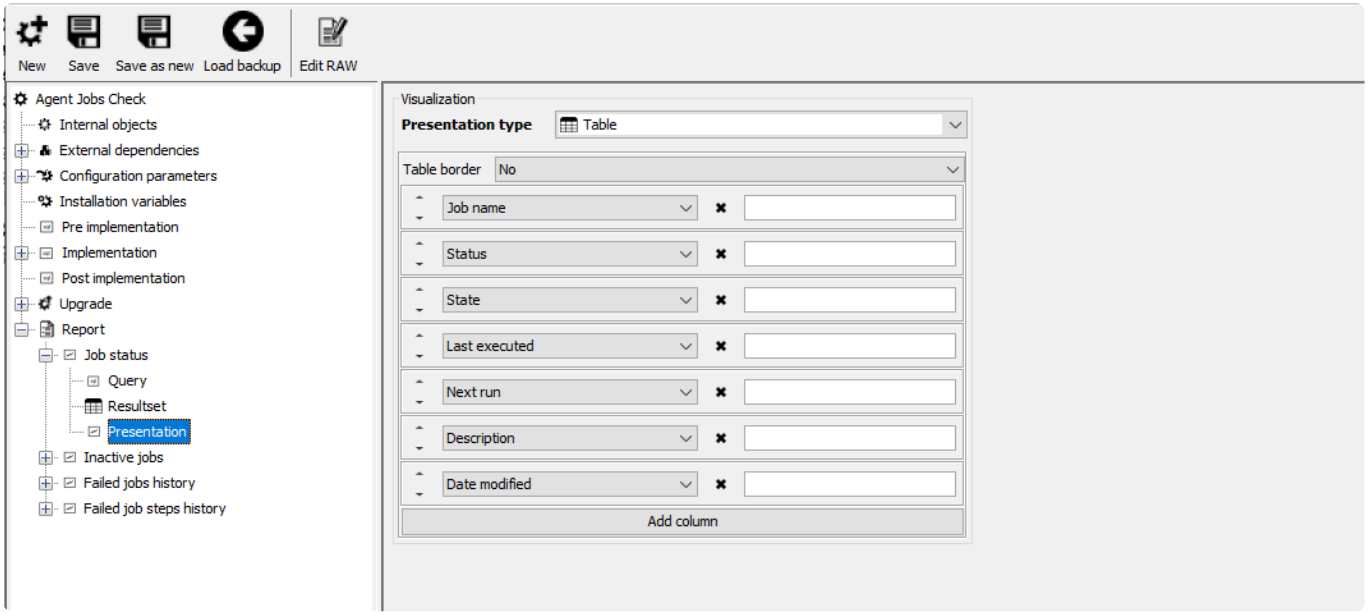
The first section is where you can set the section title, and a text that will come above the graph or table presented. It is there to add context and provide the user information of what data is displayed in the graph or table below.



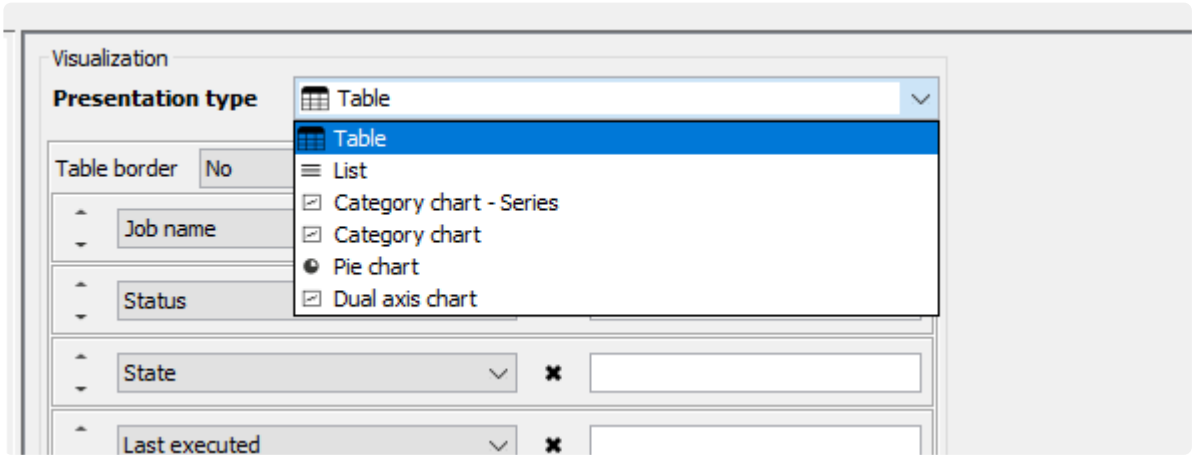
The “Query” section is where you can put in the SQL query that you will run. Typically the report is developed after the main procedure of the job and the underlying tables are created, (and installed on a database instance), and the report will query that data. This will also make the report development more streamlined, as you can then run the query (select instance with “Connected to”, and “Execute” to run), this will then display below, and you can see that you have the correct data to move forward. Additionally this gives the editor information about what datatypes to expect, aiding in the next step of the process. Once you have the correct data, click on “Export to resultset” (note that this will overwrite any changes you have made to the resultset section).



In the “Resultset” section, we can adjust datatypes (this will help format numbers if needed), add labels to the fields, remove duplicate rows and limit resultset rows. The resultset limit is done post-database-processing, so the full query will run on the database side.
Once happy, you can click on “Export to presentation”. This changes the presentation section, so don’t do this if you have done many changes to the presentation section. Its usually done the first time to get the right column name populated.



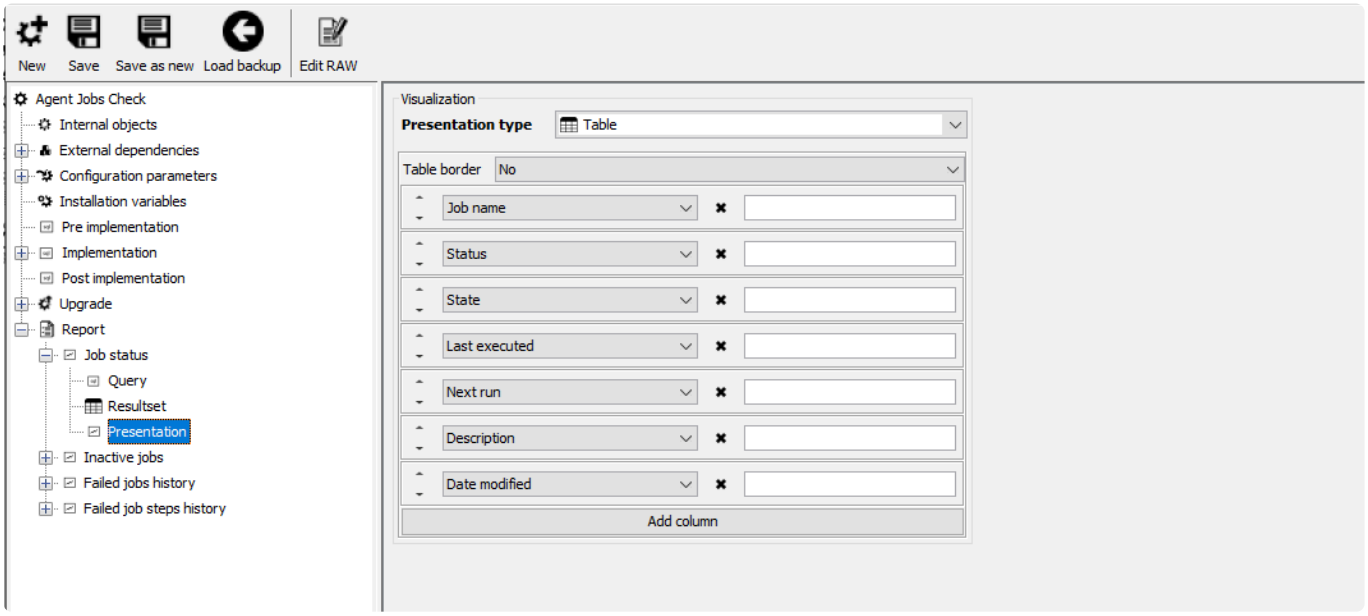
In the “Presentation” section you can present the data in different ways. The default is the table view. You can set if you want a border around the table, and order of, selection of, and label on, the column values from the result set.



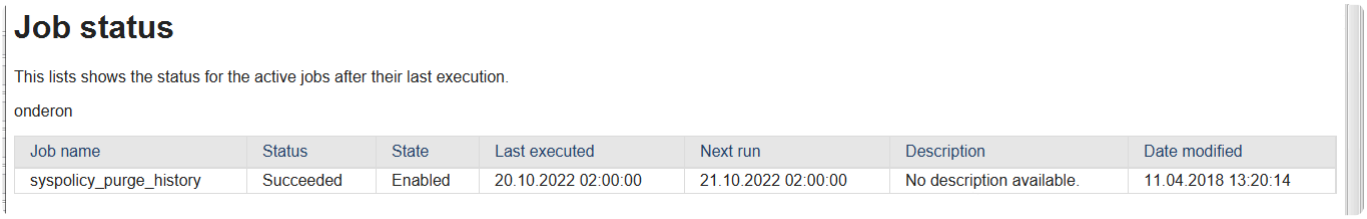
There are six presentation types:

“Table”, the default, to create a table with contents.
“List”, is best for text values, one column that should be displayed like a file. Such as output from a logfile etc.
“Category chart – Series”
“Category chart”
“Pie chart”
“Dual axis chart”

Table format



Definition



Job status

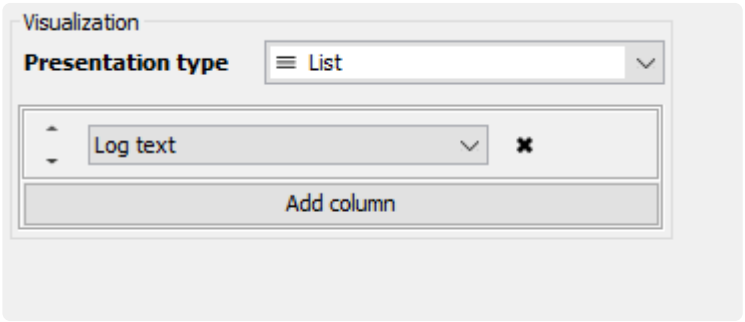
This lists shows the status for the active jobs after their last execution.

onderon

Job name	Status	State	Last executed	Next run	Description	Date modified
syspolicy_purge_history	Succeeded	Enabled	20.10.2022 02:00:00	21.10.2022 02:00:00	No description available.	11.04.2018 13:20:14

Result

List



Definition

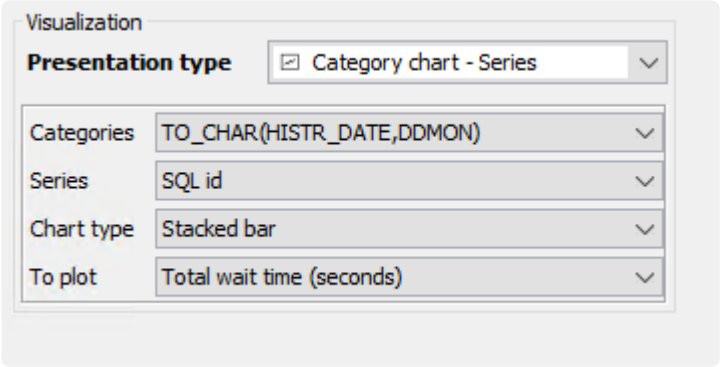
Status on check information

The following list shows configuration and status of the Alert log file.

- Log file name : alert_alaris1.log
- Log file path : DBW_DUMP_DIR
- **********
- Error status : OK
- Did not found error text : ORA-
- EOF status : OK
- Exeption status : OK
- **********
- **********
- Elapsed time : 0 sec.
- Date : 20.10.2022 12:10:42
- Tot. lines read from log : 5411
- Number of lines analyzed : 0

Result

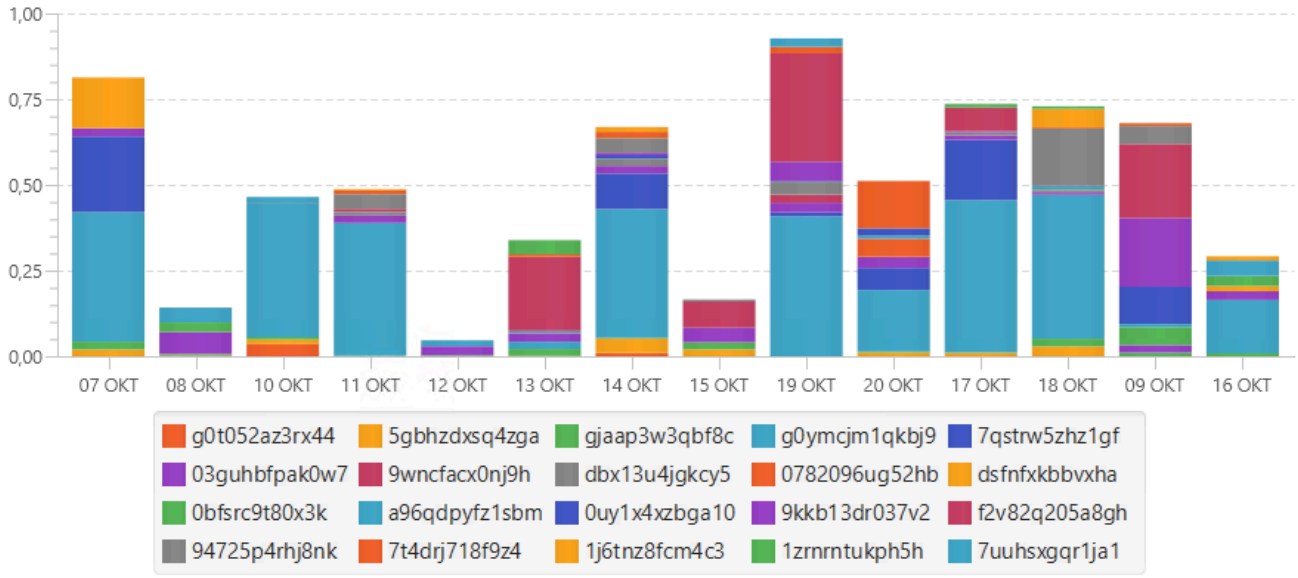
Category chart – series



Definition

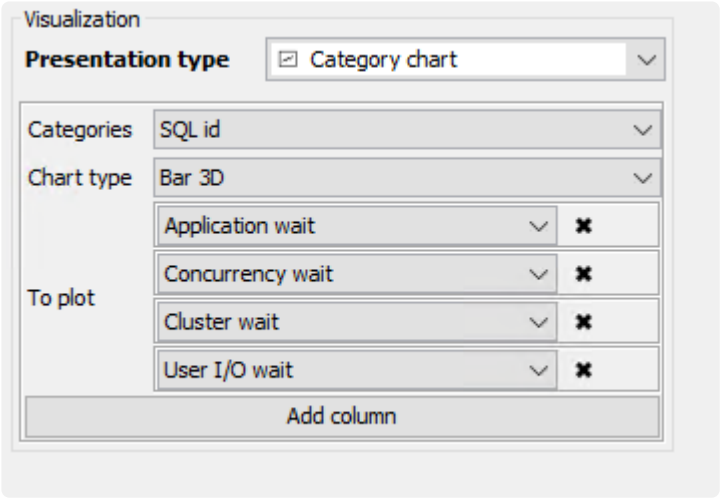
Daily SQL waits last 14 days (top 20)

antar



Result

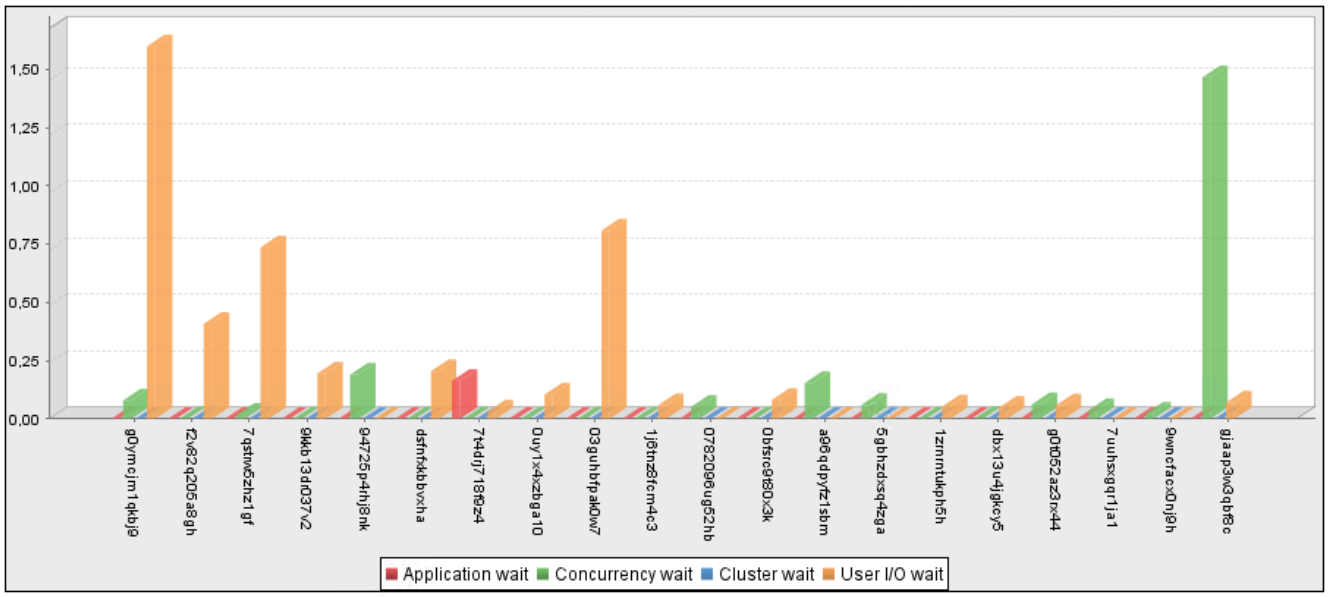
Category chart



Definition

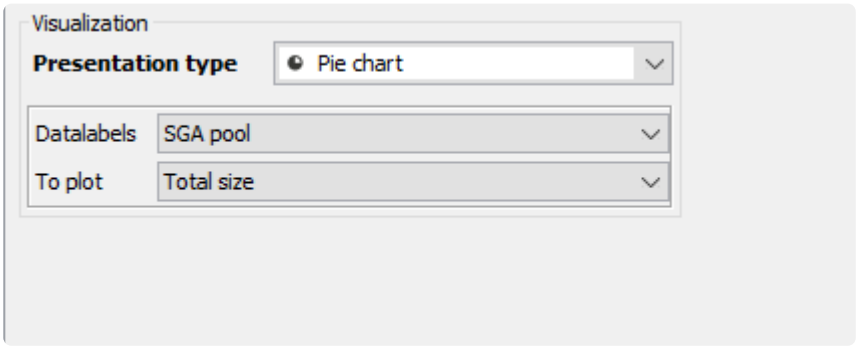
Top 20 statements and wait types

antar



Result

Pie chart

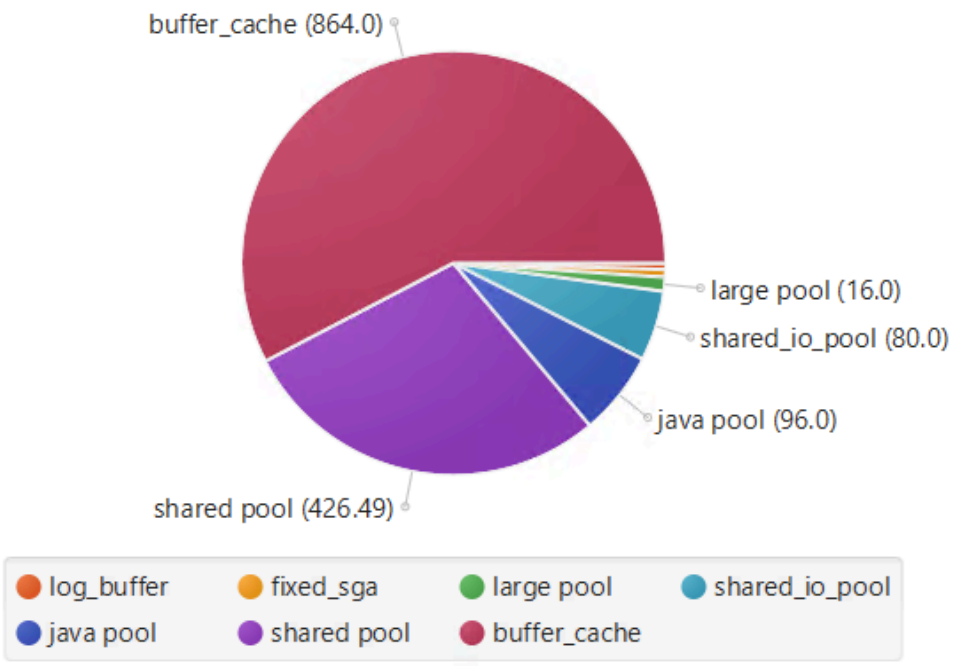


Definition

SGA pool size

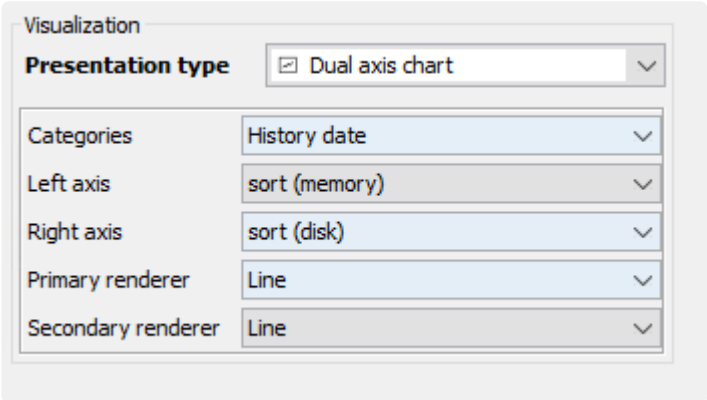
The following chart shows SGA pool components sizes in MB.

antar



Result

Dual axis chart

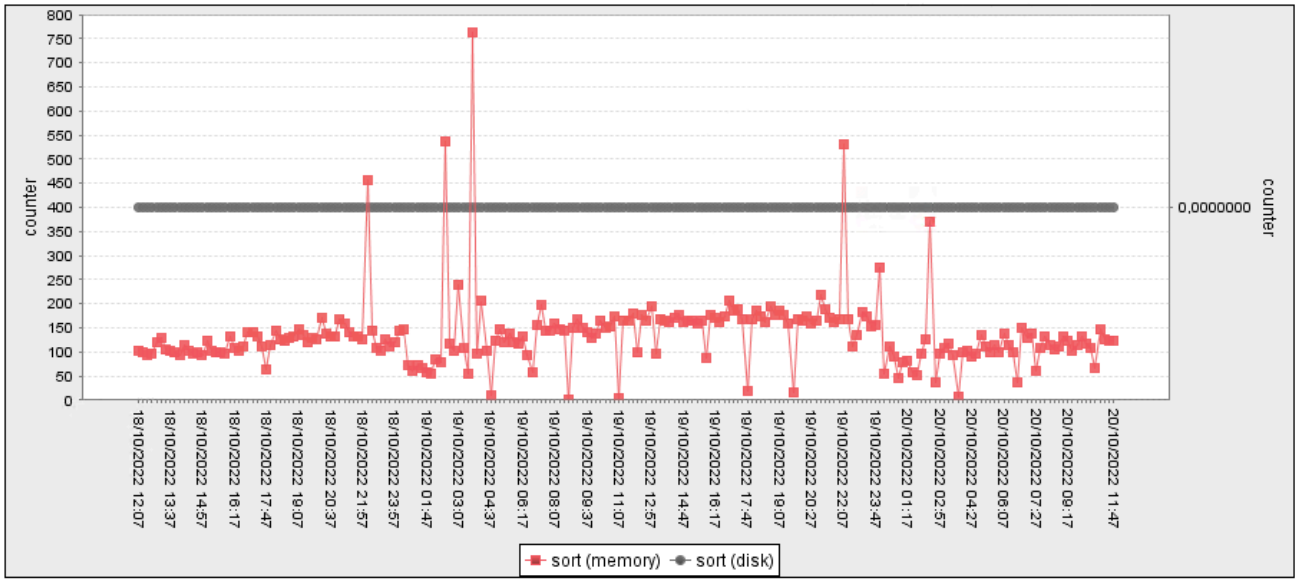


Definition

Sorts statistics for memory and disk.

The following chart shows the relationship between sorts in memory vs. sorts to disk for the last 48 hours. .

antar



Result

<< Upgrade / Example engine jobs >>

Example engine jobs

Example jobs

To simplify development efforts for customers we provide example jobs with comments that should help development efforts.

To use, right-click on the link and save as. Make sure its an xml file, with the name you want. You can edit it as its xml to change names etc. before you use “upload resource” in dbWatch Control Center monitor to upload the file.
Should then be visible in “Configure jobs”. Once installed, it can be edited by right-clicking on the job, and choosing Edit.

Github links to example jobs:

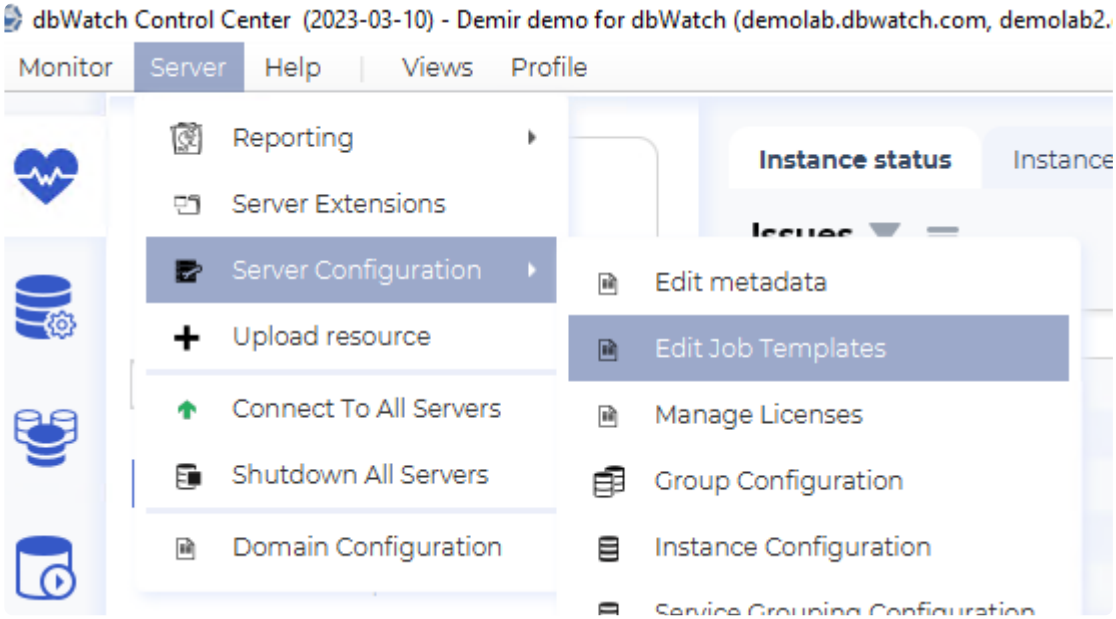
Example jobs

Job templates

Creating templates

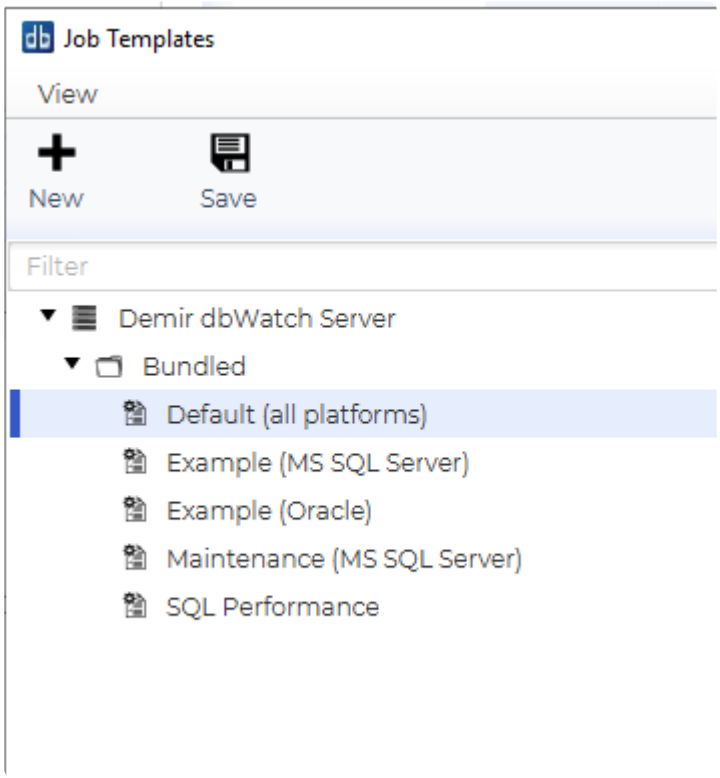
Template functionality makes it possible to automate installation and configuration of jobs across the entire database farm you are managing.

The configuration interface is available in the menu Server -> Server Configuration -> Edit job templates



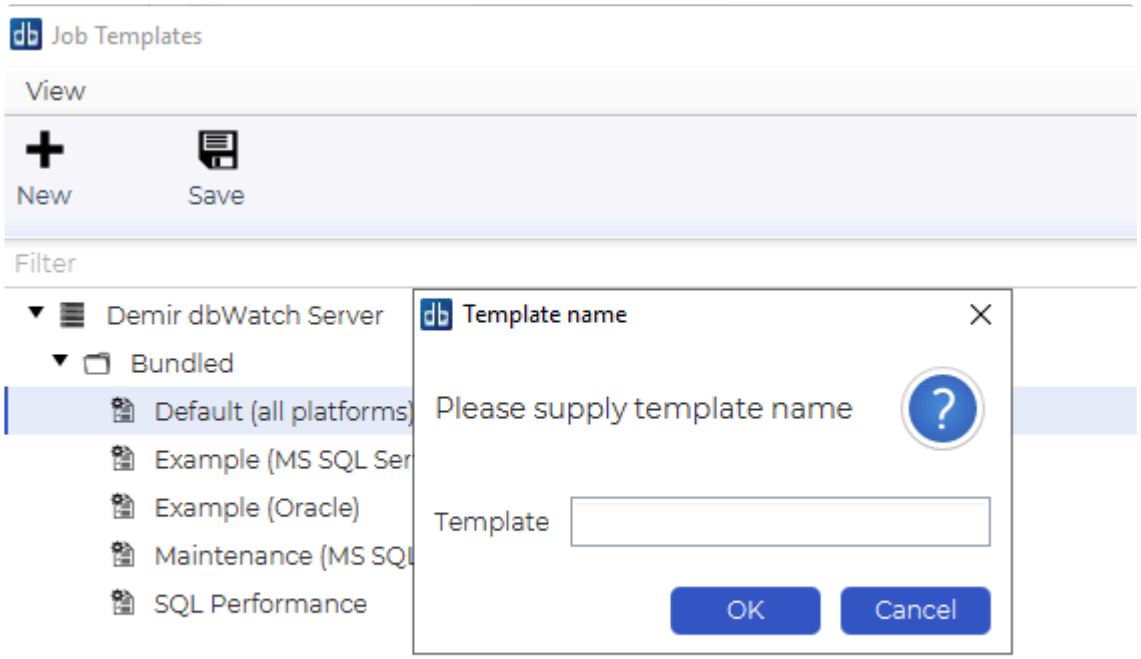
There are 5 templates available in dbWatch Control Center:

- Default (all platforms)
- Test (MS SQL Server)
- Test (Oracle)
- Maintenance (MS SQL Server)
- SQL Performance

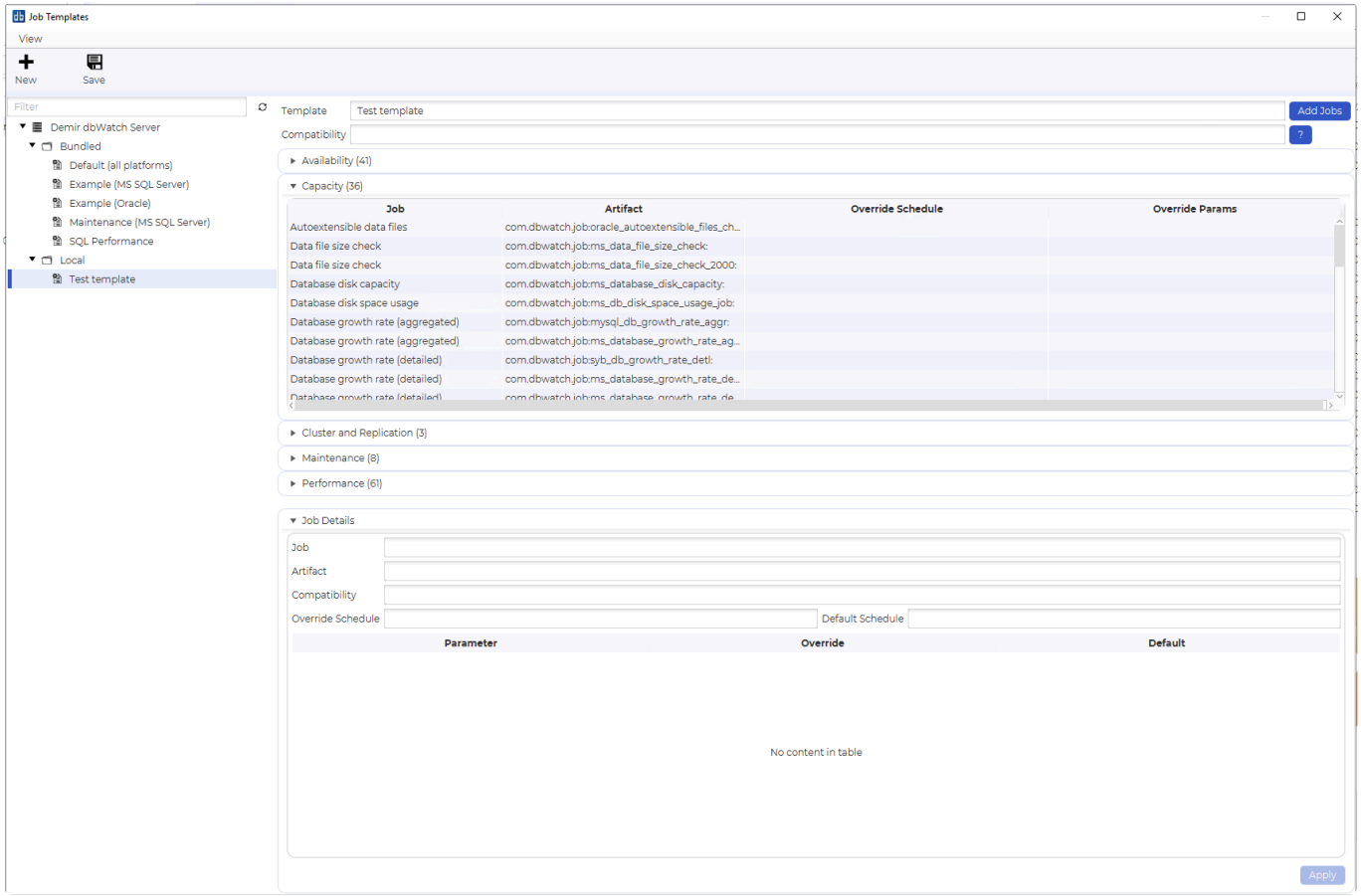


You can not change or delete a bundled template. To create your own templates, you can either right-click on a bundled or local template and choose to “Create Copy”, then enter a new template name.

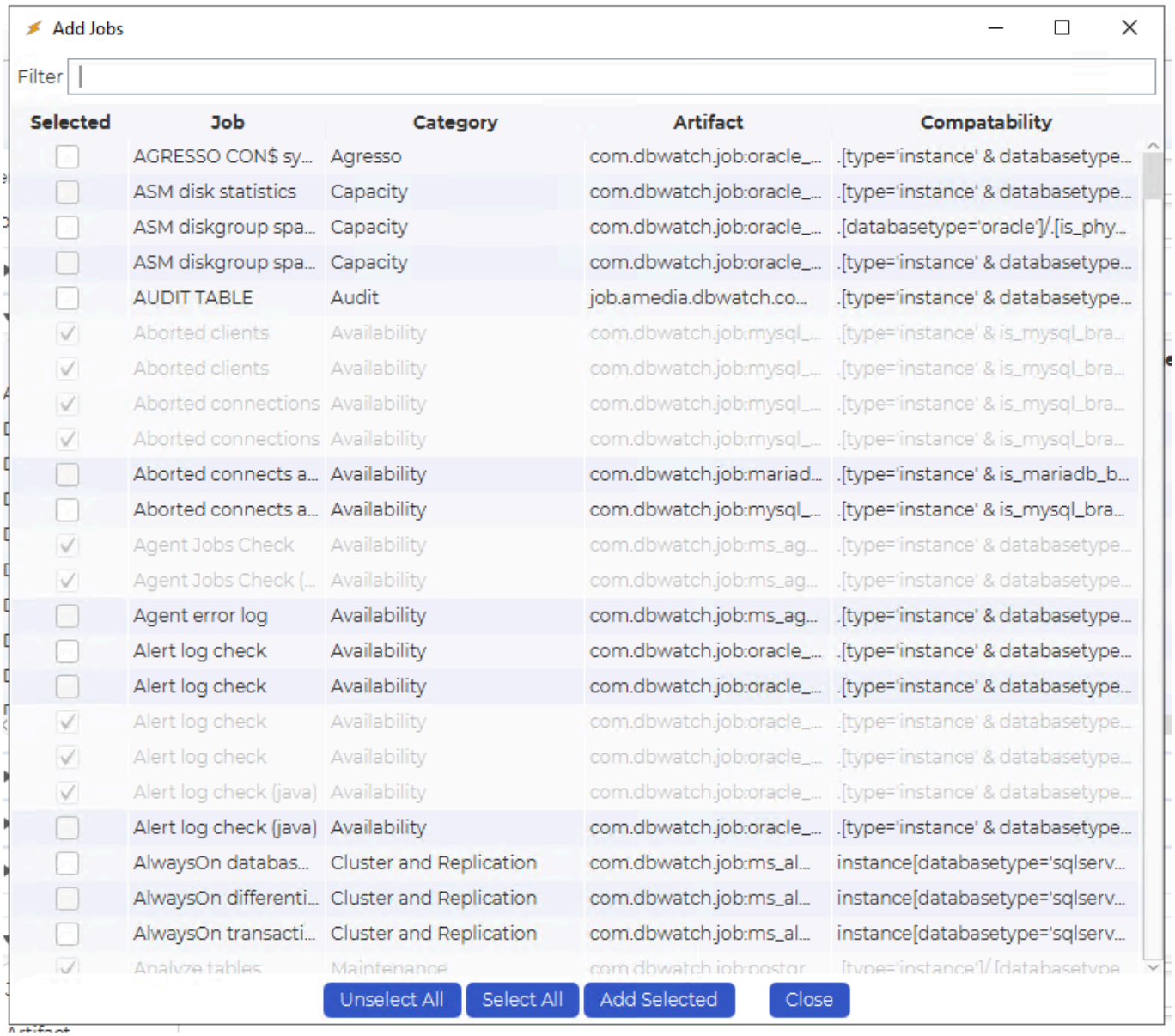
Or you can click on “New” to create a new empty template with no contents.



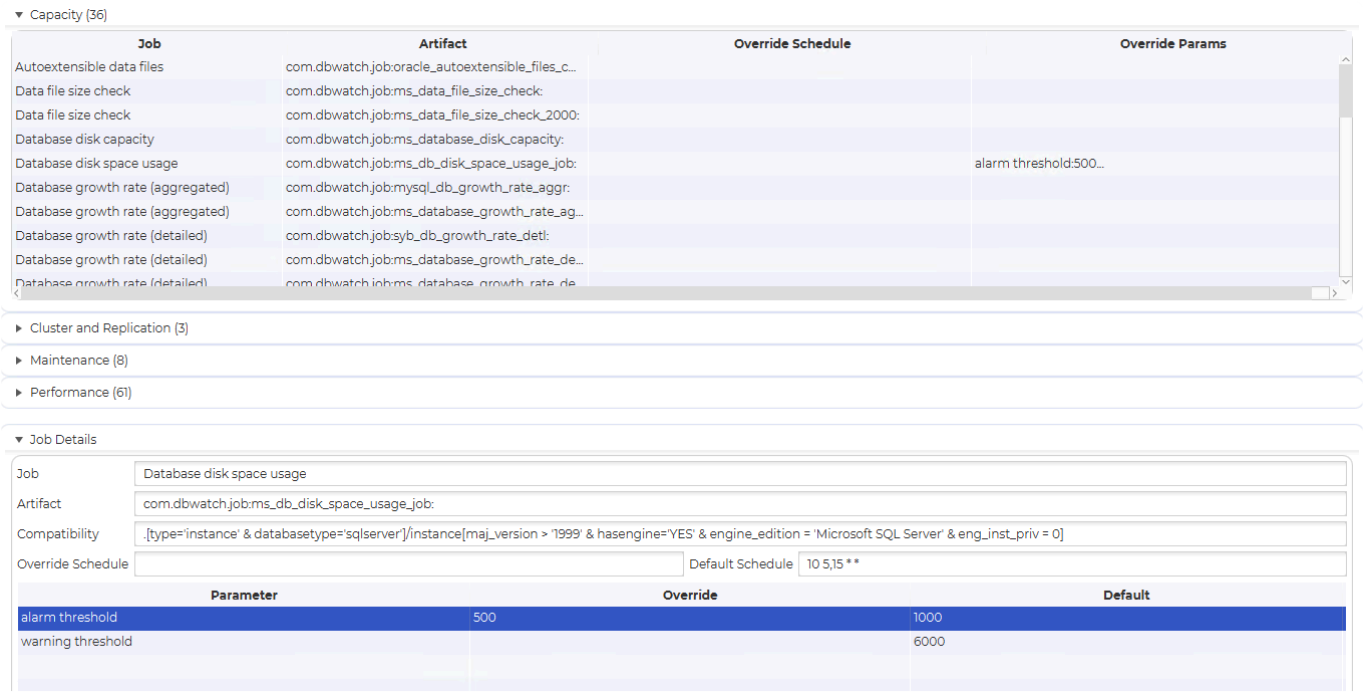
This interface will list the jobs that is part of the template, and if they have overrides in schedule or configurations. They are grouped by job groups such as “Availability”, “Capacity” and “Performance”.



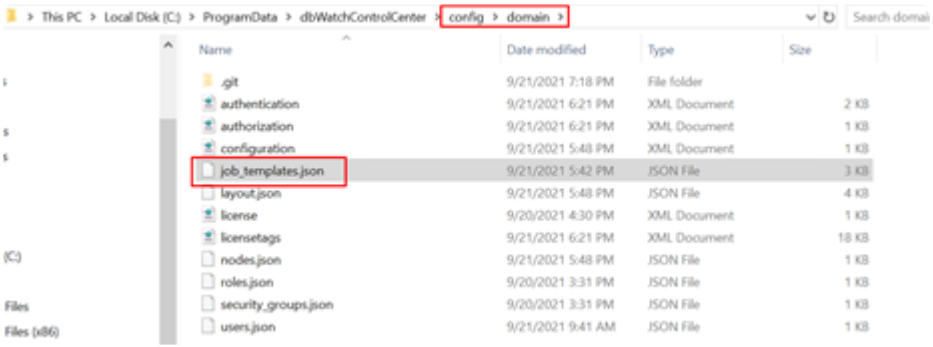
In the edit job template GUI, you can click on the blue “Add jobs” button to add new jobs to the template.



And you can choose to change schedules or parameters on the jobs.



The results of any changes are stored as a json file (“job_templates.json “) in the “config/server” directory (“C:\ProgramData\dbWatchControlCenter\config\server”).

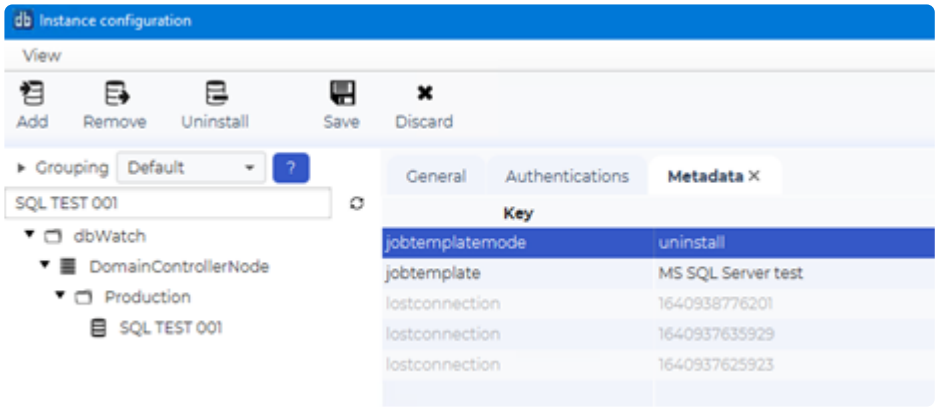


Deploying templates

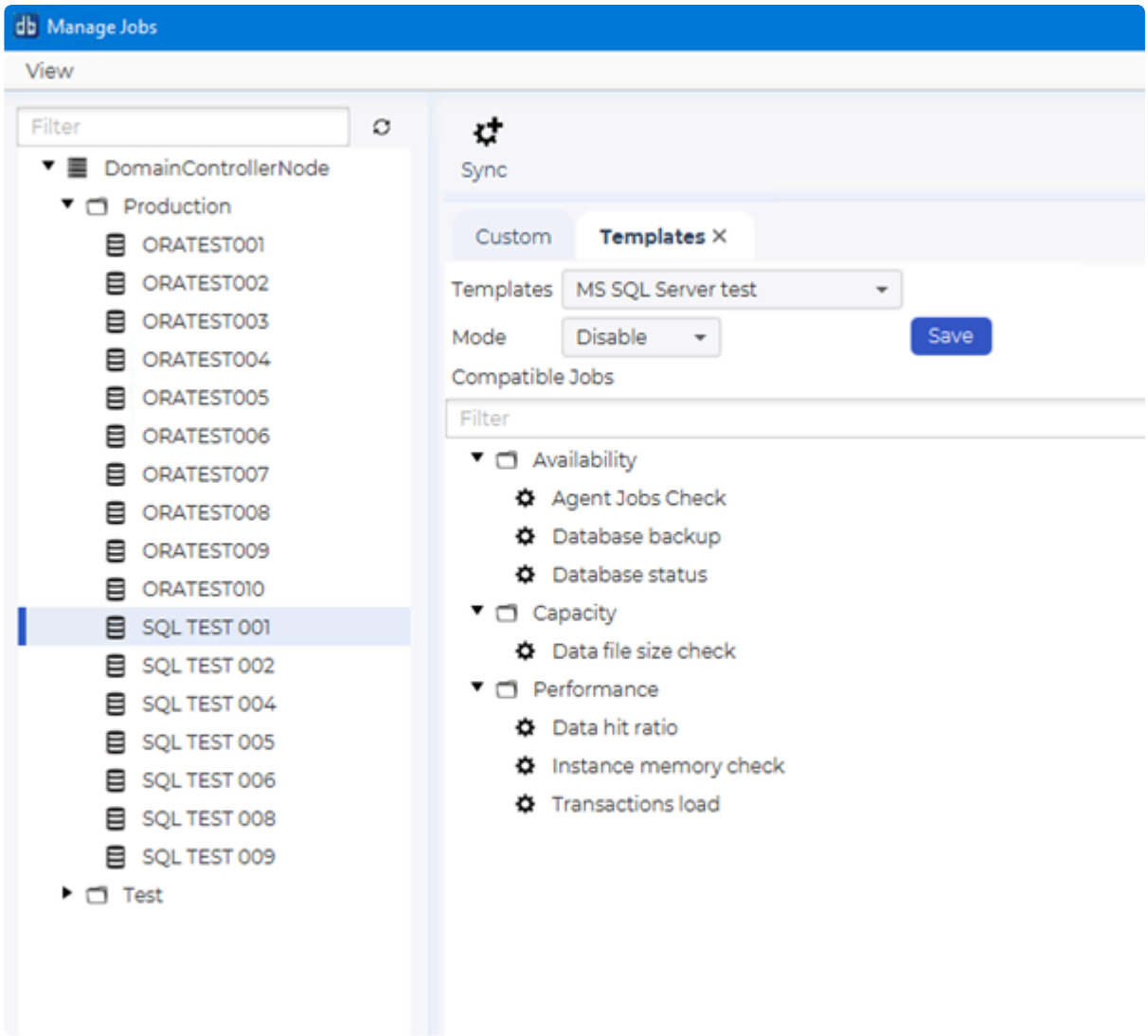
Templates are deployed in the “Manage jobs” view (Server -> Server Configuration -> Manage jobs)

When you install dbWatch jobs using template there are 3 different options regarding what to do with jobs that are already installed on the instance.
- Keep others
- Disable others
- Uninstall others

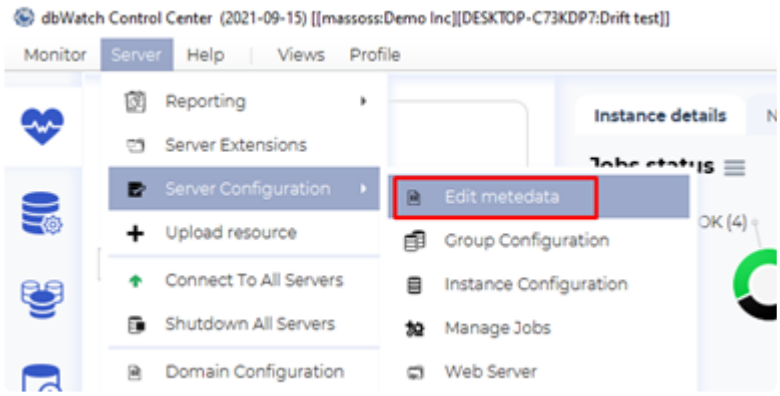
The default value is “Keep others” which means that jobs which are already installed will be unaffected. If you want to uninstall all jobs which are not specified in a template you have to add a metadata “jobtemplatemode” with value “Uninstall others” and “Disable others” will only disable the jobs that are not part of the template.



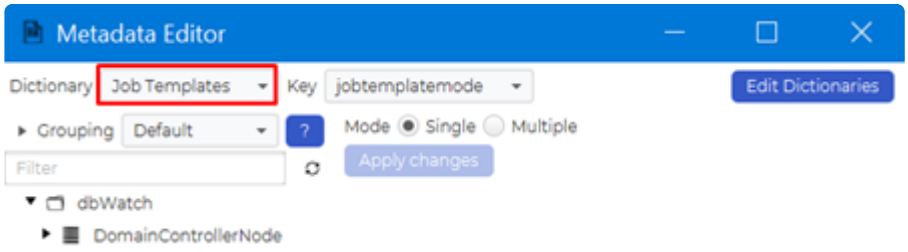
Both meta data (“jobtemplate” and “jobtemplatemode”) can be set using “Manage Jobs” GUI. For each instance go to the “Template” tab and specify template name and installation mode, and then click save. It is also possible to deploy several templates to one instance. In that case jobs from all templates will be merged.



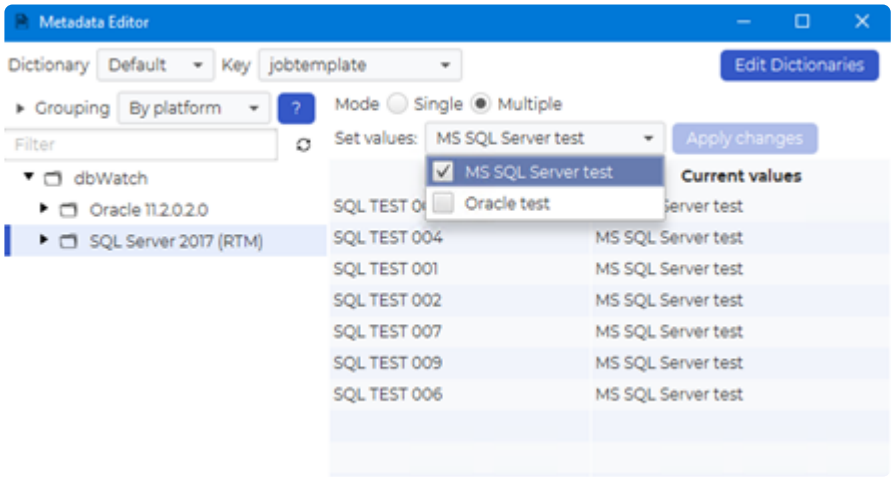
If you are going to add templates to several hundred of instances, we recommend to use “Edit metadata” GUI.
Go to the main menu, select “Server”, “Server configuration”, and then “Edit metadata”



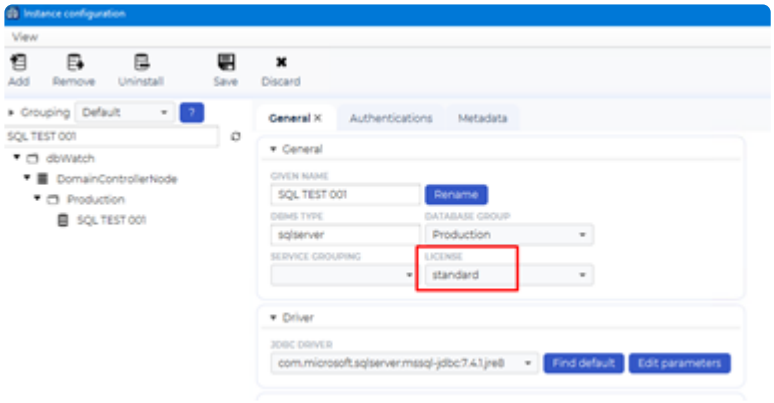
Then select Dictionary «Job Templates»



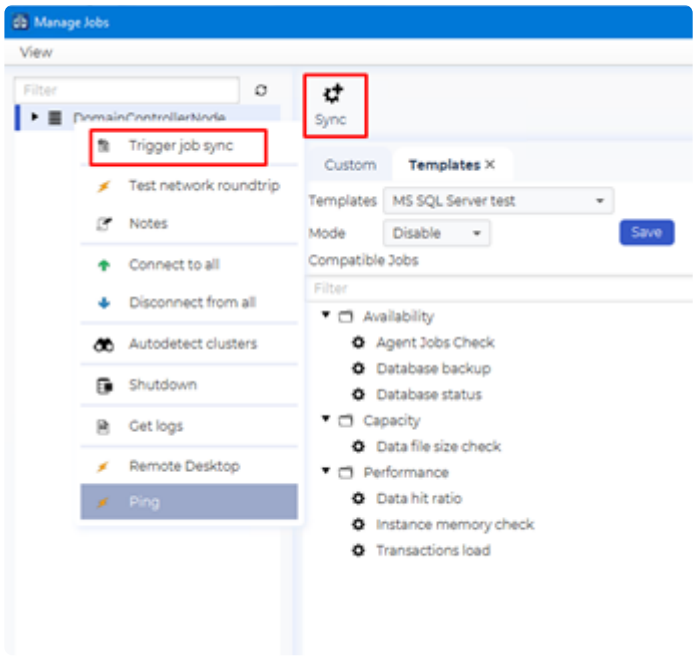
In this dictionary, two metadata are defined, “jobtemplate” with values that refer to all template names found in the job_templates.json file, and “jobtemplatemode” with three values, keep, uninstall and disable.
In Metadata Editor you can easily add metadata to many instances in the same view.
Select the correct “Key”, then “Mode”. If you select “Single” mode, you must select a value for each instance.
If you select “Multiple” mode, you can add a specific value to many instances at once.



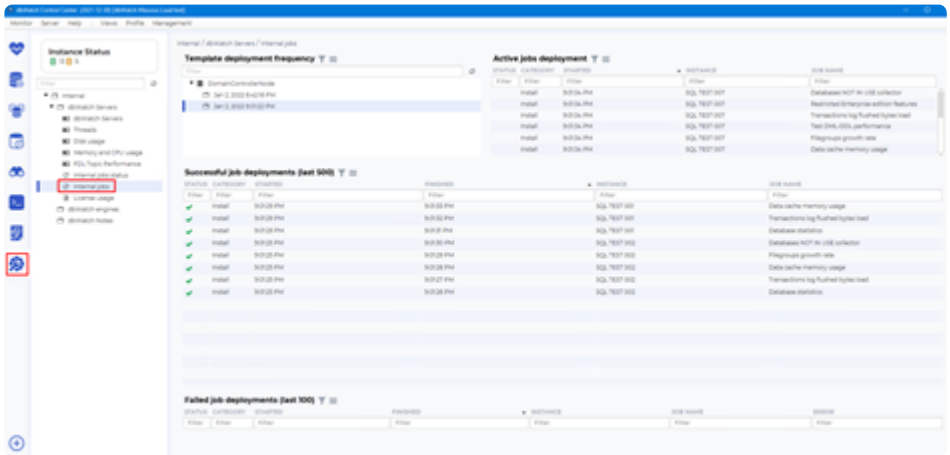
It is important to remember that the instances, where you want to install jobs from a template, must have a “license”. If the “License” value is empty, it will not be possible to install any jobs at all.
The license (in this case “standard”) is also linked to a list of jobs that you are allowed to install on an instance. If, for example, template jobs (from the template “MS SQL Server test”) are not in the “standard” list, then no jobs from this template will be installed.



To trigger deployment of a template on an instance (or many instances) click on “Sync” in “Manage Jobs” GUI, or right click on “DomainControllerNode” and select “Trigger job sync”.



You can monitor deployment of templates in “Internal jobs” view.



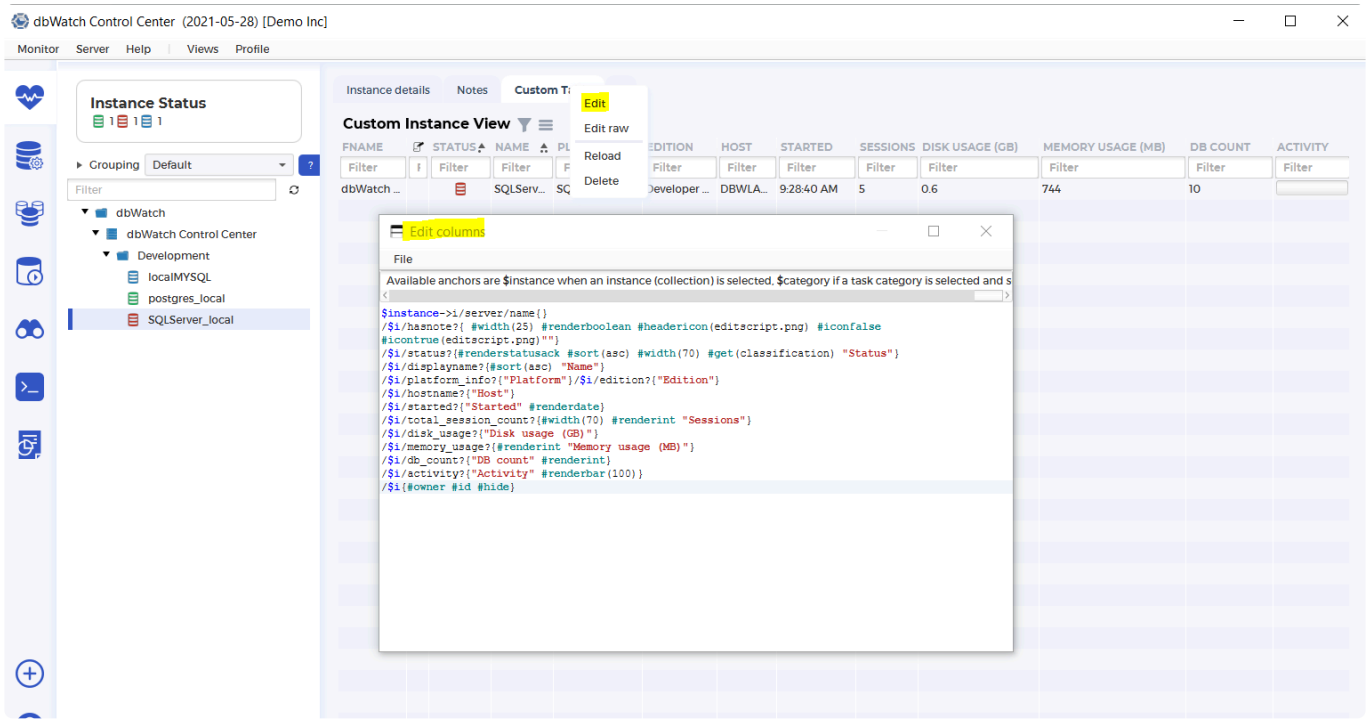
Customize View using FDL

Before proceeding to this section, you must be familiar with Farm Data Language. To give you an overview, Farm Data Language is a type of query dbWatch uses to read and display data to its user. Think of it as your query tool when using dbWatch.

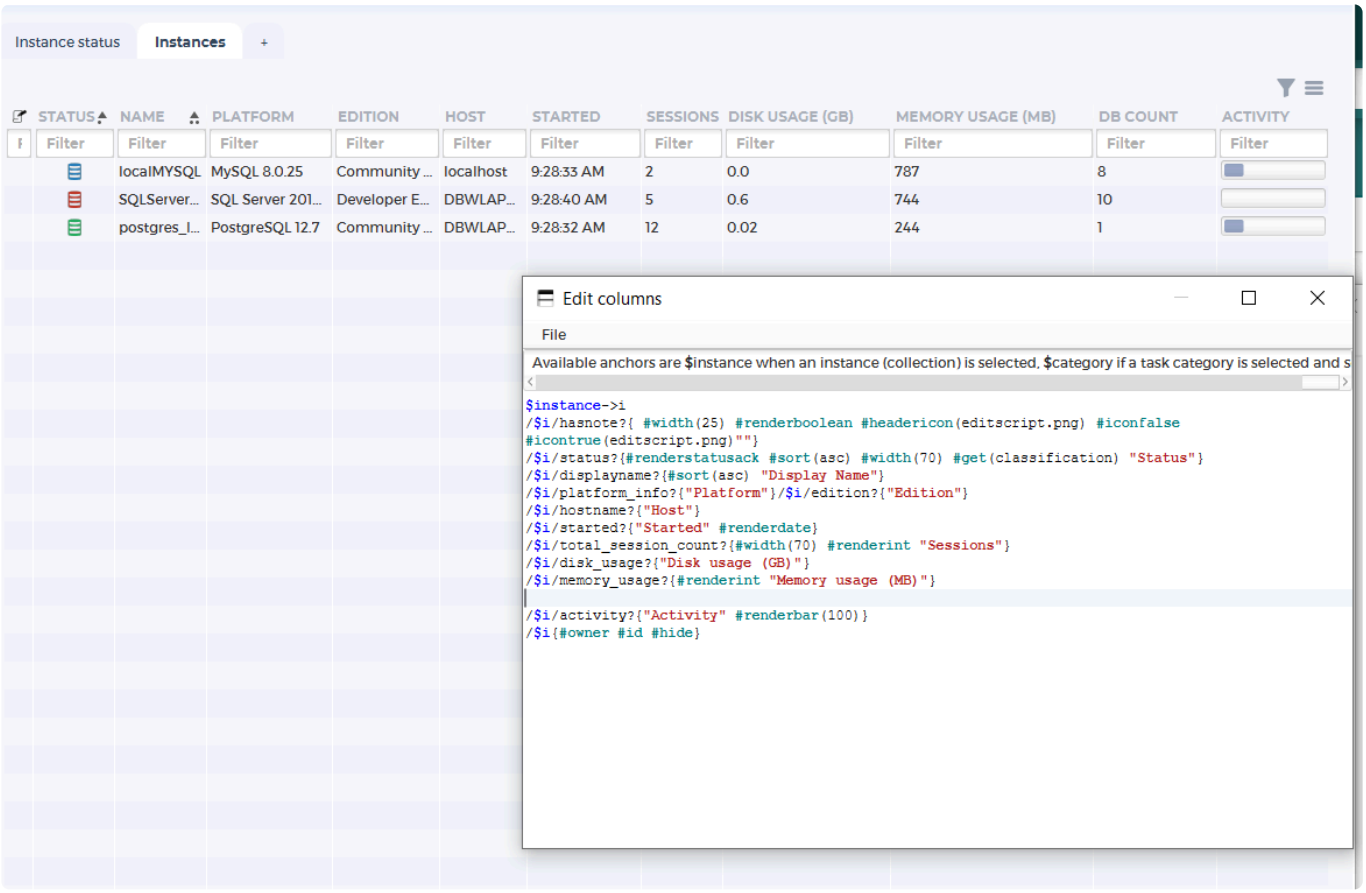
How to edit Views

Edit View Directly

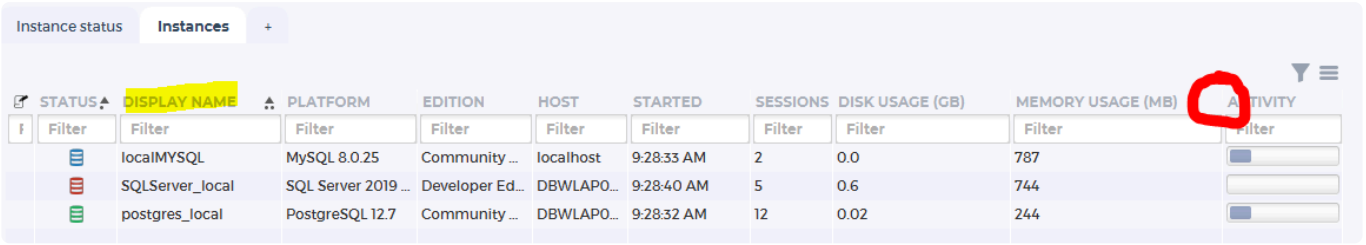
To edit your views, you need to right-click on one of the tabs and select “Edit”. You should be able to see a window labeled as “Edit Columns”. In this window, you can change the FDL script to fit your own business needs. But, be careful, when you edit a column, it may disrupt some feature of the Module.



Let’s use the instance tab in the monitoring module for example. I want to change the column header of **Name** to **Display Name**, and remove **dbcount** column entirely. First, I open go to Edit and manipulate the FDL query.

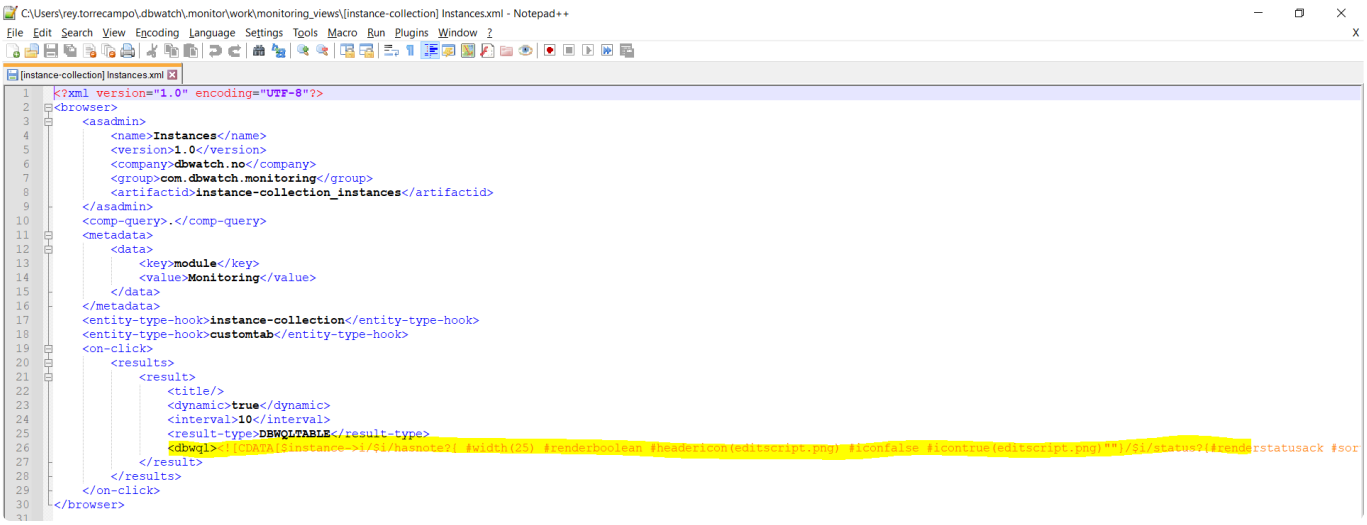


Once, I'm happy with the changes, go to File and click Save. Afterwards, restart dbWatch Control Center. Your changes should be applied.

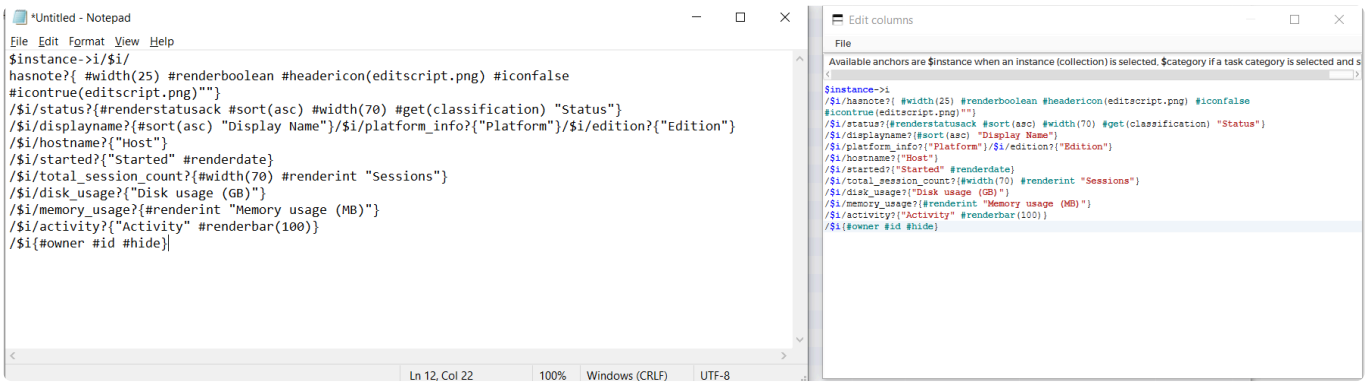


Editing View in Raw Data

Another approach is to open the XML file directly. Same as before, but you need to select "Edit Raw". Your machine will then read the XML file. A rule of thumb is to use amiable software for reading XML files and set them as your default reader.

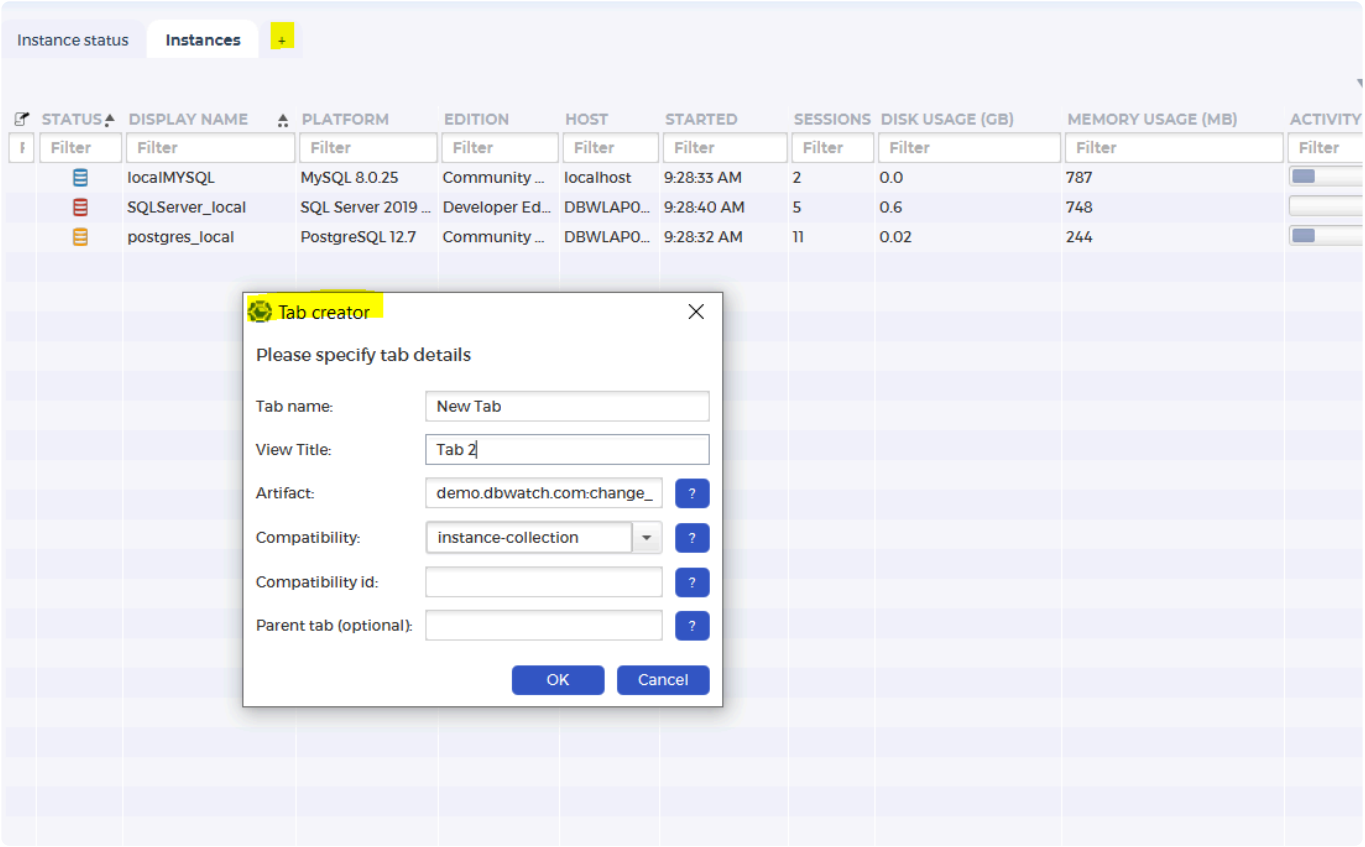


Once opened, you will see the following. What you want to focus on is the line enveloped in . Take note that the structure inside follows this format: < ! [CDATA[***FDL query***]] > and the query inside it essentially the same found when editing it directly.

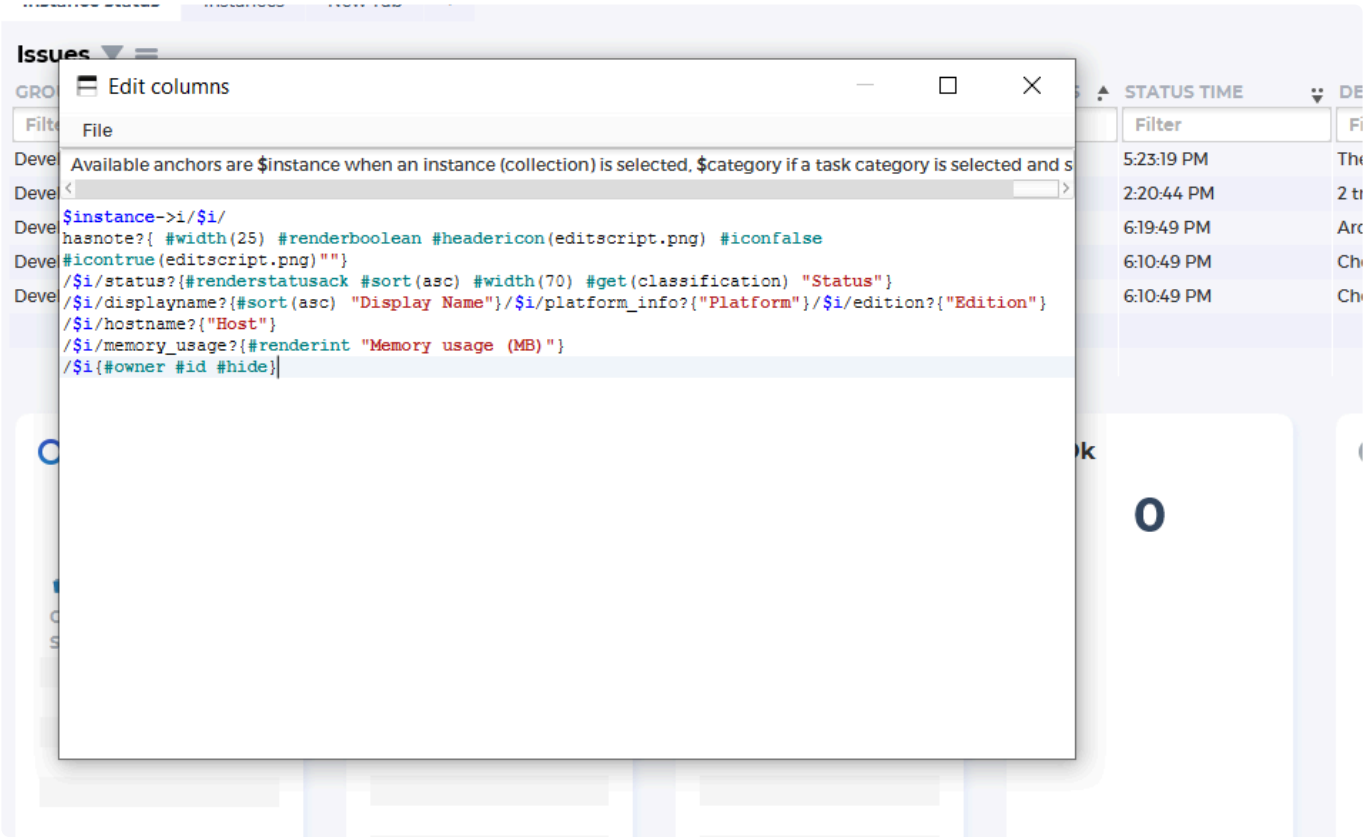


Creating a new tab

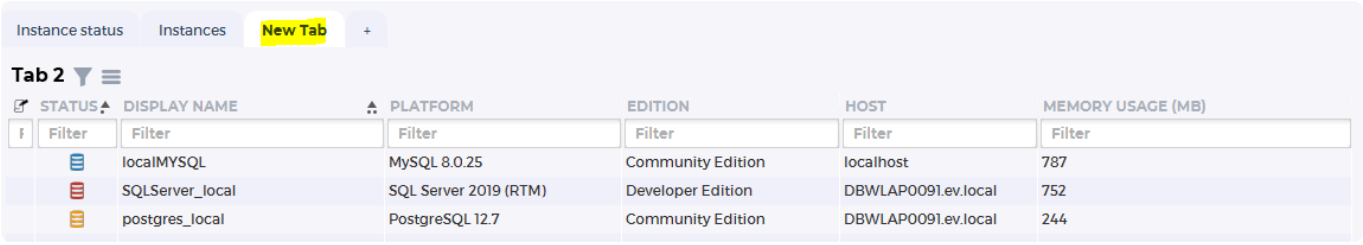
Instead of Editing a view, you can create your own view. First, click on the “+” to add a tab. A tab creator window will appear. Next input the tab name and tab title. Also, change the artifact version to another number other than 1.0.



Afterward, the Edit Columns window will appear. Input your FDL Code. Then, Click File > Save.



Close the application and relaunch it. The new tab should appear.



<< Job templates / No-engine jobs >>

No-engine jobs

No-engine jobs

No-engine jobs are monitoring jobs that does not run in the database engine. No-engine jobs are jobs that run on the dbWatch Server, and typically run a SQL statement and process the result of this statement in javascript.

Example code for different platforms are available here: Example no-engine jobs

The contents are in xml format, and can be uploaded to the dbWatch Server as a file with .xml extension.

We go through the code section by section.
The entire code for the monitoring job is within the properties tags

```
<properties>
```

Under this level there are tree tags, comp-query, asproperties and property.

The tag “comp-query” is an optional tag to make sure this file is not read for database instances where its not compatible with. Its is there purely for performance reasons as it helps dbWatch Control Center to not read files that are not relevant for database instances that they are not needed for. As a developer that makes code for internal use this is not required.
The tag:

```
 <comp-query><![CDATA[.[type='instance' & databasetype='oracle']]]></comp-quer
y>
```

This will let dbWatch Control Center know that this file is only relevant for objects that are of the type “instance”, and also of the databasetype “oracle”.

The tag “asproperties” sets metadata for this no-engine job. These metadata tags are used for resolving versioning and how this job will appear. Name is the name of the job, but not the name it will show up as in the “Configure jobs” view, this is “display-name” in the main property definition. Version will be used to determine which job will be used, so if you update with a new job, this needs to have a higher version number. Company should match the domain name for your license. Group should match your domain name in reverse notation + .job, as this is used in grouping files internally. If your domain is “example.com”, company should be example.com and group should be “com.example.job”. The tag “artifactid” must be unique to this job, and typically follow the naming convention “databaseplatform_noshema_name_of_job”.

Example:

```
<asproperties>
                <name>Oracle Simple Query</name>
```

```
                <version>1.0</version>
                <company>company.com</company>
                <group>com.company.job</group>
                <artifactid>oracle_noschema_simple_query</artifactid>
</asproperties>
```

The tag “property” consists of the bulk of the no-schema job, and the property tag itself sets an internal name for the job. This should not conflict with any other FDL property used, so its naming convention is “_job_noschema_name_of_job”, but must be unique to this job. At the end there is a closing xml tag.

```
<property name="_job_noschema_simple_query">
```

There are many tags defined under the property tag. The first tags defines mandatory keys for status (alarm, warning, ok) and details (short information about status) that is used in the monitoring gui. They can be reused as-is.
bc.
details

The “display-name” tag is the name this monitoring job will have in the “Configure jobs” and Monitoring gui. It is what the end user will see as the name of the monitoring job.

```
<display-name>Simple Query</display-name>
```

This will name this job “Simple Query”.

The “description” tag is the description about this job that will be visible in the “Configure jobs” interface when you choose that monitoring jobs to install. It is a short description on what this monitoring job does, that can say a bit more than its name.

```
<description>Doing a simple query</description>
```

The “category” tag will group this job in a category. Typical categories are Availability, Capacity, Maintenance and Performance, but this can be something new, like an application name or the name of a company, to make it easier to spot. The “Maintenance” category is given special treatment, however, as the jobs defined in this category will **have no timeout**, so its intended for jobs that will not run quickly, and have their own internal control of when they should stop to run. Most maintenance jobs have a configured time-window where they will stop themselves if they run for too long.

```
 <category>Availability</category>
```

The “parameters” tags are the configurable parameters that you get when you right-click and choose “Configure” on a job. Their “name” will be the name of the automatically declared variable in the javascript routine (so no spaces). The “default” is the default configured setting. The “type” is the javascript datatype, such as int, bigint, string etc, and “description” is the description of what this parameter adjusts that will be shown to the end user when they choose to “Configure” a job.

```
<parameters>
                        <parameter name="warning_threshold" default="20" typ
e="int" description="Number threshold for warning"/>
                        <parameter name="alarm_threshold" default="5" type="in
t" description="Number threshold for alarm"/>
</parameters>
```

The “installable” tags are used to determine if this job can be installed on a specific instance. The key must be set to the “property name”. The “compatibility” is a FDL tag to what kind of databasetype or other FDL tags this job is compatible with. The “instance” tag here is only true on Oracle RAC type systems where we should have one installed per instance in a cluster.

```
<installable>
                        <key>_job_noschema_simple_query</key>
                        <compatability><![CDATA[.[databasetype='oracle']
]]></compatability>
                        <instance>false</instance>
</installable>
```

The next “compatibility” tag is for when this should run, which is only when its installed, so only when this job is installed on an instance.

```
<compatability>.[source like "_job_noschema_simple_query"]/instance</compatabi
lity>
```

The “default-schedule” tag is what its default schedule should be. Here we can have both interval format in s (seconds), m (minutes) or h (houres), as well as the crontab-like format is allowed. To short intervals, so less than 30s should be avoided as it could lead to performance issues.

```
<default-schedule>1h</default-schedule>
```

The “entityType” tag will tell the dbWatch Control Center server what type of property this is and can be left as-is.

```
<entityType>scheduledtask</entityType>
```

The “value” tags are a water-fall type set of tags, where tags furter down in the file can inherit data from the previous tag or tags. They are executed as sorted in the file. First value tag is executed first.
The value tag has a engine variable, which tells us what type of engine will evaluate the value contents. Engines are sql (run a SQL statement), fdl (run a FDL query), javascript (run a javascript routine), dbwql (same as fdl) and regex (run a regular expression).
Example SQL:

```
<value engine="sql"> select 10 from dual
```

```
                        <instance-resolver>at/instance</instance-resolver>
</value>
```

Example javascript:

```
<value engine="javascript" foreach="row">
                        <![CDATA[
                    try {
                            var value=input.get(0).asLong();
                                        var status=0;
                                        var msg;
                                        if (value > warning_threshold) {
                                                status = 1;
                                                msg="Value more than warning t
hreshold" + value +" ";
                                        }
                                        if (value > alarm_threshold) {
                                                status = 2;
                                                msg="Value more than alarm thr
eshold" + value +" ";
                                        }
                                        else {
                                                msg="Value " + value +" ";
                                        }
                                        result.setStatus(status, msg);
                }
                catch (err) {
                    result.setStatus(2, err.messag
e);
                }
                ]]>
                </value>
```

This javascript routine, gets the output from the SQL as “input” and parameters are automatically declared variables. The routine result.setStatus(int,string) will set status value (0=ok,1=warning and 2=alarm) and a string details message.

The “dbwatch-report-template” is the report template for what gets generated when you click on “Details” for a job. Its not mandatory, but follows the same format as engine-jobs and general reports.

```
<dbwatch-report-template>
                        <version>2</version>
                        <title>Simple Query</title>
                        <description>Simple Query</description>
                        <default-schedule>0 * * *</default-schedule>
                        <text>Simple Query</text>
```

```
                        <chapter>
                                <type>
                                </type>
                                <title>Details</title>
                                <text>
                                </text>
                                <presentations>
                                        <presentation>
                                                <resultset>
                                                        <column>
                                                                <name>Detail
s</name>
                                                                <sql-type>0</s
ql-type>
                                                        </column>
                                                </resultset>
                                                <table>
                                                        <to-plot>Details</to-p
lot>
                                                </table>
                                                <select-call><![CDATA[ select
sysdate from dual ]]></select-call>
                                                <text>
                                                        This table shows just
the sysdate
                                                </text>
                                                <title>
                                                        Details
                                                </title>
                                        </presentation>
                                </presentations>
                        </chapter>
</dbwatch-report-template>
```

Full example code for Oracle:

```
<?xml version='1.0' encoding='utf-8'?>
<properties>
        <comp-query><![CDATA[.[type='instance' & databasetype='oracle']]]></co
mp-query>
        <asproperties>
                <name>Oracle Simple Query</name>
                <version>1.0</version>
                <company>company.com</company>
                <group>com.company.job</group>
                <artifactid>oracle_noschema_simple_query</artifactid>
```

```
        </asproperties>
        <property name="_job_noschema_simple_query">
                <key name="status" entityType="status"/>
                <key>details</key>
                <display-name>Simple Query</display-name>
                <description>Doing a simple query</description>
                <category>Availability</category>
                <parameters>
                        <parameter name="warning_threshold" default="20" typ
e="int" description="Number threshold for warning"/>
                        <parameter name="alarm_threshold" default="5" type="in
t" description="Number threshold for alarm"/>
                </parameters>
                <installable>
                        <key>_job_noschema_simple_query</key>
                        <compatability><![CDATA[.[databasetype='oracle']
]]></compatability>
                        <instance>false</instance>
                </installable>
                <compatability>.[source like "_job_noschema_simple_query"]/ins
tance</compatability>
                <default-schedule>1h</default-schedule>
                <entityType>scheduledtask</entityType>
                <value engine="sql"> select 10 from dual
                        <instance-resolver>at/instance</instance-resolver>
                </value>
                <value engine="javascript" foreach="row">
                        <![CDATA[
                    try {
                            var value=input.get(0).asLong();
                                    var status=0;
                                    var msg;
                                    if (value > warning_threshold) {
                                            status = 1;
                                            msg="Value more than warning t
hreshold" + value +" ";
                                    }
                                    if (value > alarm_threshold) {
                                            status = 2;
                                            msg="Value more than alarm thr
eshold" + value +" ";
                                    }
                                    else {
                                            msg="Value " + value +" ";
                                    }
                                    result.setStatus(status, msg);
                }
```

```
                catch (err) {
                    result.setStatus(2, err.messag
e);
                }
                ]]>
                </value>
                <dbwatch-report-template>
                        <version>2</version>
                        <title>Simple Query</title>
                        <description>Simple Query</description>
                        <default-schedule>0 * * *</default-schedule>
                        <text>Simple Query</text>
                        <chapter>
                                <type>
                                </type>
                                <title>Details</title>
                                <text>
                                </text>
                                <presentations>
                                        <presentation>
                                                <resultset>
                                                        <column>
                                                                <name>Detail
s</name>
                                                                <sql-type>0</s
ql-type>
                                                        </column>
                                                </resultset>
                                                <table>
                                                        <to-plot>Details</to-p
lot>
                                                </table>
                                                <select-call><![CDATA[ select
sysdate from dual ]]></select-call>
                                                <text>
                                                        This table shows just
the sysdate
                                                </text>
                                                <title>
                                                        Details
                                                </title>
                                        </presentation>
                                </presentations>
                        </chapter>
                </dbwatch-report-template>
        </property>
</properties>
```

Property System

There are several types of properties in dbWatch.

Predefined properties

Predefined properties are always present. For example, instances have properties called name and host. For more details see available properties

Metadata properties

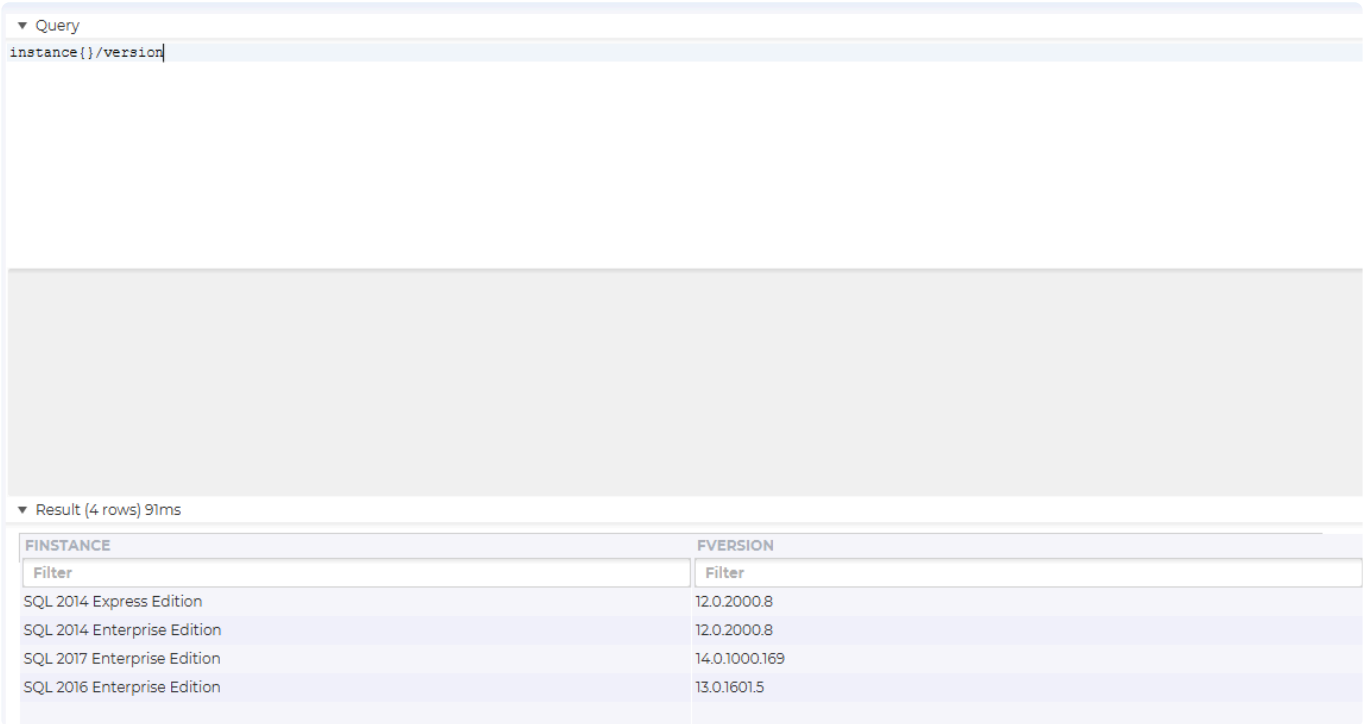
Metadata properties are manually assigned to particular objects in dbWatch (like instances or tasks). Read more.

Dynamic properties

Dynamic properties are properties that are dynamically defined for objects of a particular type. Read more.

Accessing properties

All properties can be queried using FDL. For example to access the version property for instances:



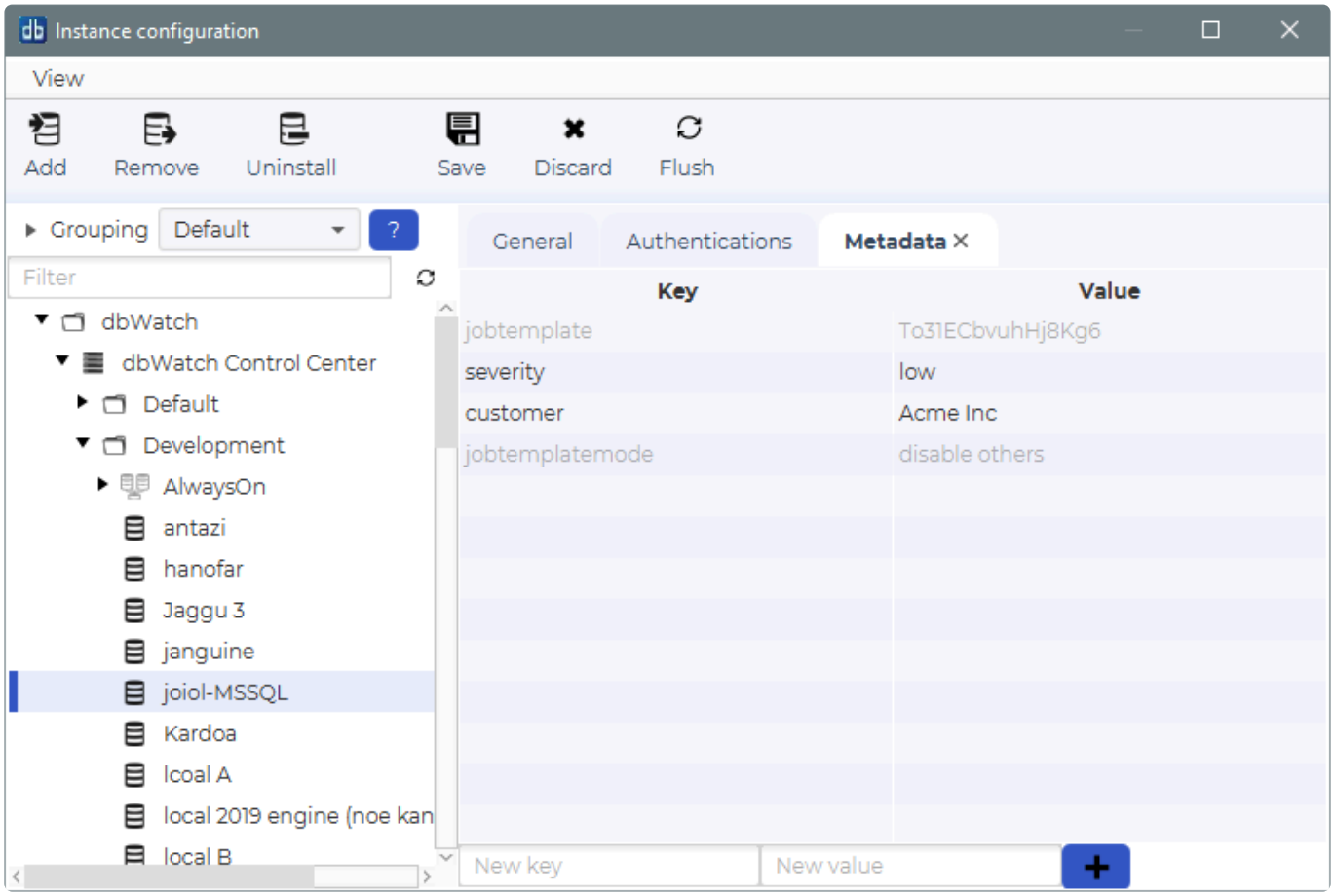
Read more about fdl.

<< No-engine jobs / Metadata properties >>

Metadata properties

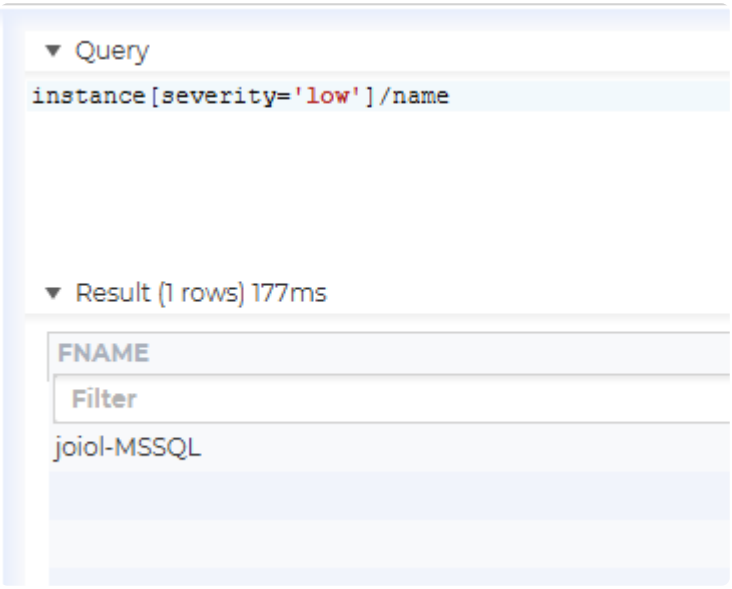
You can put metadata on instances, and this metadata will be automatically available in FDL.

To add metadata to an instance, right click on it and select “Configure”.
Then click on the “Metadata” tab.



Here you can add key/value pairs by typing in the input fields on the bottom, and clicking the “+” button.

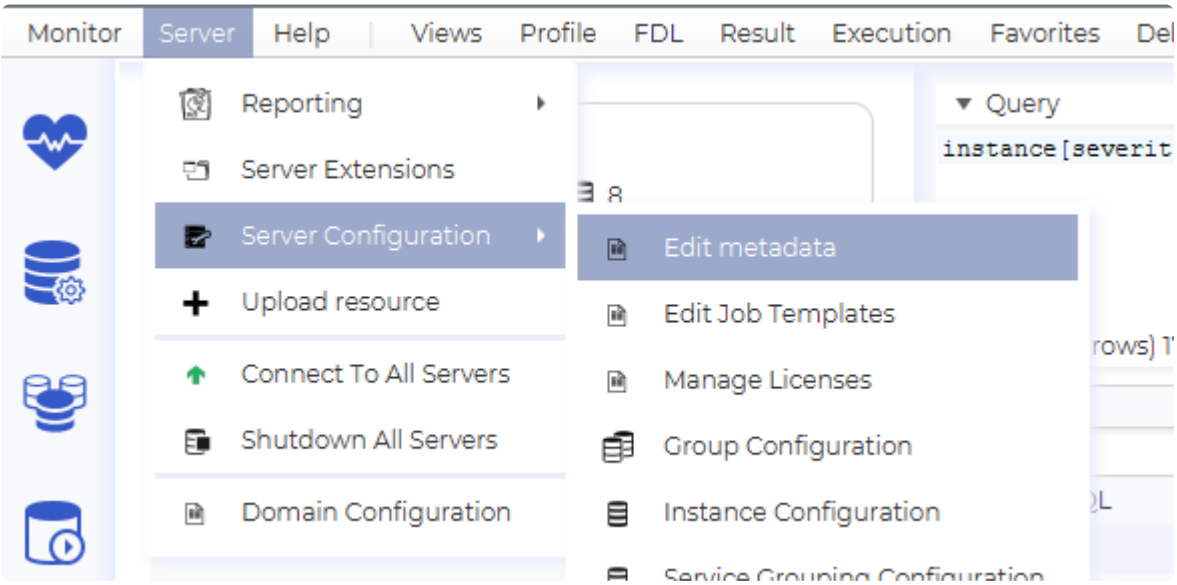
F.ex if you enter “severity” as key and “low” as the value, click “+” and then save. The information will be available in FDL.



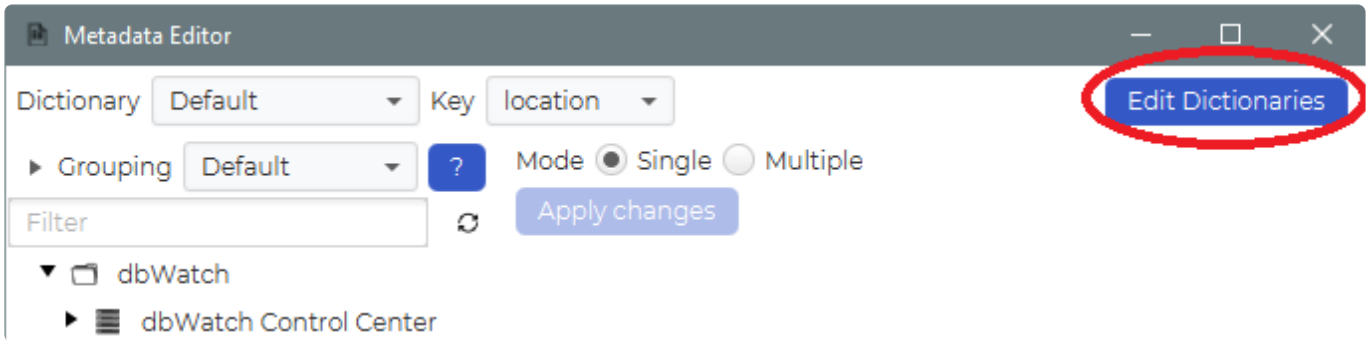
This way of setting metadata is fine for simple ad hoc metadata. But as you might imagine, if you have multiple users freehand entering metadata, you quickly end up with a lot of different naming paradigms and spelling mistakes. To avoid this, use the Metadata dictionary to define legal metadata keys and values.

Dictionary Editor

To open the Dictionary editor, find “Edit metadata” on the “Configure” -> “Server Configuration” menu.

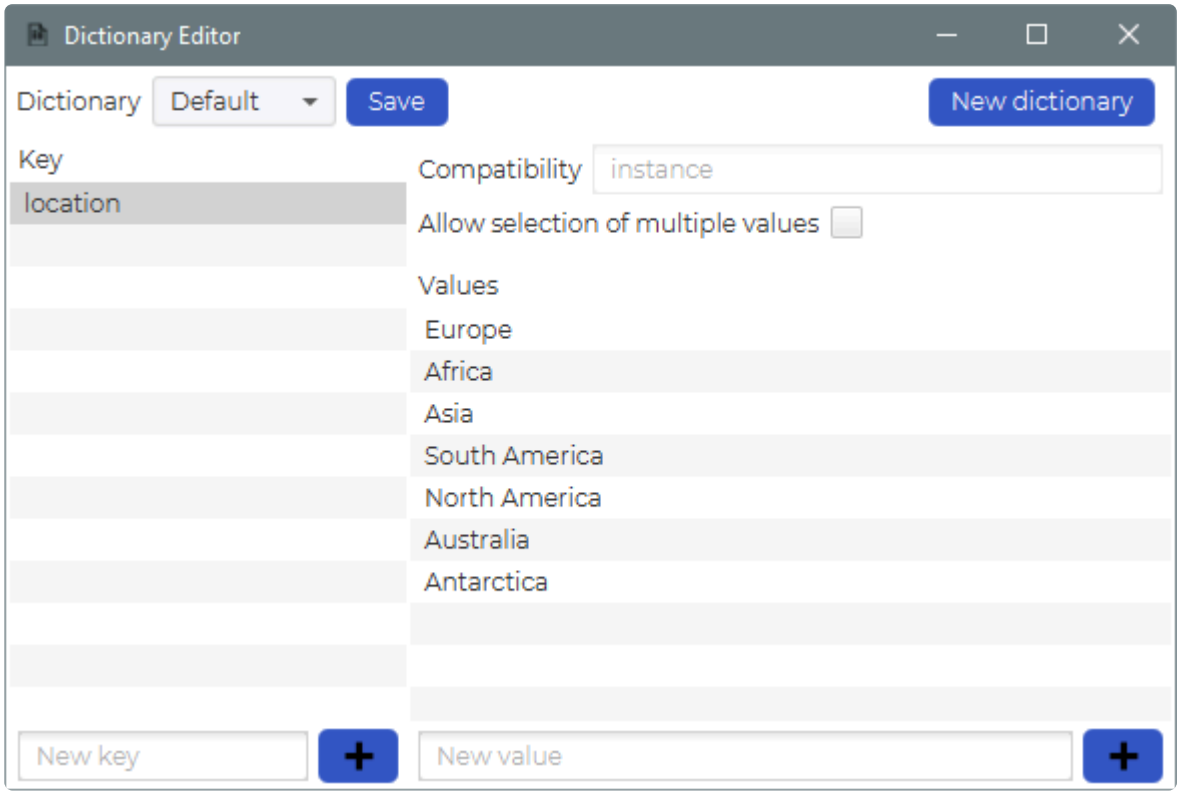


Then click on “Edit Dictionaries” in the Metadata Editor.



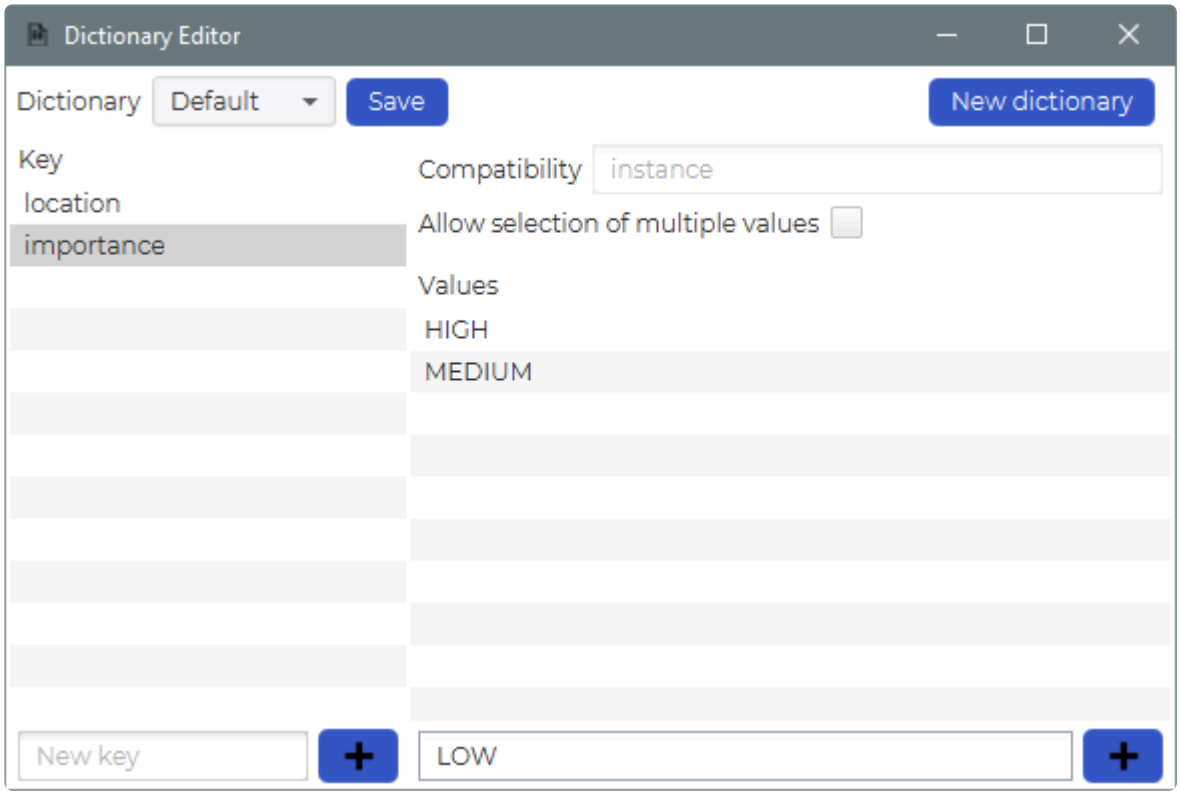
Now you should see the Editor.

It is divided into two main parts. On the left side you define the metadata keys, and on the right side you define the legal metadata values.



In the example above we have defined one key “location” and added the different continents as legal values.

To add a new metadata key, just type it into the field marked “New key” on the lower left and click the pluss sign.
For example we can add a key called “importance”, then type in HIGH, MEDIUM and LOW as legal values.



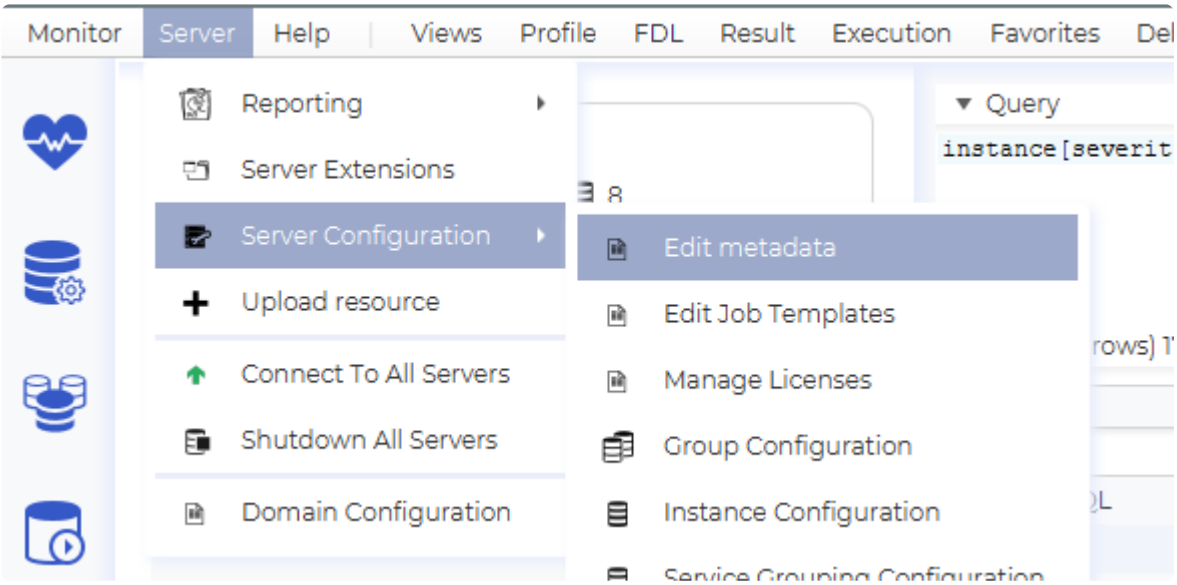
On the right side you can also see an uneditable field called “Compatibility” set to “instance”. That means that currently you can only assign this metadata to instances, but in the future other types will be supported.

Below is a checkbox called “Allow selection of multiple values”. If this is checked, it means that you can assign multiple values to a single key. For instance if you create a key called “application” with all your apllications as legal values, one instance might be assiciated with more than one application.

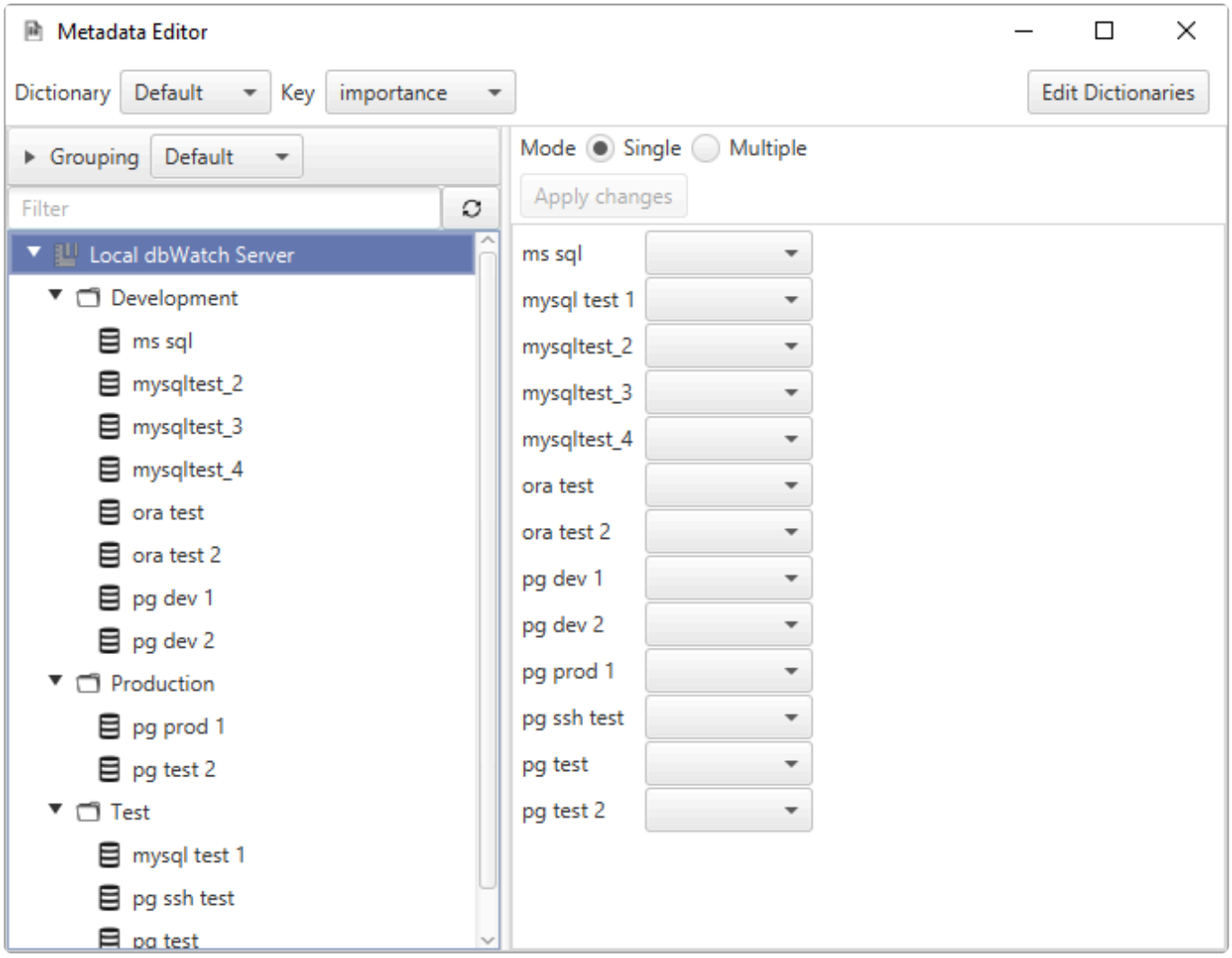
Metadata Editor

To assign metadata to instances in an efficient manner, you can use the Metadata Editor. (You can also edit metadata on a single instance through the normal “Configure instance” view)

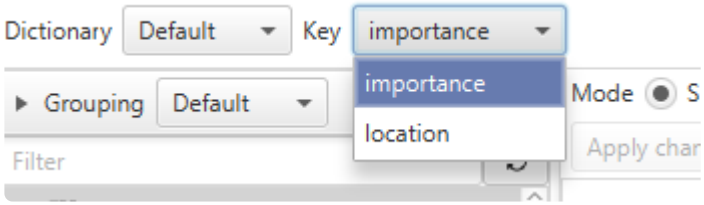
To open the Metadata editor, find “Edit metadata” on the “Configure” -> “Server Configuration” menu.



You should now see the Editor.



On the top you will see the different keys that are defined.

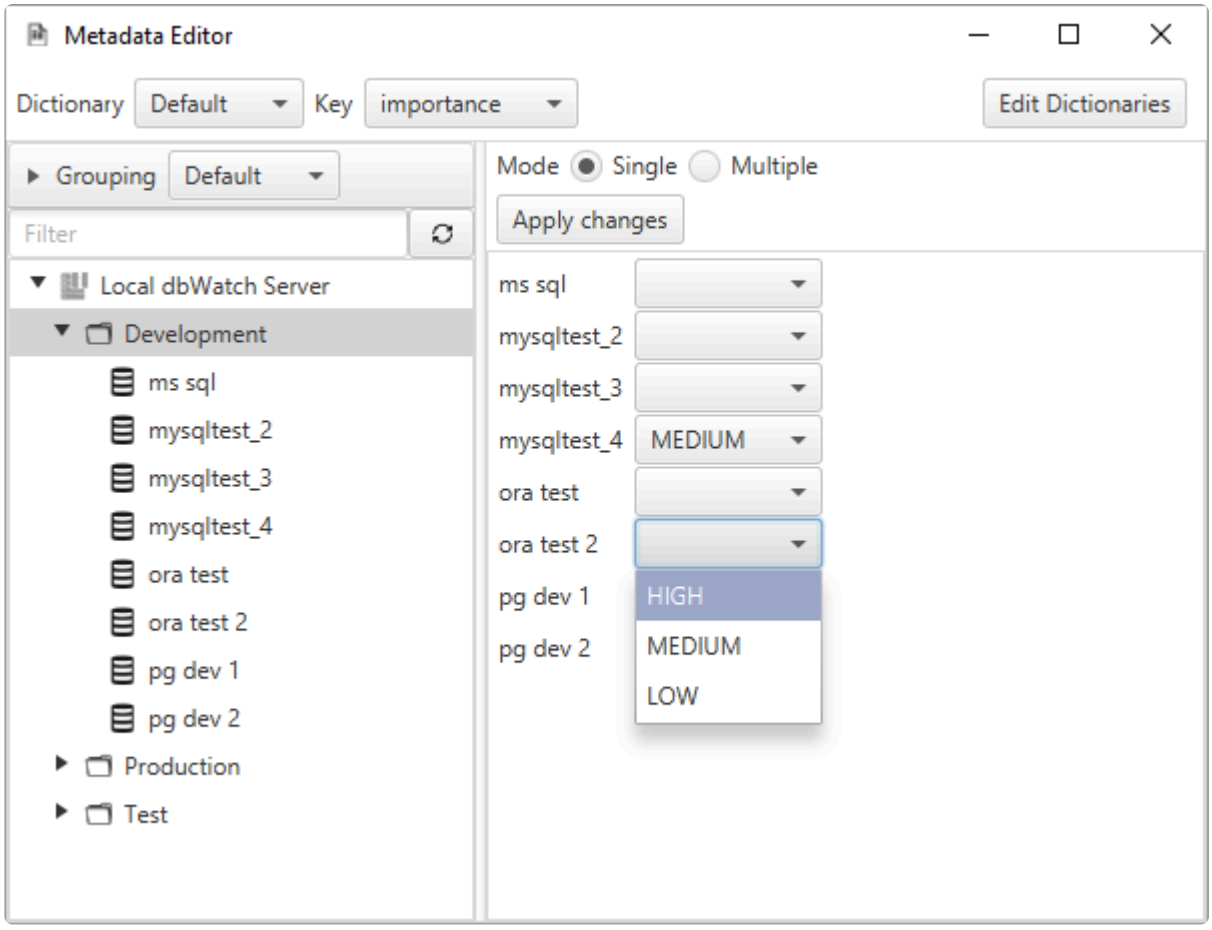


On the left you see a tree containing all the registered instances, and on the left a panel where values can be assigned.

The instances that are selected in the tree, are the ones you assign values to. So for example, if you select the “Development” group, the right side is populated with all the instances in that group.

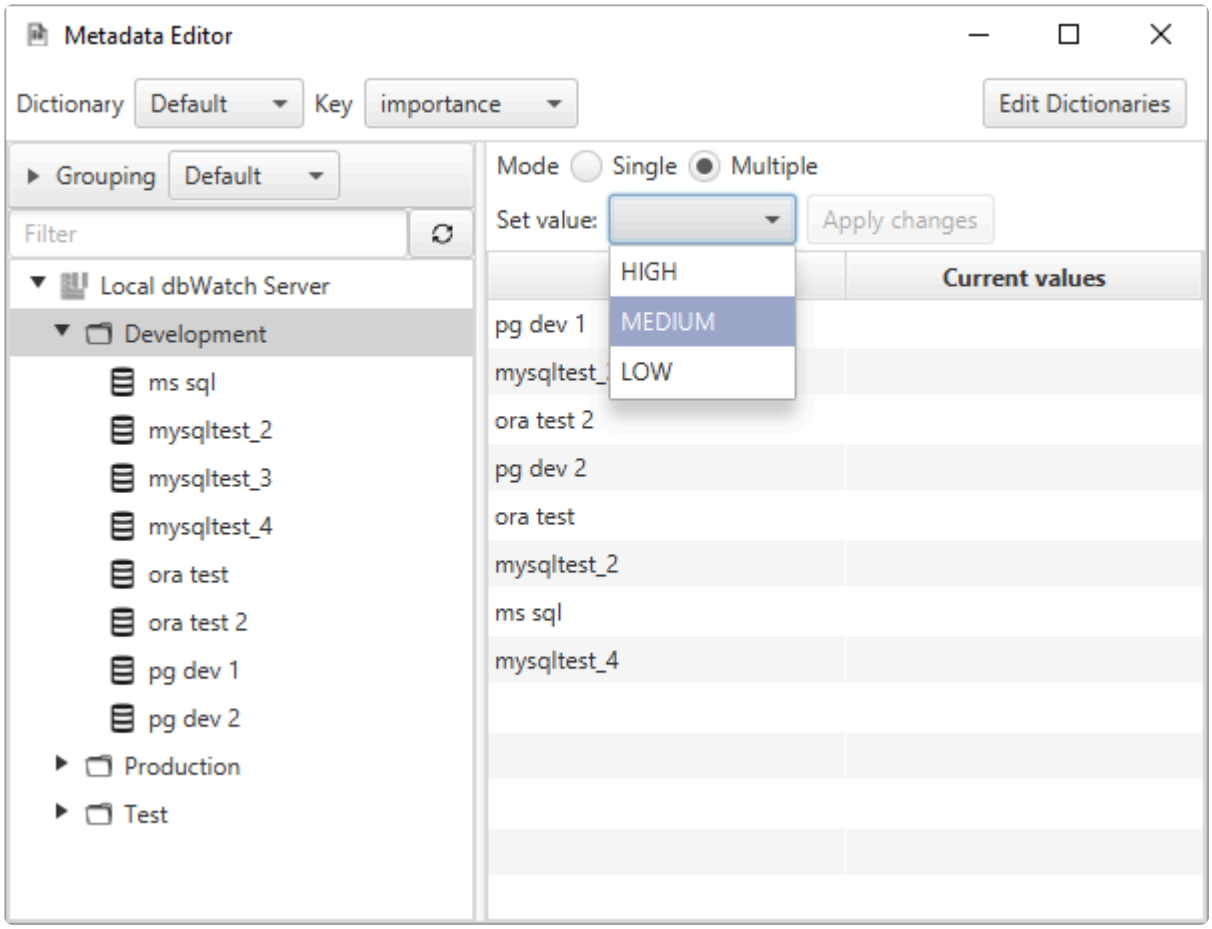
The panel on the right has two modes, “Single” and “Multiple”. Single means you can assign values to individial instances, while multiple lets you assign values to all instances at once.

First lets look at **Single** mode.



As you see, each instance has a dropdown where you can select the desired value for that instance. When you are satesfied, click apply.

Multiple mode



In multiple mode you have a single dropdown, and the value you select here will be applied to all the selected instances.

Instance selection

The metadata values already applied, can of course be used in the instance selection, by expanding the *Grouping" panel above the tree. See the Tree Navigation section for details.

<< Property System / Dynamic properties >>

Dynamic properties

Dynamic properties are defined in xml files located in the [dbWatch Server]/server/resources/properties/ catalog.

You cannot add files to this catalogue manually, but import them by placing them in the [dbWatch Server]/server/resources/import_area catalog.

Examples of property definitions:

```
<property>
  <key>version</key>
  <compatability>instance[databasetype='oracle']</compatability>
  <value engine="sql">select VERSION from v$instance</value>
  <valid-for>24h</valid-for>
</property>
<property>
  <key>active_session_count</key>
  <compatability>instance[databasetype='oracle']</compatability>
  <value engine="sql">select count(*) from v$session where status = 'ACTIVE' a
nd username is not null</value>
  <valid-for>5m</valid-for>
</property>
```

The fields are:

- **key** : The name that identifies this property (used in dbwql queries).
- **compatability** : FDL query that defines the database instances that this property is compatible with.
- **value** : An query that defines the property value.
- **valid-for** : How long this property can be cached. Format xh/xm/xs, examples “24h”, “15m”, “60s”. (hours/minutes/seconds cannot be combined, so “1m 30s” is not allowed and must be written as “90s”).

The **value** tag has an **engine** tag that specifies the type of the contained code. Legal values are currently: sql, fdl, ssh, javascript.
There can be several value tags, they form a sequence of steps that are executed when resolving the property.

[<< Metadata properties] / [Javascript property >>]

Javascript property

From dbWatch 12.2.1, one of the engines available in properties is javascript.

Example that sums each column from an sql result:

```
<property>
   <key>scripttest</key>
   <compatability>instance[databasetype='sqlserver']</compatability>
   <value engine="sql">select 1, 2, 3</value>
   <value engine="javascript">
      <![CDATA[
         try {
            var sum = 0;
            var rows = input.getRows();
            var j;
            for(j = 0; j < rows.length; j++)
            {
               var row = rows[j];
               var i;
               for(i = 0; i < row.length(); i++) {
                  sum = sum + row.get(i).asLong();;
               }
            }

            result.setProperty("scripttest", sum);
         } catch (err) {
            result.setProperty(2, err.messag
e);
         }
      ]]>
   </value>
   <valid-for>5s</valid-for>
</property>
```

In the javascript code there are two speicial objects to be aware of: **input** and **result**.

Input

If foreach=“row” or foreach=“column” is specified input contains the values for a row/column of the previous step as a ResultList.
Otherwise input contains all the values from the previous step as a Table object.

Table has the methods:

ResultList[] getRows() Returns an array of ResultList objects representing each row
ResultList[] getColumns() Returns an array of ResultList objects representing each column

ResultList has the methods:

int length() Returns the number of columns
ResultWrapper get(int colNumber) Returns the value for this column.

ResultWrapper has the methods:

long asLong() Returns the value as a long
long asLong(long default) Returns the value as a long, using the default if the result is null or not convertible to long
String asString() Returns the value as a String.
double asDouble() Returns the value as double, using 0 if the value is null or not convertible to double
double asDouble(double default) Returns the value as double, using the default if the value is null or not convertible to double

Result

Result is where you put the result of this step, it is a JavascriptPropertiesWrapper object.

JavascriptPropertiesWrapper has the following methods:

void setProperty(String name, Object value) Sets a property
void addProperty(String name, Object value) Adds a property
void setStatus(int status, String details) Sets the status. Used when the property is a standard task.
void setStatus(short status, String details) Sets the status. Used when the property is a standard task.
long toLong(long value) Utility function for converting String to long.
ResultWrapper last(String name, Object default) Gets the value this property had after the last execution.
ResultWrapper[] last(String name, Object default) Gets the values this property had after the last execution.

<< Dynamic properties / The property file format >>

The property file format

Any queriable node in the dbwatch server can have properties.
These can be statically set values or dynamically created.
This document describes the format used to define dynamic properties.

The file header

```
<properties>
        <asproperties>
                <name>Oracle properties</name>
                <version>1</version>
                <company>dbwatch.no</company>
                <group>no.dbwatch</group>
                <artifactid>oracle_properties</artifactid>
                <version>1</version>
        </asproperties>
        <property>
.......
</properties>
```

The property declarations

A simple property

A simple property decleration consists of:
- Key decleration. Which specifies which properties the declaration produces.
- Compability tag. Which specifies which queriable nodes this property chould be mounted under. This is specified using dbwql.
- Valid-for tag. Which specifies how long the property value should be cached.
- A series of value tags. Which contains the queries which will be executed during the resolution of this property.

```
        <property>
                <key>one</key>
                <compatability>instance[databasetype='oracle']</compatability>
                <value engine="sql">select 1 from dual</value>
                <valid-for>24h</valid-for>
        </property>
```

The key tag.

The key declaration is used to declare the structure, types and names of the output of the property declaration, as well as mapping to the output of value queries.

Nesting keys

Keys can be nested inside each other to create a hierarchy of properties.
A key declaration like this:

```
                <key name="tablespace" entityType="tablespace" column="NAME">
                        <key name="allocated" column="ALLOCATED"/>
                        <key name="used" column="USE
D"/>
                </key>
```

Can produce a graph like this.

```
        instance:myInstance
                |--tablespace=>tablespace:myTablespace@instance:myInstance
                |                |-allocated=>allocated:10G@tablespace:myTable
space@instance:myInstance
                |                |-used=>used:1G@tablespace:myTablespace@insta
nce:myInstance
                |
                |--tablespace=>tablespace:myOtherTablespace@instance:myInstanc
e
                |                |-allocated=>allocated:15G@tablespace:myOther
Tablespace@instance:myInstance
                |                |-used=>used:2G@tablespace:myOtherTablespac
e@instance:myInstance
```

Value mapping

A key can be declared to have a constant value. This can be usefull for structuring data or for creating flags.

```
    <key name="databasetype" entityType="databasetype" constant="oracle">
```

If not declared constant the key will get its value from the evaluation of the value declarations.
Some value declarations produce tables of values.
If so these will be mapped to the keys by giving the first value to the first key, the second value to the second key and so forth.
This behavior can be configured by setting the column attribute on each key, which wil map each key to the column with the declared name.

So the table:

Name	Available	Used
MyTablespace	10	1

and the keys:

```
                <key name="tablespace" entityType="tablespace">
                        <key name="allocated"/>
                        <key name="used"/>
                </key>
```

Would produce:

```
        instance:myInstance
                |--tablespace=>tablespace:myTablespace@instance:myInstance
                |                |-allocated=>allocated:10G@tablespace:myTable
space@instance:myInstance
                |                |-used=>used:1G@tablespace:myTablespace@insta
nce:myInstance
```

Installable properties

Parameters

```
    <property>
        .....
        <parameters>
            <parameter name="used_alarm_threshold" default="95" type="int"/>
            <parameter name="used_warning_threshold" default="90" type="int"/>
        </parameters>
```

[<< Javascript property](#) / [Management >>](#)

Management

Currently there is no graphical interface for editing the managment interface. However, everything is defined in xml files that can be edited directly. Documentation of the syntax is not complete, and you have to rely on the existing code for examples.

The following documentation is a work in progress, it should be correct but not complete.

When you have created a new xml file (or changed an existing file) it can be imported into dbWatch by placing it in the “[dbWatch Server]/server/resources/import_area” catalog.

[<< The property file format](#) / [Formatting table results >>](#)

Formatting table results

When you are displaying a table result based on an SQL query, you have the same render hints that are offered as part of dbwql.

To use these hints, add a “result-table-format” tag to the result.

```
<result>
   <sql>select a, b, c from sometable</sql>
   <result-type>TABLE</result-type>
   <result-table-format>
      <columns>
         <column>
            <number>2</number>
            <hashtag>#width(200)</hashtag>
         </column>
      </columns>
   </result-table-format>
</result>
```

The previous definition will force the second column of the result to a width of 200.

For a complete list of available hashtags, see the dbwql specification

<< Management / Result types >>

Result types

You can specift the result type with a “result-type” tag inside the result.

```
<result>
   <sql>select ... </sql>
   <result-type>table</result-type>
</result>
```

The currently available result types are

- **table / dbwqltable** Will display the result as a table. This is the default type.
- **code** Will display the result as a code block.
- **command** Will display an OK dialog if the statement succeeds.
- **graphical** Will display various charts (details).
- **text** Will display the result as text.
- **xml** Will attempt to format and display the result as xml.

<< Formatting table results / Graphical >>

Graphical

The graphical type

When specifying the graphical type there must also be a graphical tag inside the result.

```
<result>
   <sql>select ... </sql>
   <result-type>graphical</result-type>
   <graphical>
      ...
   </graphical>
</result>
```

This graphical tag then again contains the various chart specifications.

The currently supported charts are:

- **category-line-chart** Example
- **category-chart** Example
- **line-chart** Example
- **stacked-area-chart** Example
- **pie-chart** Example

Common tags

Tags that are common to all charts are:

title

The title.

```
<graphical>
   <title>Some title</title>
   ...
</graphical>
```

legend

Specifies details about the legend.
Contains an optional “side” specification, that specifies where the legend should be placed. Legal values are BOTTOM (default), LEFT, RIGHT, TOP
Contains an options “cols” specification that forces the number of columns in the legend.

```
<graphical>
   <legend>
      <side>RIGHT</side>
      <cols>2</cols>
   </legend>
   ...
</graphical>
```

top

Specifices an ordering and maximun number of rows for the resultset.

```
<graphical>
   <top>
      <column>A column name</column>
      <maxrows>20</maxrows>
      <order>asc</order>
   </top>
   ...
</graphical>
```

css

Can specify css for the chart.

For details of valid css referer to the JavaFX documentation, f.ex: JavaFX css tutorial

```
<graphical>
        <css>
                .default-color0.chart-area-symbol { -fx-background-color: #3e6
9a6, #233c5f; }
                .default-color1.chart-area-symbol { -fx-background-color: #66c
859, #296321; }
                .default-color2.chart-area-symbol { -fx-background-color: #e6e
6e6, #3f3f3f; }
                .default-color0.chart-series-area-line { -fx-stroke: #233c5f;
}
                .default-color1.chart-series-area-line { -fx-stroke: #296321;
}
                .default-color2.chart-series-area-line { -fx-stroke: #3f3f3f;
}
                .default-color0.chart-series-area-fill { -fx-fill: #3e69a6; }
                .default-color1.chart-series-area-fill { -fx-fill: #66c859; }
                .default-color2.chart-series-area-fill { -fx-fill: #e6e6e6; }
        </css>
```

```
        ...
</graphical>
```

y-range

You can specify the range of the y-axis (if relevant). It is specified in the form **from-to**, where from and to can be either a number or an *.

```
<graphical>
        ....
   <line-chart>
         <y-range>0-100</y-range>
          ...
   </line-chart>
</graphical>
```

min-x-range

You can specify a minimum x range (typically used when the x axis is a time period). Normally the range will be dynamic based on the data that is currently plotted, but sometimes it is relevant to say that the range should contain some minimum period. The value is number of seconds.

```
<graphical>
        ....
   <line-chart>
         <min-x-range>600</min-x-range>
          ...
   </line-chart>
</graphical>
```

functions

There are a set of functions that can be applied to the values before plotting.

- **negative_to_zero** Converts negative values to 0
- **negative_to_last** Converts negative values to the last known value
- **multiply** Multiplies the value by a constant
- **divide** Divides the value by a constant

The negative_to_zero and negative_to_last functions can be applied to specific columns or to all columns.

All columns:

```
<graphical>
```

```
   <functions>
      <function>negative_to_last</function>
   </functions>
   ...
</graphical>
```

A column called ‘counter’:

```
<graphical>
   <functions>
      <function>
            <perform>negative_to_last</perform>
            <column>counter</column>
      </function>
   </functions>
   ...
</graphical>
```

The multiply and divide functions can only be applied to specific columns. Here values in the column ‘counter’ are multipled by 10:

```
<graphical>
   <functions>
      <function>
            <perform>multiply</perform>
            <args>
                  <arg>10.0</arg>
            </args>
            <column>counter</column>
      </function>
   </functions>
   ...
</graphical>
```

Dynamic chart tags

max-age

The max age (in seconds) for plotted values. Relavant when the x axis displays time. Typically used to specify that you only want to plot values from the last x hours/minutes.

```
<graphical>
   <max-age>7200</max-age>
   ...
```

```
</graphical>
```

max-points

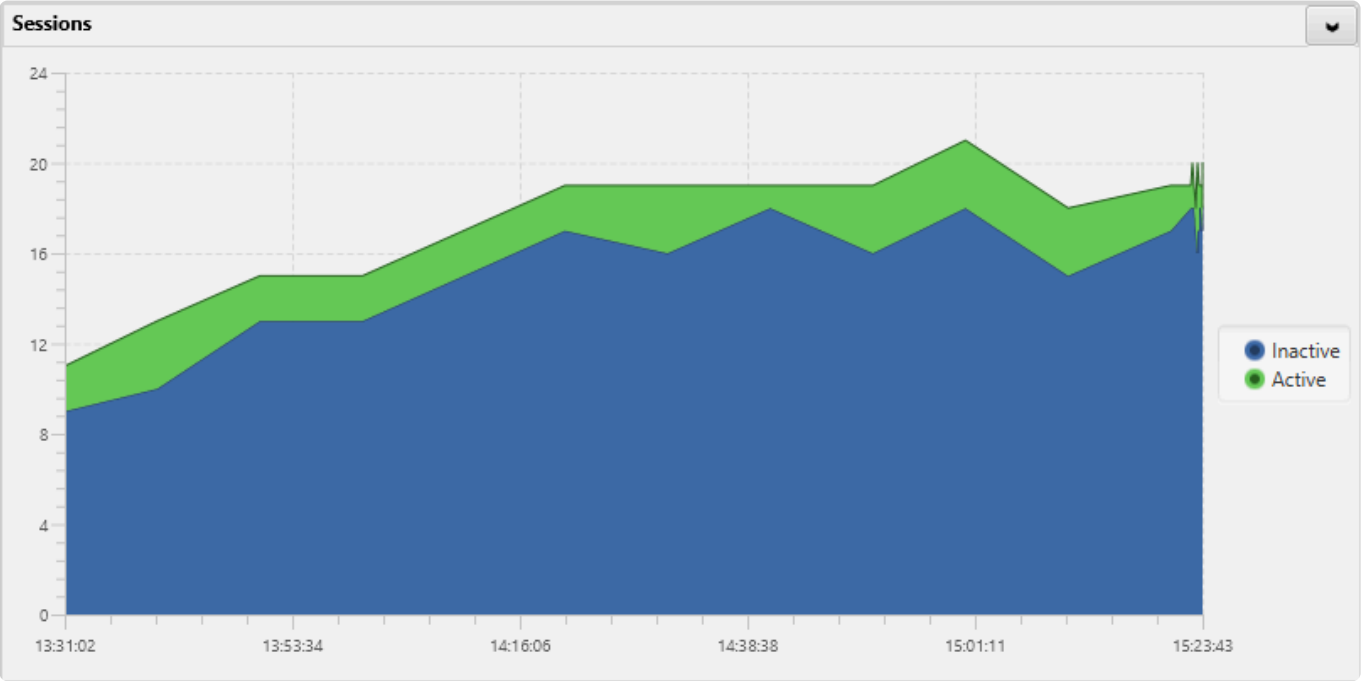
The maximum number of points to plot. This is a feature to avoid issues when accidentally trying to plot a large number of points. There is a threshold where you will have 100% cpu load trying to plot everything.

```
<graphical>
   <max-points>1000</max-points>
   ...
</graphical>
```

<< Result types / Stacked area chart >>

Stacked area chart

Here is an example of a stacked area chart showing the relationship between active and inactive sessions over time.



The specification for this chart:

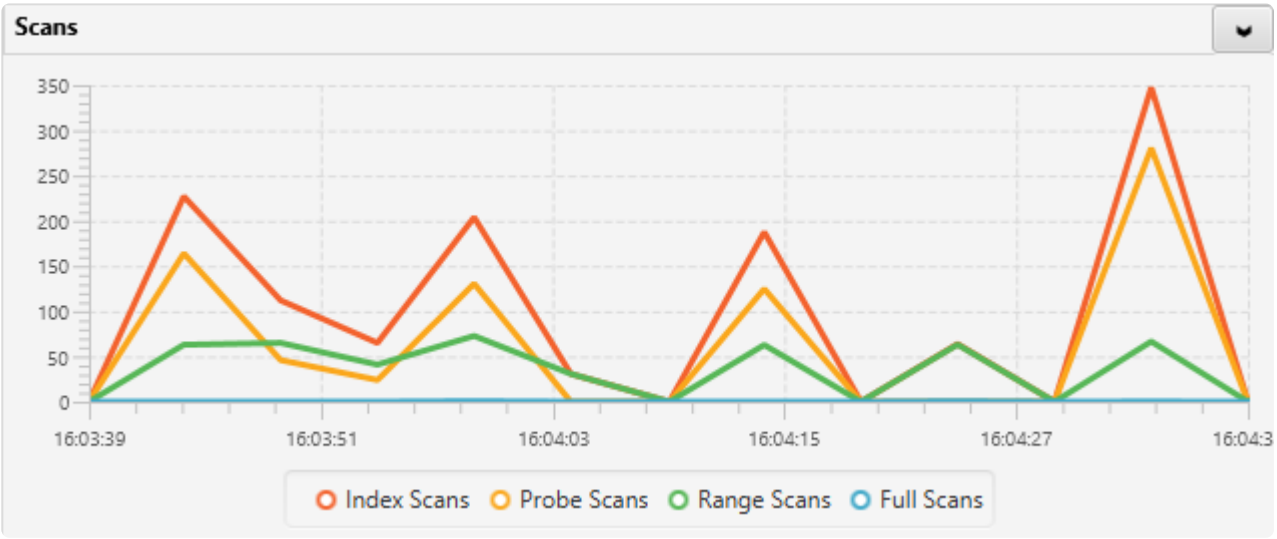
```
<result>
        <query engine="sql" type="static">
                select number_of_processes_active as "Active", number_of_proce
sses_total - number_of_processes_active as "Inactive", histr_date as "History
date"
                from #ARG_ENGINE_SCHEMA#..dbw_session_load_histr order by 3
        </query>
        <dbwql>
                instance/$instance/active_session_count_ts{"Active"}/$instanc
e/inactive_session_count_ts{"Inactive"}/timestamp{"History date"}
        </dbwql>
        <result-type>GRAPHICAL</result-type>
        <graphical>
                <title>Sessions</title>
                <interval>10</interval>
                <dynamic>true</dynamic>
                <max-age>7200</max-age>
                <resultset>
                        <column>
                                <name>Active</name>
                                <sql-type>1</sql-type>
                                <label/>
```

```
                        </column>
                        <column>
                                <name>Inactive</name>
                                <sql-type>1</sql-type>
                                <label/>
                        </column>
                        <column>
                                <name>History date</name>
                                <sql-type>10</sql-type>
                                <label/>
                        </column>
                </resultset>
                <legend-side>RIGHT</legend-side>
                <stacked-area-chart>
                        <to-plot>Inactive</to-plot>
                        <to-plot>Active</to-plot>
                        <time>History date</time>
                        <chart-type>5</chart-type>
                </stacked-area-chart>
                <css>
                .default-color0.chart-area-symbol { -fx-background-color: #3e6
9a6, #233c5f; }
                .default-color1.chart-area-symbol { -fx-background-color: #66c
859, #296321; }
                .default-color2.chart-area-symbol { -fx-background-color: #e6e
6e6, #3f3f3f; }
                .default-color0.chart-series-area-line { -fx-stroke: #233c5f;
}
                .default-color1.chart-series-area-line { -fx-stroke: #296321;
}
                .default-color2.chart-series-area-line { -fx-stroke: #3f3f3f;
}
                .default-color0.chart-series-area-fill { -fx-fill: #3e69a6; }
                .default-color1.chart-series-area-fill { -fx-fill: #66c859; }
                .default-color2.chart-series-area-fill { -fx-fill: #e6e6e6; }
                </css>
        </graphical>
</result>
```

<< Graphical / Line chart >>

Line chart

A line chart example



The specification for this chart:

```
<result>
        <sql>select
                (SELECT cntr_value FROM sys.dm_os_performance_counters
                where counter_name = 'Index Searches/sec') as "Index Scans",
                (SELECT cntr_value FROM sys.dm_os_performance_counters
                where counter_name = 'Probe Scans/sec') as "Probe Scans",
                (SELECT cntr_value FROM sys.dm_os_performance_counters
                where counter_name = 'Range Scans/sec') as "Range Scans",
                (SELECT cntr_value FROM sys.dm_os_performance_counters
                where counter_name = 'Full Scans/sec') as "Full Scans"
        </sql>
        <result-type>GRAPHICAL</result-type>
        <graphical>
                <title>Scans</title>
                <interval>5</interval>
                <dynamic>true</dynamic>
                <resultset>
                        <column>
                                <name>Index Scans</name>
                                <sql-type>1</sql-type>
                                <diff>1</diff>
                                <label/>
                        </column>
                        <column>
                                <name>Probe Scans</name>
                                <sql-type>1</sql-type>
```

```
                                <diff>1</diff>
                                <label/>
                        </column>
                        <column>
                                <name>Range Scans</name>
                                <sql-type>1</sql-type>
                                <diff>1</diff>
                                <label/>
                        </column>
                        <column>
                                <name>Full Scans</name>
                                <sql-type>1</sql-type>
                                <diff>1</diff>
                                <label/>
                        </column>
                </resultset>
                <line-chart>
                        <to-plot>Index Scans</to-plot>
                        <to-plot>Probe Scans</to-plot>
                        <to-plot>Range Scans</to-plot>
                        <to-plot>Full Scans</to-plot>
                        <chart-type>5</chart-type>
                </line-chart>
        </graphical>
</result>
```

You can change the colors of the chart by adding a css tag like this:

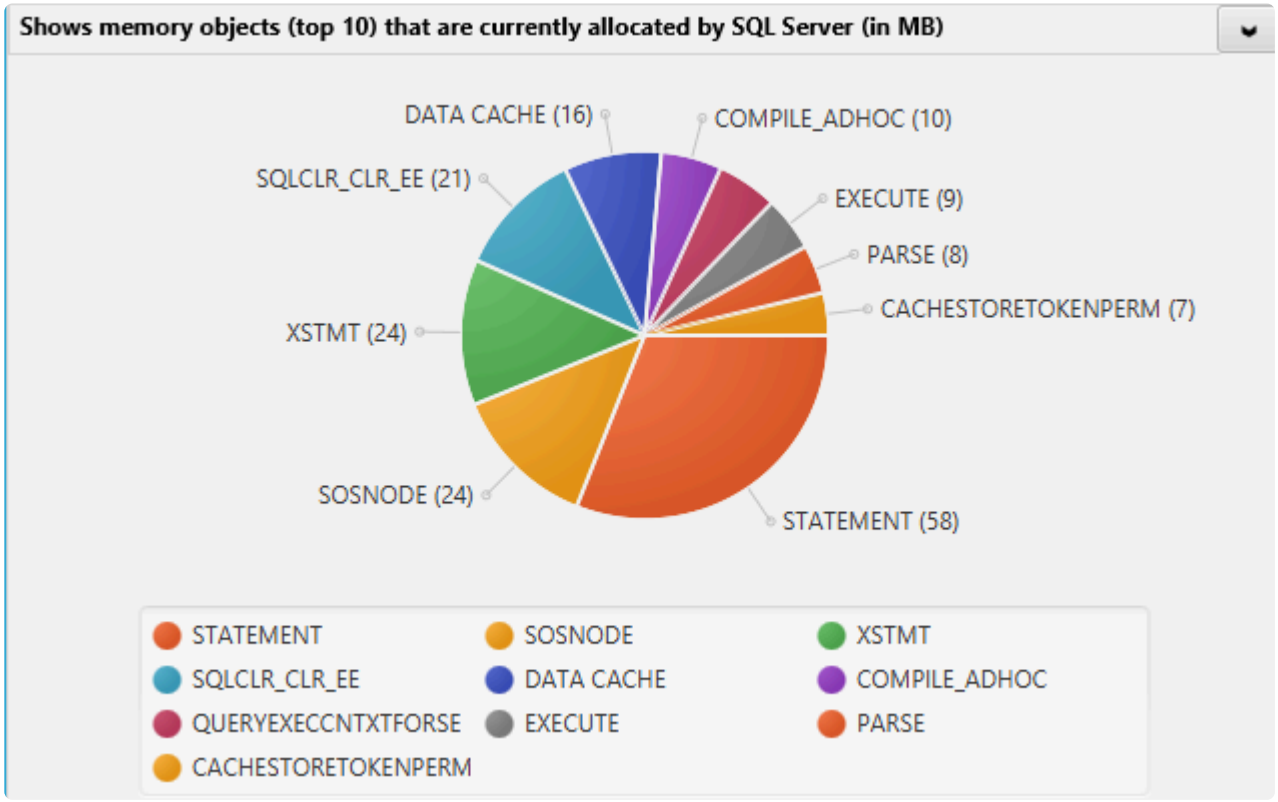
```
...
        </line-chart>
        <css>
                .default-color0.chart-series-line { -fx-stroke: #185AA9; }
                .default-color0.chart-line-symbol { -fx-background-color: #185
AA9, white; }
                .default-color1.chart-series-line { -fx-stroke: #f0E68C; }
                  .default-color1.chart-line-symbol { -fx-background-color: #f
0E68C, white; }
        </css>
</graphical>
```

A css tag like this would change the color of the first line to #185AA9 and the second line to #f0E68C (line 3 and 4 would still use default colors).
You can add as many lines as you need by specifying the corresponding number X in ".default-colorX.chart…"

[<< Stacked area chart](#) / [Pie chart >>](#)

Pie chart

Pie chart example



The specification for this chart

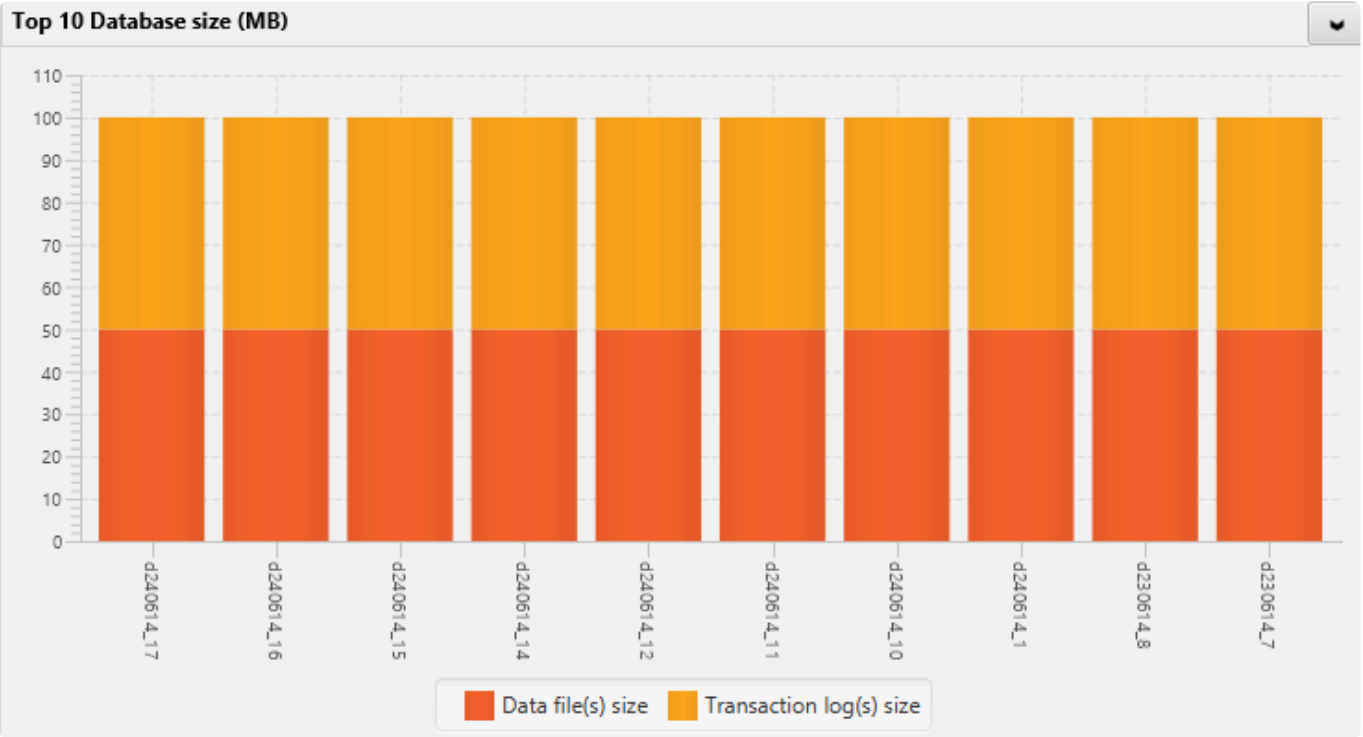
```
<result>
        <sql>select top 10 'type' = replace(A.type, 'MEMOBJ_', ''), 'value' =
A.value from (
                SELECT type, SUM (cast((pages_in_bytes/1024) as int) )/1024 a
s "value"
                from master.sys.dm_os_memory_objects
                group by type
                union
                SELECT
                'DATA CACHE', cast(round(count(*)/128.,0) AS int)
                FROM sys.dm_os_buffer_descriptors ) A
                order by 2 desc
        </sql>
        <result-type>GRAPHICAL</result-type>
        <graphical>
                <title>Shows memory objects (top 10) that are currently alloca
ted by SQL Server (in MB)</title>
                <resultset>
                        <column>
                                <name>type</name>
```

```
                                <sql-type>0</sql-type>
                                <label/>
                        </column>
                        <column>
                                <name>value</name>
                                <sql-type>1</sql-type>
                                <label>MB</label>
                        </column>
                </resultset>
                <pie-chart>
                        <datalabels>type</datalabels>
                        <to-plot>value</to-plot>
                </pie-chart>
        </graphical>
</result>
```

<< Line chart / Category chart >>

Category chart

A catagory chart example



The specification for this chart

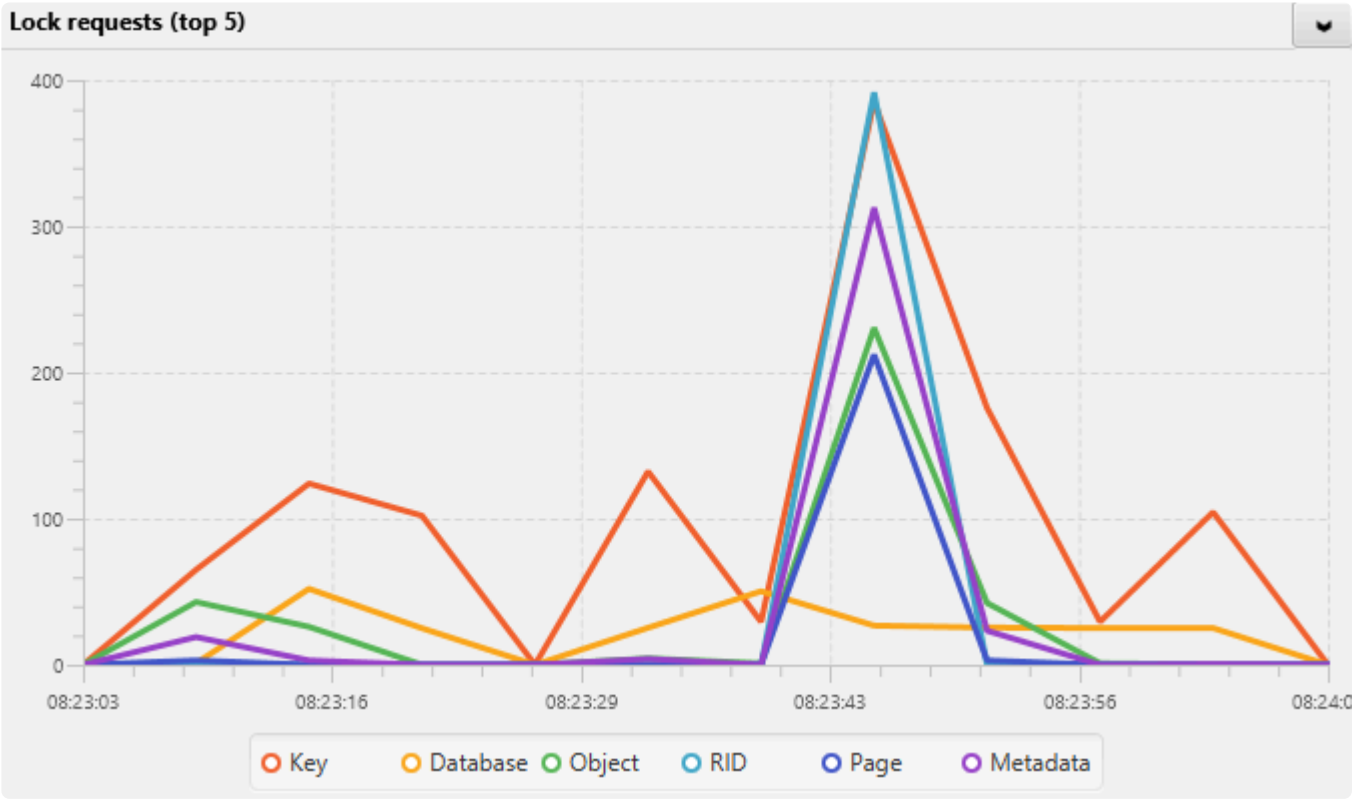
```
<result>
        <result-type>GRAPHICAL</result-type>
        <sql>
                SELECT top 10
                case when len(DB.name) &gt; 30 then substring(DB.name,1,3
0)+'_DBID_'
                        +cast(DB.database_id as varchar(4)) +' (alias)' else D
B.name end AS
                        "Database name",
                (SELECT SUM((size*8)/1024) FROM sys.sysaltfiles WHERE DB_NAM
E(dbid) =
                        DB.name AND groupid!=0)+
                (SELECT SUM((size*8)/1024) FROM sys.sysaltfiles WHERE DB_NAM
E(dbid) =
                        DB.name AND groupid=0) "Total size",
                (SELECT SUM((size*8)/1024) FROM sys.sysaltfiles WHERE DB_NAM
E(dbid) =
                        DB.name AND groupid!=0) AS "Data file(s) size",
                (SELECT SUM((size*8)/1024) FROM sys.sysaltfiles WHERE DB_NAM
E(dbid) =
                        DB.name AND groupid=0) AS "Transaction log(s) size",
```

```
                (SELECT COUNT(1) FROM sys.sysaltfiles WHERE DB_NAME(dbid) = D
B.name )
                        AS "Total file/log count"
                FROM sys.databases DB where DB.name not in (N'master',N'mode
l',N'msdb',N'tempdb')
                order by 2 desc
        </sql>
        <graphical>
                <title>Top 10 Database size (MB)</title>
                <resultset>
                        <column>
                                <name>Database name</name>
                                <sql-type>0</sql-type>
                                <label></label>
                        </column>
                        <column>
                                <name>Total size</name>
                                <sql-type>1</sql-type>
                                <label>MB</label>
                        </column>
                        <column>
                                <name>Data file(s) size</name>
                                <sql-type>1</sql-type>
                                <label>MB</label>
                        </column>
                        <column>
                                <name>Transaction log(s) size</name>
                                <sql-type>2</sql-type>
                                <label>MB</label>
                        </column>
                        <column>
                                <name>Total file/log count</name>
                                <sql-type>6</sql-type>
                                <label></label>
                        </column>
                </resultset>
                <category-chart>
                        <categories>Database name</categories>
                        <chart-type>5</chart-type>
                        <to-plot>Data file(s) size</to-plot>
                        <to-plot>Transaction log(s) size</to-plot>
                </category-chart>
        </graphical>
</result>
```

[<< Pie chart](#) / [Category line chart >>](#)

Category line chart

A category line chart example



The specification for this chart

```
<result>
        <sql>
                SELECT rtrim(ltrim(instance_name)) "Type", cntr_value "Value"
                FROM sys.dm_os_performance_counters
                where counter_name like  'Lock Requests/sec%' and instance_nam
e in (
                select top 6 instance_name FROM sys.dm_os_performance_counter
s
                where counter_name like 'Lock Requests/sec%' and instance_nam
e != '_Total'
                order by cntr_value desc )
                order by 2 desc
        </sql>
        <result-type>GRAPHICAL</result-type>
        <graphical>
                <title>Lock requests (top 5)</title>
                <interval>6</interval>
                <dynamic>true</dynamic>
                <functions>
                    <function>negative_to_last</function>
```

```
                </functions>
                <resultset>
                        <column>
                                <name>Type</name>
                                <sql-type>1</sql-type>
                                <diff>1</diff>
                        <label/>
                        </column>
                        <column>
                                <name>Value</name>
                                <sql-type>1</sql-type>
                                <diff>1</diff>
                                <label/>
                        </column>
                </resultset>
                <category-line-chart>
                        <to-plot>Value</to-plot>
                        <chart-type>3</chart-type>
                        <categories>Type</categories>
                </category-line-chart>
        </graphical>
</result>
```

[<< Category chart](#) / [Item active >>](#)

Item active

You can hide or disable elements (typically nodes or menu items) by using the item-active tag

You can specify multiple item-active tags, and all of them have to resolve to “active” in order for the element to be classified as active.

The item-active tag can specify either an sql query, a dbwql query or a “text” to be exexuted. This is done with a “sql”, “dbwql” or “text” tag.

(You can always add an “invert” tag set to true to flip the result.)
(inactive-mode can be “hide” or “disable”)

Dbwql “queries”

```
<menu-item>
        <item-active>
                <dbwql>$instance/is_readwrite</dbwql>
                <inactive-mode>disable</inactive-mode>
        </item-active>
        ...
</menu-item>
```

The result from the query is parsed as follows:

A positive number	Active
0 or a negative number	Inactive
Empty result or empty string	Inactive
A non empty result, that is not a number	Active

SQL queries

If the sql query is a block you can add a “pl-sql” tag set to true.
If the query can potentially take a long time, you can add a “timeout” tag specifying the number of seconds to wait.

```
<menu-item>
        <item-active>
                <sql>select count(*) from sometable</sql>
                <pl-sql>false</pl-sql>
                <inactive-mode>hide</inactive-mode>
                <use-cache>true</use-cache>
        </item-active>
```

```
        ...
</menu-item>
```

The result is parsed as:

A value of 0	Inactive
Everything else	Active

Text “queries”

Text “queries” are a bit different.

They compare values that are resolved by other means than sql or dbwql (f.ex through menu selections, or based on the node you are on)

They specify a series of “cases” that are checked, and the first case to “hit” gives the result. If no cases hits, the result is considered “active” (unless you have added the optional “default” tag to change the default)

A case tag contains a value that will be checked. If the case also contains an “equals” tag, this is used for comparing. If there is no equals tag, any value except the empty string is considered a hit.

You do not have to specify both an “active” and “inactive” tag (usually you will just have one of them)

```
<item-active>
        <text>
                <default>active</default>
                <active>
                        <case>
                                <value>#ARG_OPTION_NAME#</value>
                                <equals>Some name</equals>
                        </case>
                        <case>
                                ..
                        </case>
                </active>
                <inactive>
                        <case>
                                <value>#ARG_OPTION_NAME#</value>
                                <equals>Some other name</equals>
                        </case>
                        <case>
                                ..
                        </case>
                </inactive>
```

```
        </text>
</item-active>
```

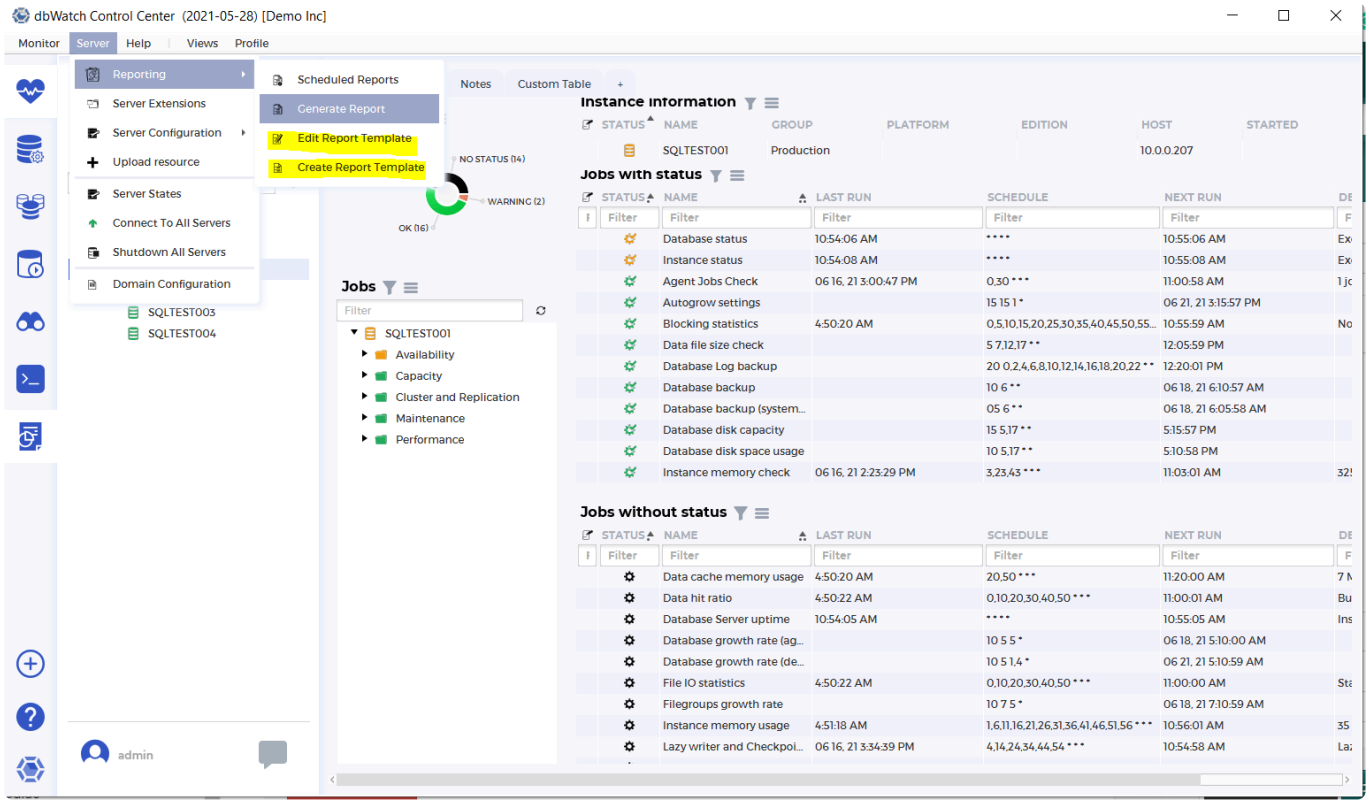
[<< Category line chart](#) / [Reports >>](#)

Reports

Before proceeding to this topic, you must be familiar with generating a report. You can check it out by click on the link.

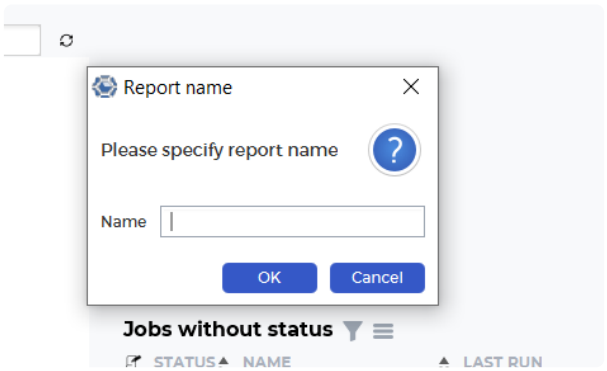
Customizing your Report by changing the logo and Company Name

To customize, your report. First click “Server” on the upper ribbon of your dbWatch. Then, hover over “Reporting”. You can select to either create or edit a template. For this process, we will focus first on creating a template. The processes are similar to each other.

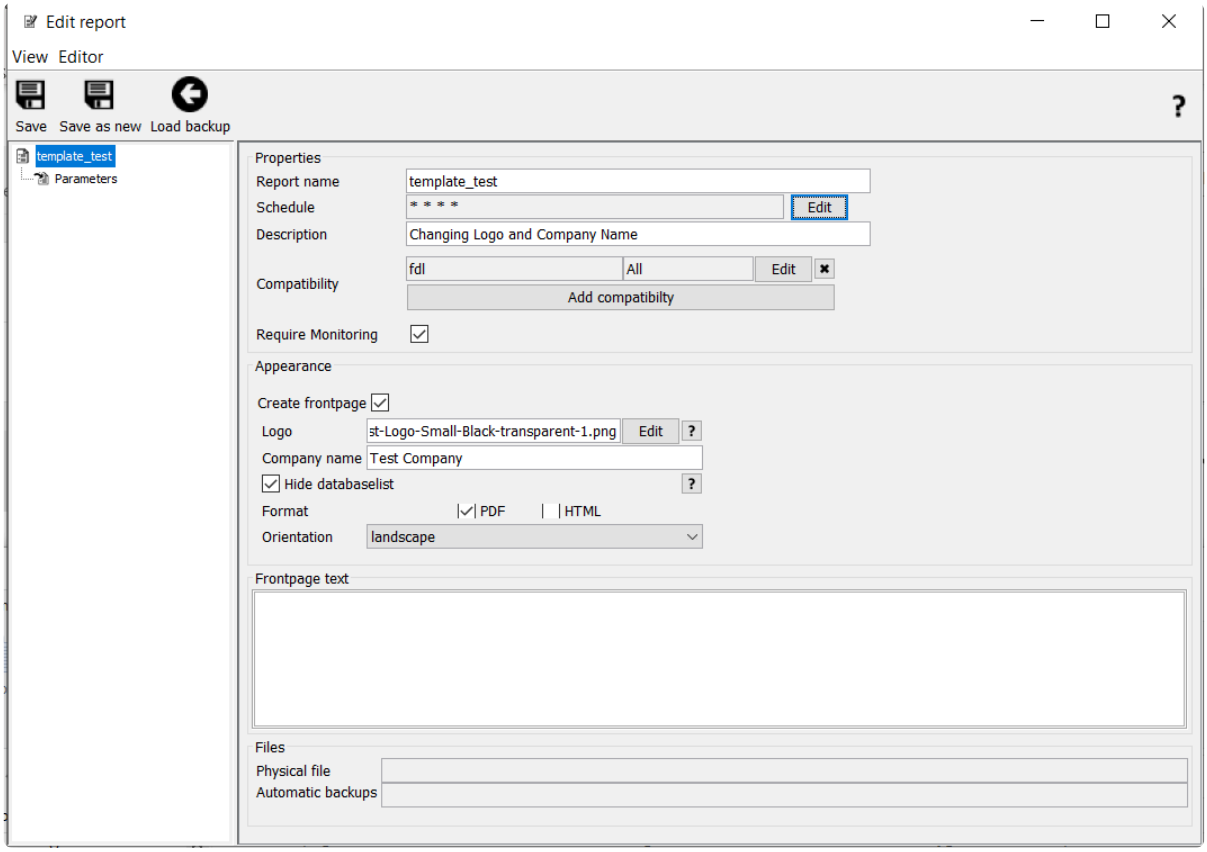


Creating a Report

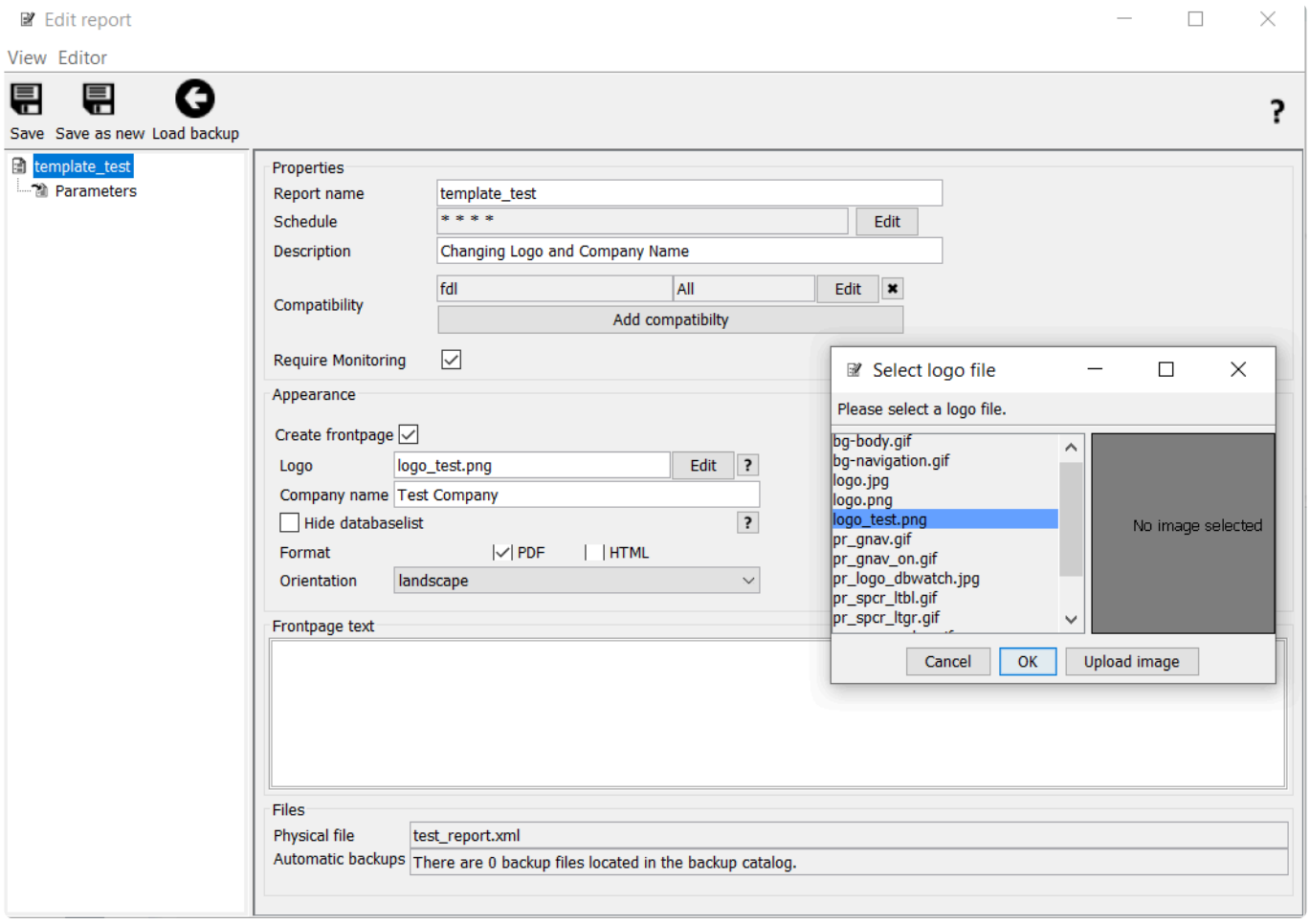
Click on “Create Report Template”. A window will appear asking you to specify the name of the template.



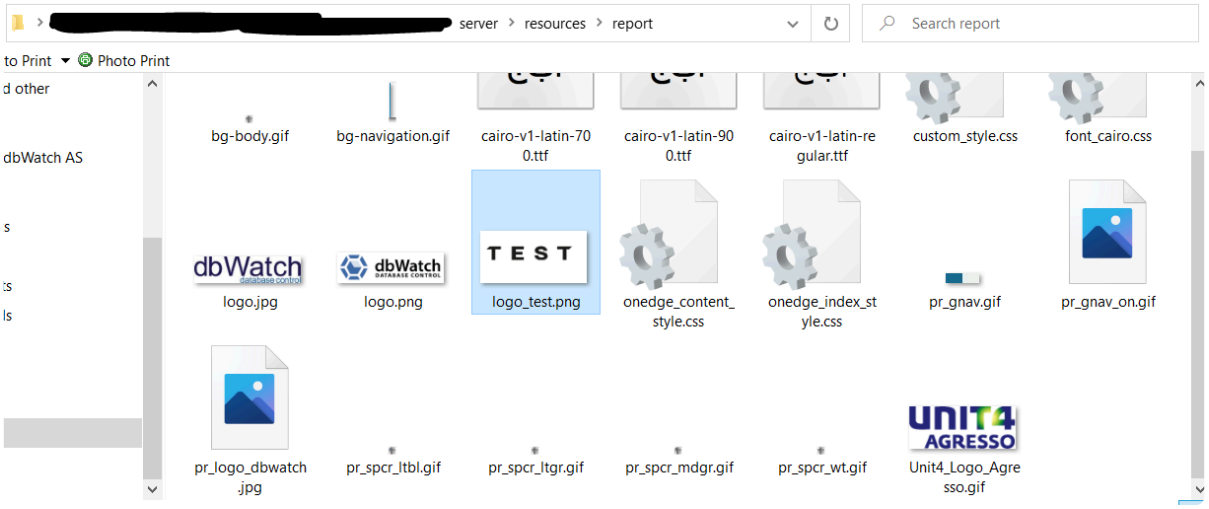
We'll name the template as *testtemplate*. Once you clicked, **OK**, a window will appear. You can edit the template in this window.



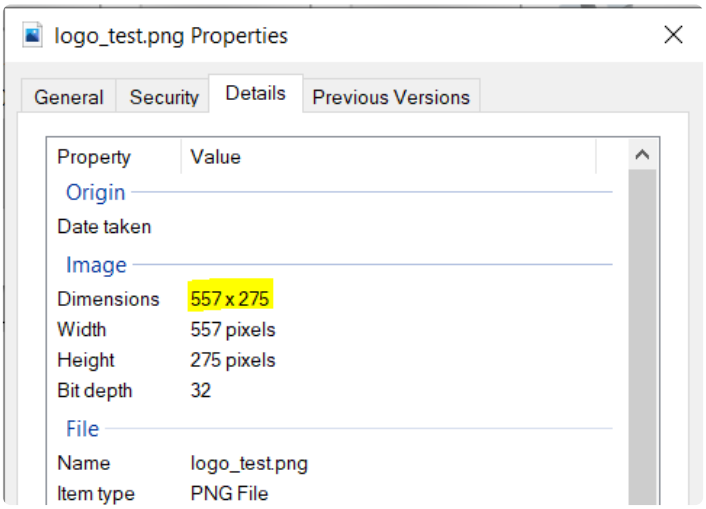
To edit the logo, click the "Edit" button in line with the Logo label. A window showing all images of logos will appear. You can either upload the logo or copy the logo image to *C:\Program Files\ControlCenter\server\resources\report*. Take note that that's the default destination directory of dbWatch Control Center. You should know what your default directory should be to manually add an image.



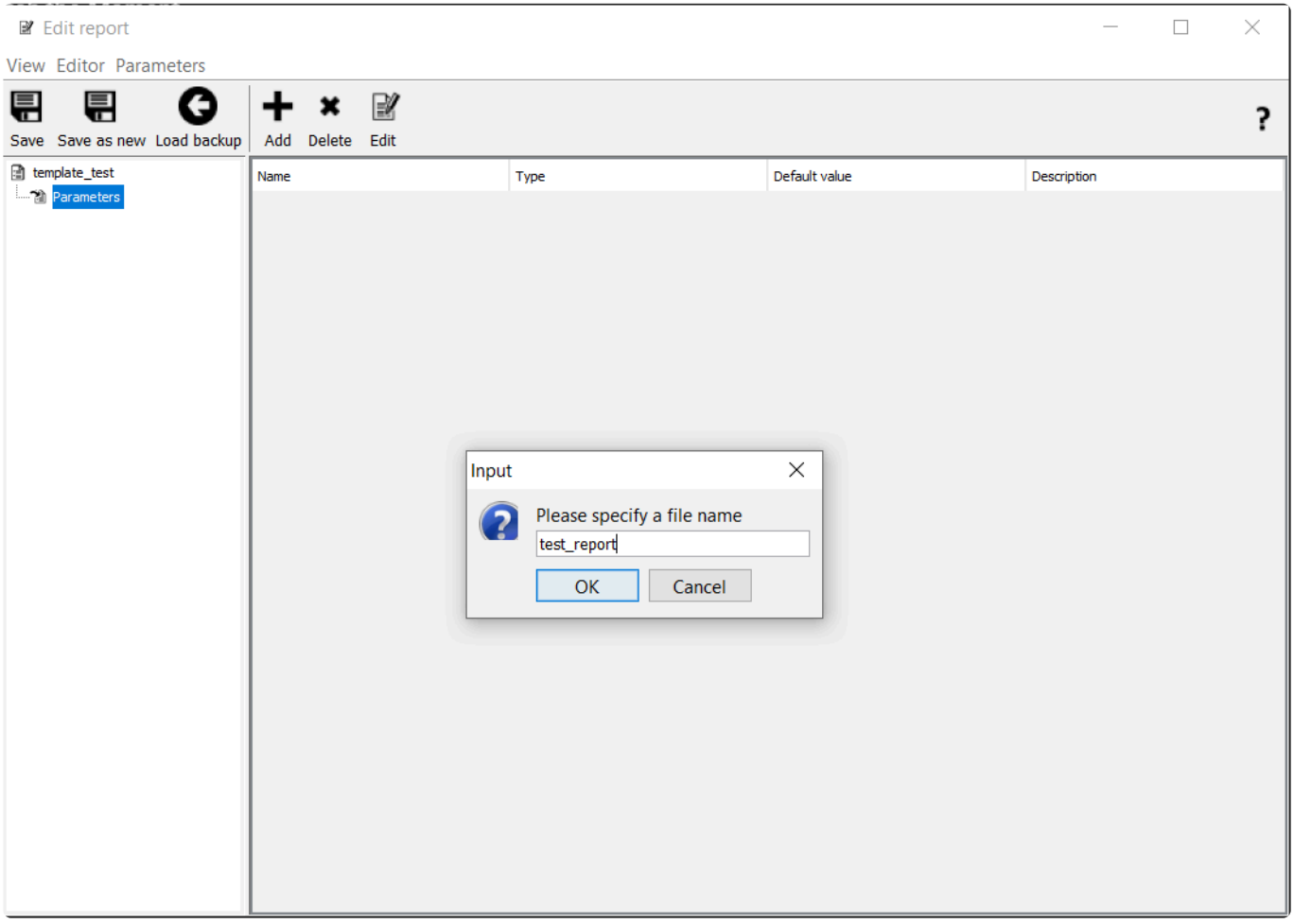
p(banner important).You can manually drag the logo image from one directory to another. The logo should appear in your list of choices.



> **!** More importantly, always keep your eye on the dimensions of your logo image. dbWatch will not scale up or scale down the image and will take it in its absolute dimension. This means that if you use a logo having dimensions of more than 600 by 600 pixels, it might eat the entire bottom half of your report. Our recommendation is to keep it as small as possible. Less than 200 pixels is good.

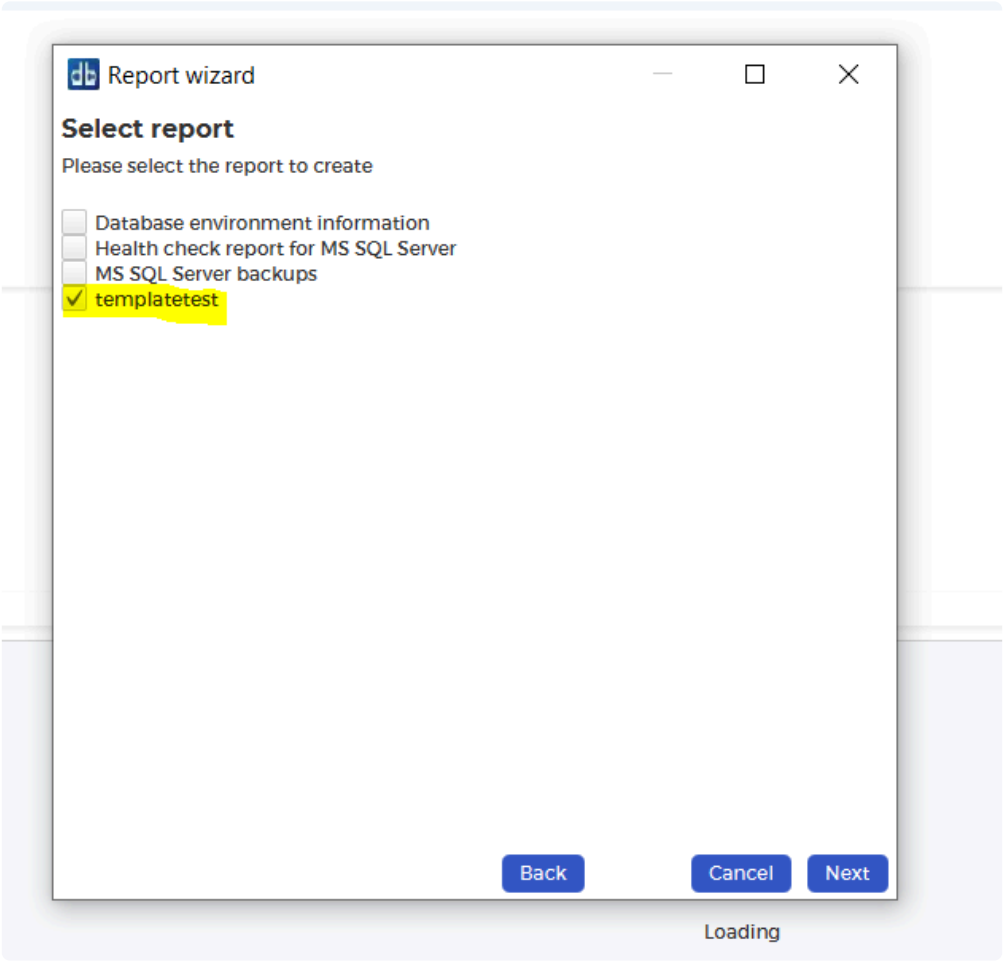


After adding the logo image, provide the organization's name in the field **Company Name**. Change the other settings base on what you need. Click "Save" or Editor > Save.

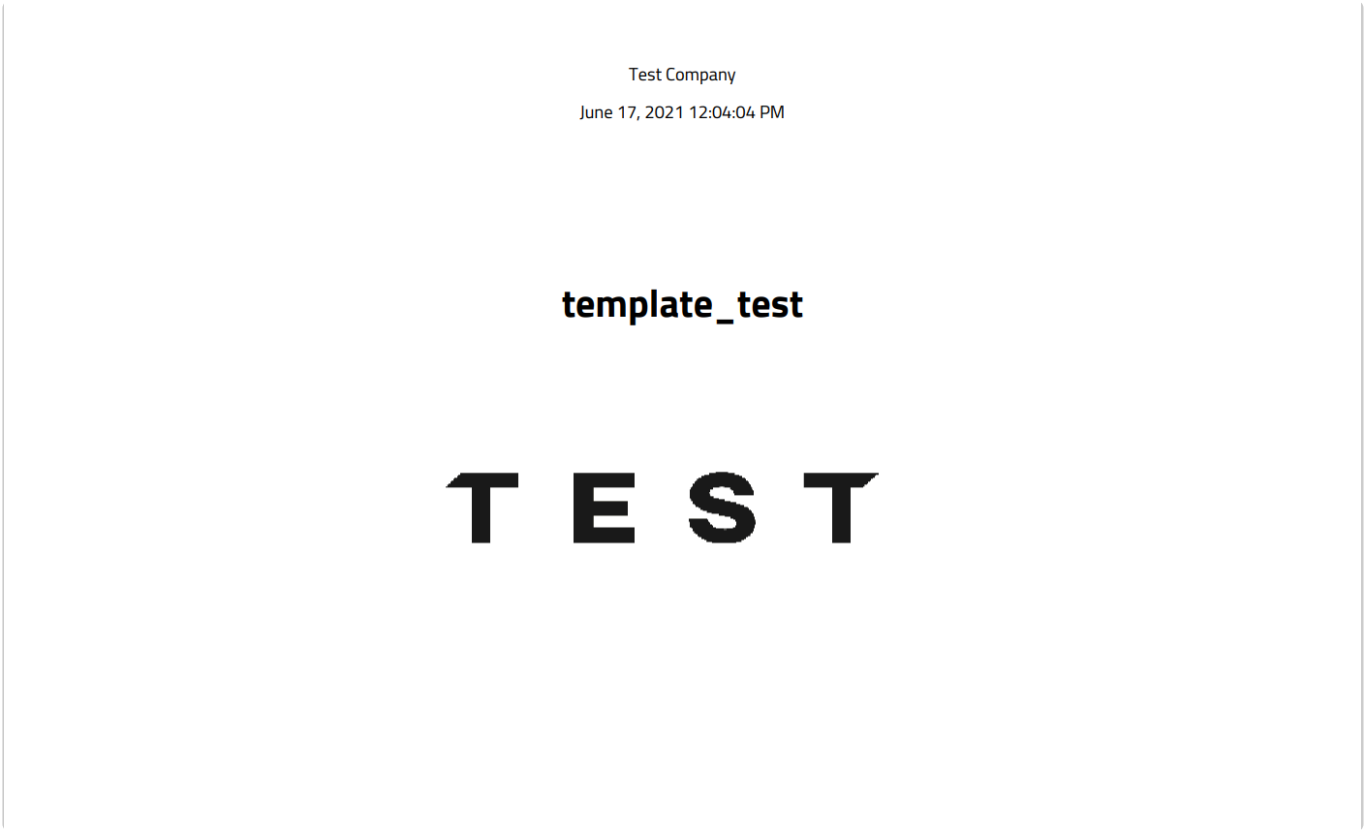


Provide the Physical Filename of the template. Click "OK" and you're done. To Test if your report is working, click on the Reporting Module and go to the steps of generating a report.

Make sure you select the correct report when generating it.

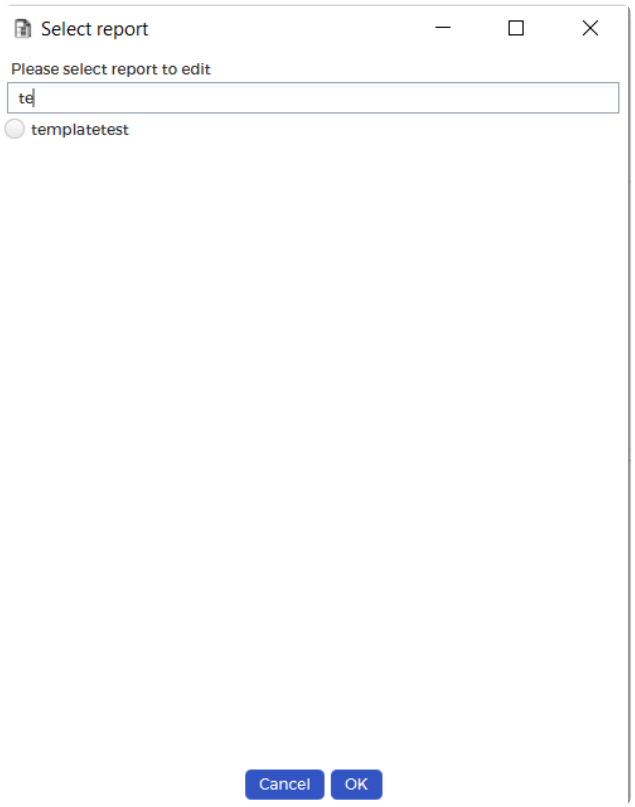


You can see the changes in the Logo and Report once it's generated:



Editing a report

Editing a report is similar to creating one. The only difference is that you need to select an existing report before proceeding. Once you clicked on “Edit Report Template”, you will see the window below:



Follow all the steps in the previous section to edit the report. There’s little difference when editing a template.

[<< Item active](#) / [Extensions and 3rd party integrations >>](#)

Extensions and 3rd party integrations

Configuring extensions and 3rd party integrations

- Email

Contact support if you want to make sure you have installed the latest version of your extension.

<< Reports / Email and SMS >>

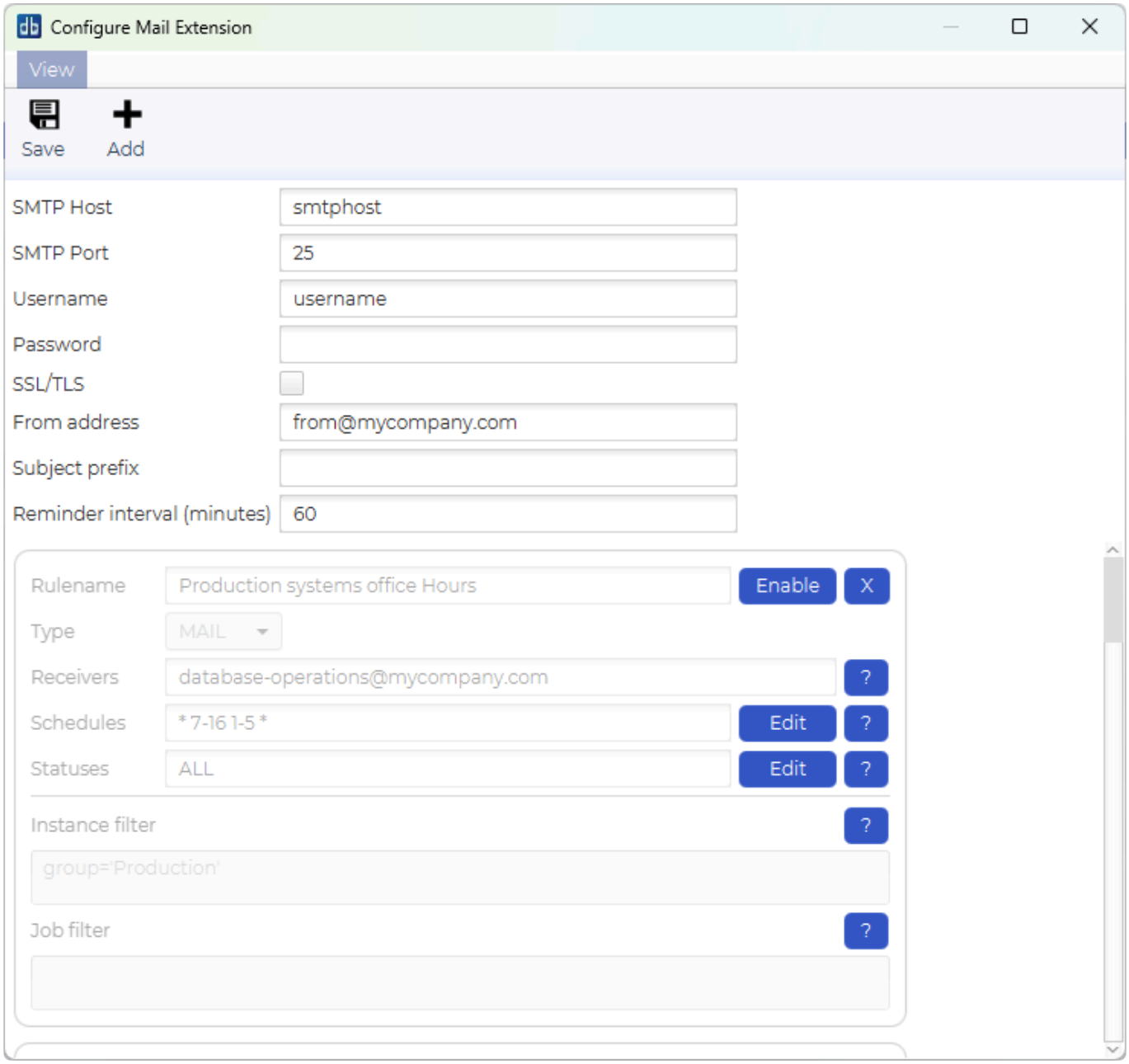
Email and SMS

Configure

In the dbWatch Monitor, go to the “Server”->”Server Extensions” menu. An item called “E-mail extension” should now be visible.

To configure the extension, select it and click “Configure”.

You should now see the initial configuration screen (below)



The first section lets you define connection details to the SMTP server.

SMTP Host is the hostname or IP address for the email server.
SMTP Port is the port the SMTP process is listening to.

Username is the used when you need to provide credentials when sending emails.
Password is the password used to authenticate the user.
SSL/TLS will enable the encryption for this connection.
From address is the sender address used.
Subject prefix is an optional text that can be used as a prefix in the email subject.
Reminder interval is the number of minutes between each lost connection reminder email sent.

Example for Office365 user:
SMTP Host: smtp.office365.com
SMTP Port: 587
Username: some-email@dbwatch.com
Password: SomethingSecret
SSL/TLS: enabled

If you have a provider that uses individual app passwords, that will just replace the regular password used for that user.

The second section consists of a set of “Rules”.

Rulename is your name for this rule, if you make this somewhat descriptive its easier to see what you intended next time you go into this configuration.

Type can be MAIL or SMS. The difference is that SMS will cut down the message, as to fit it better on a SMS and is intended for email-to-sms gateways.

Receivers are the email addresses this message will be sent to. It can be multiple addresses with a comma separating the emails.

Schedules is a filter to only send statuses that occur in a given timeframe. If the status occur outside the timeframe it will not send a message.

Statuses is a filter to send all or just a selection of statuses. Valid statuses are “ALL” for all statuses, “LICENSE-EXPIRING” for alerting on dbWatch license that are expiring, “EXCEPTION” for procedure exceptions, errors in the monitoring jobs. “ENGINE-WARNING” is warnings from the monitoring jobs. “ENGINE-ALARM” is alarms from the jobs. “RECONNECTION-FAILURE” is a signal for when we cant connect again to an instance, after waiting and trying multiple times. (You can configure this in Server->Server Configuration->Reconnection configuration). “ENGINE-LOST-CONNECTION” is the signal we send when we lose connection to an instance. If you have many problems with network or servers restarting and you don’t want emails on this, its best to use the “RECONNECTION-FAILURE” signal, which is slower, as it will try several connection attempts before giving up, but provide less alarms. “LOST-CONNECTION-REMINDER” will send repeated reminder emails if an instance is still in the LOST CONNECTION state after a specified time interval.

Instance filter is a FDL filter to select what instances to send signals from. Typical filtering is:
group=‘Production’
to get signals from only instances in the group “Production” or
name=‘Server A’
to get signals from only an instance with the name “Server A”

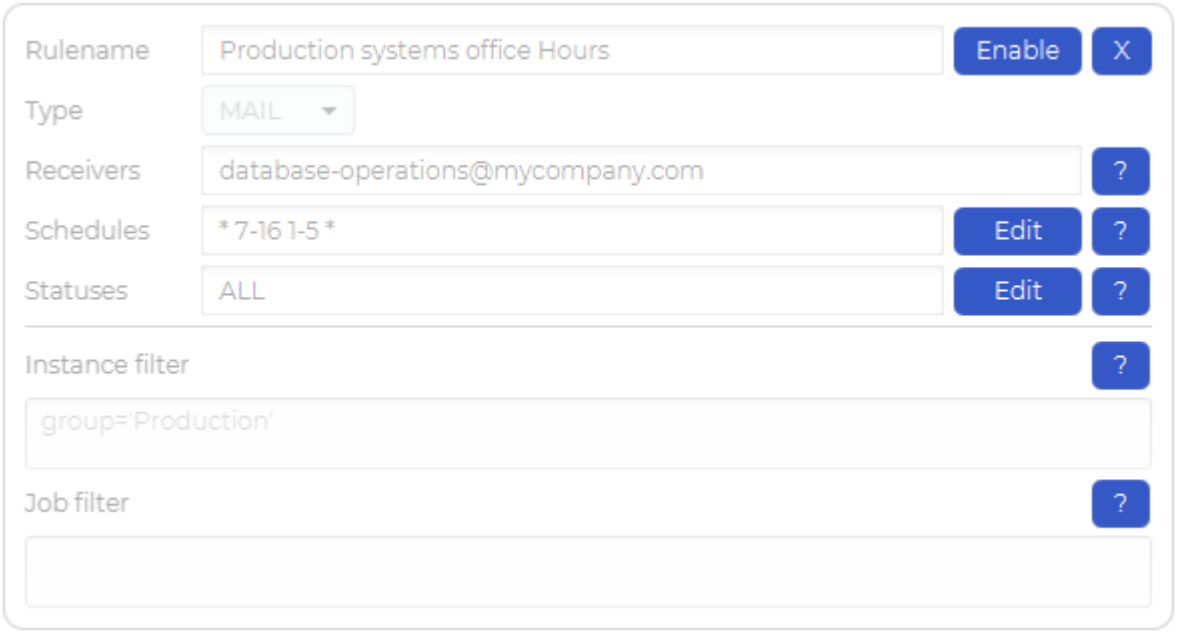
More information about FDL is here

Job filter is also a FDL filter to select what jobs you want signals from. Typical filtering is:
name like ‘backup’
to get signals from only jobs that have “backup” as part of their name.

Template rules

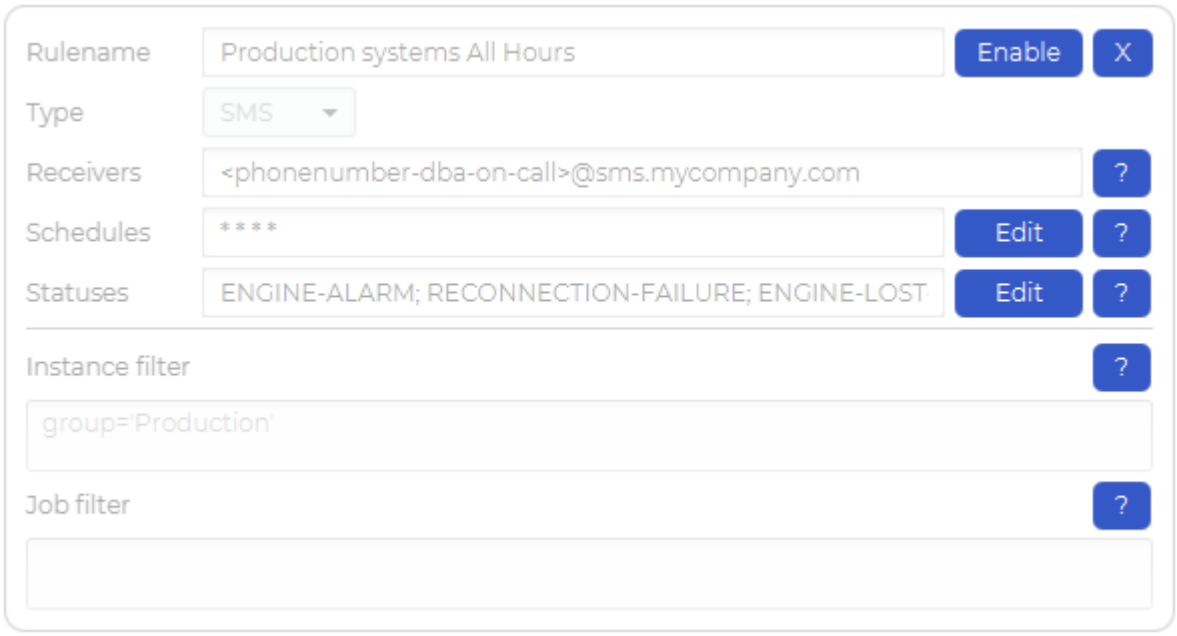
Initially there are 5 template rules created. They are disabled by default, but provide example setups that can be adjusted.

Rule 1, “Production systems office Hours”



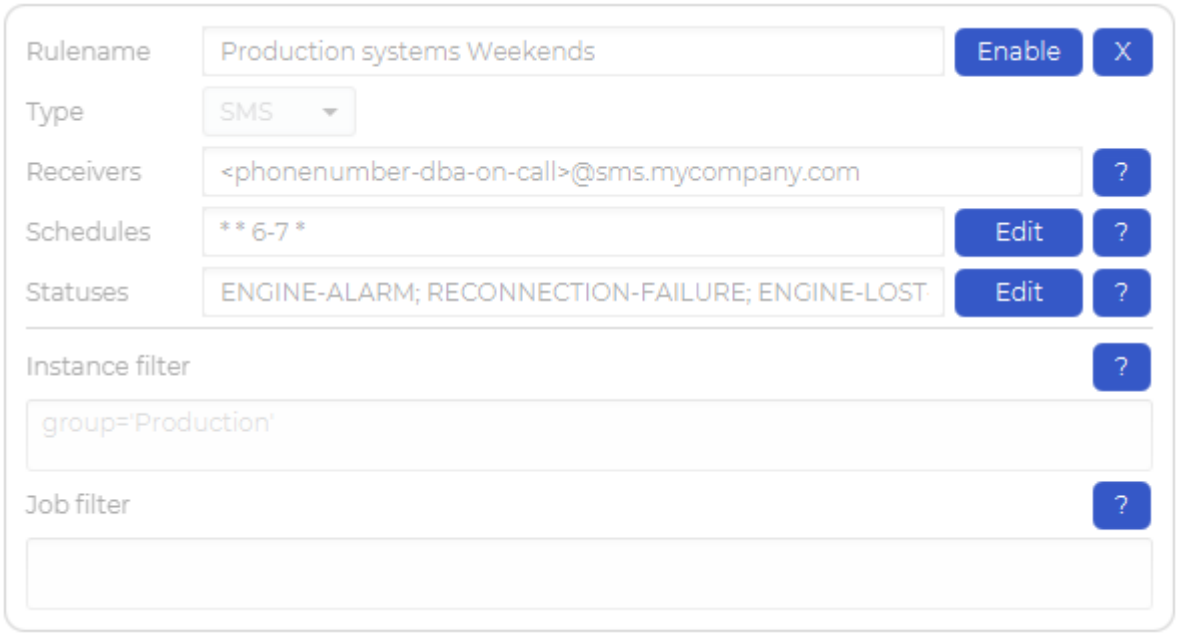
This rule is intended to send all status types from instances that is part of the group “Production”, but only within office hours. (07.00 to 16.59, Monday->Friday). Typically this would be emails that go to the DBA or helpdesk within office hours.

Rule 2, “Production systems All Hours”



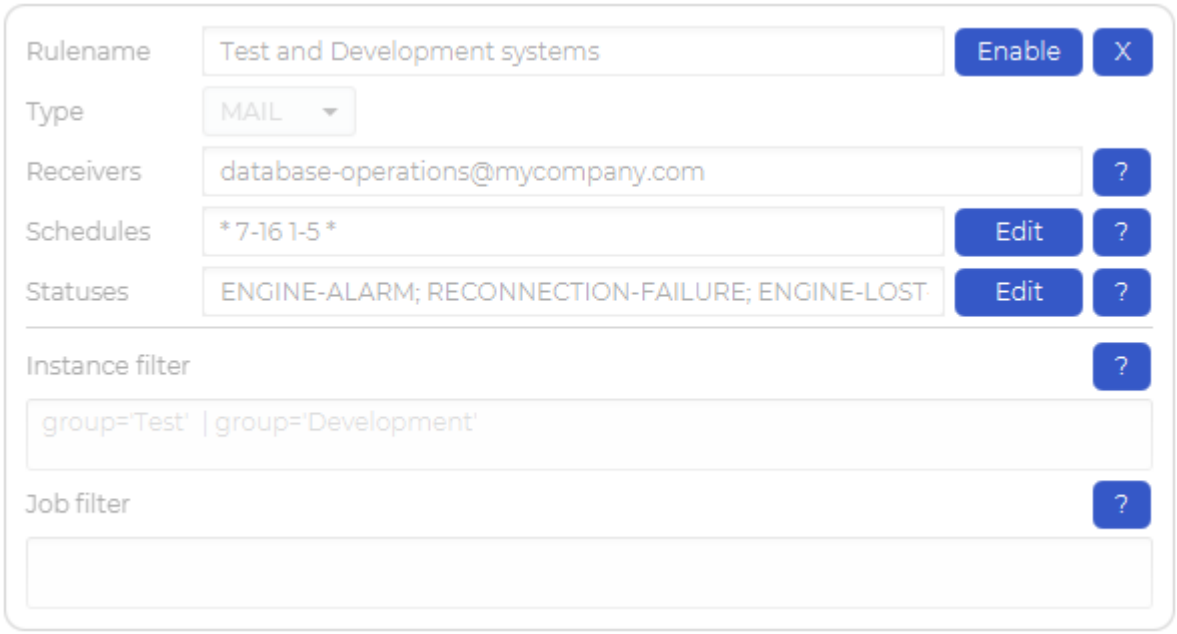
This rule is intended for a EMAIL-TO-SMS gateway, as it will cut down the message to fit a SMS. It only send ALARMS, RECONNECTION-FAILURE and ENGINE-LOST-CONNECTION signals for database instances in the group “Production”. Typically this is configured to send important alarms for production database instances directly as SMS to the DBA or on-call staff.

Rule 3, “Production systems Weekends”



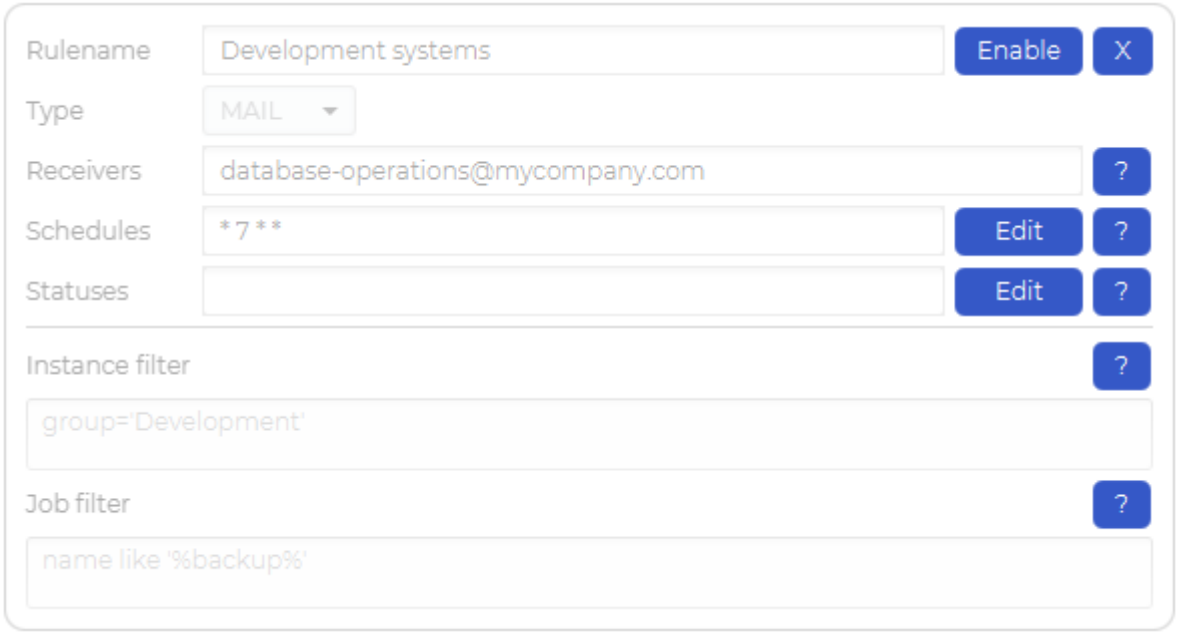
This rule is intended for a EMAIL-TO-SMS gateway, as it will cut down the message to fit a SMS. It only send ALARMS, RECONNECTION-FAILURE and ENGINE-LOST-CONNECTION signals for database instances in the group “Production”. It is configured only to send emails in weekends, so Saturday or Sunday. Typically this is configured to send important alarms for production database instances directly as SMS to the on call DBA during the weekend.

Rule 4, “Test and Development systems”



This rule only send ALARMS, RECONNECTION-FAILURE and ENGINE-LOST-CONNECTION signals for database instances in the groups “Test” or “Development”: It is configured to send emails within office hours. (07.00 to 16.59, Monday->Friday). Typically this is intended for the Dev or Test teams, helpdesk or DBA that works on dev or test systems.

Rule 5, “Development systems”



This rule sends all statuses, from the database instances in the group “Development”, if the job is has something with “backup” in its name. But only from 07.00 to 07.59. This could be used to send only backup related emails to a dedicated person for this. Note that the job (with backup in its name) will need to be scheduled between 07.00 and 07.59 to get any emails from this.

You can remove the rules you don’t need, or disable them. Rules must be enabled and the extension also needs to be enabled for messages to be sent.

Setting mail formats

To override the default mail format, you can place a set of format files in the “C:\ProgramData\dbWatchControlCenter\config\extensions\” catalog.

For mails related to alerts the files are:

- alert_format_subject.txt
- alert_format_body.txt

For mails related to lost connection the files are:

- lost_connection_format_subject.txt
- lost_connection_format_body.txt

For mails related to reconnection the files are:

- reconnect_format_subject.txt
- reconnect_format_body.txt

The files called *_subject.txt are used to generate the mail subject field, and the files called *_body.txt are used to generate the mail body.

There is a set of values that are initiated when the email is generated. These are:

- ALERT – The name of the alert that generated the event.
- STATUS – The status for the alert.
- INSTANCE – The name of the database instance for the alert.
- ALERT_DETAILS – A description of the alert state.
- GROUP – The group for the instance.

For the “ENGINE-LOST-CONNECTION”, “RECONNECTION-FAILURE” and “LICENSE-EXPIRING” type signals, STATUS, ALERT, and ALERT_DETAILS values are not generated, as it is not applicable to this signal.

So a typical format for an alert would be:

alert_format_subject.txt

```
%STATUS% for %ALERT% on %INSTANCE%
```

alert_format_body.txt

```
%STATUS% for %ALERT% on %INSTANCE% for %GROUP%

%ALERT_DETAILS%
```

```
This mail was generated automatically by the dbWatch Server.
```

This will then produce an email similar to the following;

Subject:

```
WARNING for Database disk space usage on SomeDatabaseInstance
```

Body:

```
WARNING for Database disk space usage on SomeDatabaseInstance for Acme Inc

Drive C: 396.7 GB free space. 39 database(s) effected

This mail was generated automatically by the dbWatch Server
```

Format pr. rule

In addition to the general rule format files, it is possible to specify different formats per rule.

You do this by creating additional files with the rule name in square brackets at the end of the file name.

So if you have a rule called “A”, you can create files called “alert_format_subject[A].txt and “alert_format_body[A].txt”. Then theese files will be used when creating E-mails for this rule.

Remove the “OK” messages

It is possible to stop the Mail extension from sending out the “OK” messages. However there is currently no GUI for this, so you have to edit the configuration file manually.

The configuration file can be found in the “C:\ProgramData\dbWatchControlCenter\config\extensions” catalog and is called “E-Mail extension.xml”.

Open the file in a text editor. There should be a “filter” tag for each of the filters that are defined. Here you can add a tag called “report-improvement” with the value false. This will cause the extension to skip the OK messages for this filter.

You have to restart the dbWatch Server in order for the changes to take effect.

Debugging

If you are experiencing issues when using the extension, it is possible to enable more logging by setting a tag in the configuration file manually.

The configuration file can be found in the “C:\ProgramData\dbWatchControlCenter\config\extensions” catalog and is called “E-Mail extension.xml”.

Open the file in a text editor, there should be a tag called “debug-level” with the value 0. You can change this to 1 (some logging) or 2 (a lot of logging). Doing this makes it easier to track down configuration errors as the server.log file will tell you what is going on when it is attempting to send messages. You have to restart the dbWatch Server in order for the changes to take effect.

Once this is enabled, and the dbWatch Server is restarted, when you test the email extension, you will get more information in the server.log file. (Default location C:\ProgramData\dbWatchControlCenter\log\server.log)

<< Extensions and 3rd party integrations / Configuring email extension for Service Now >>

Configuring email extension for Service Now

Integration with Servicenow

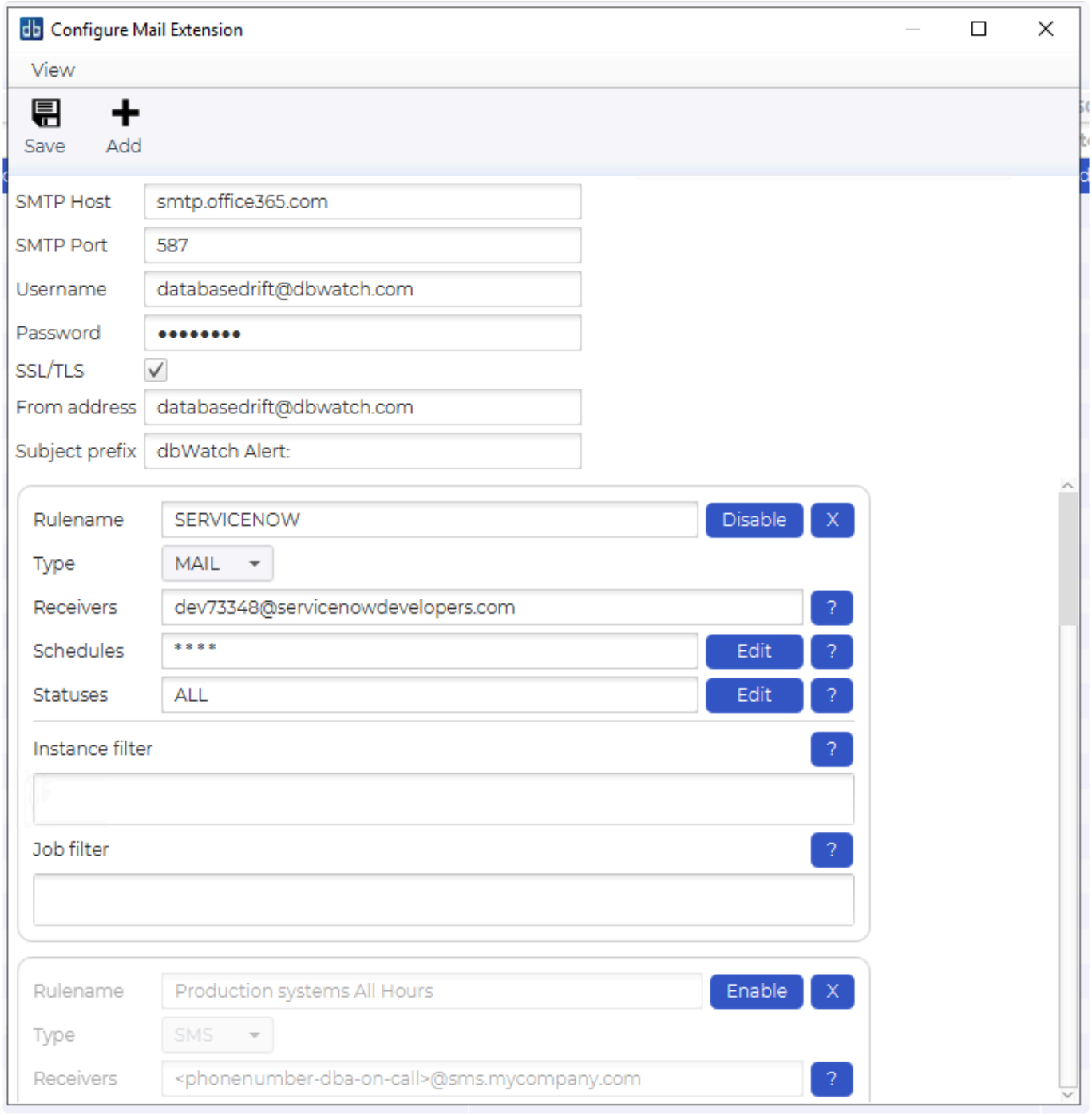
In this example we will configure dbWatch and Servicenow so that alarms from dbWatch, can create incidents in Servicenow.

To integrate the alerts from dbWatch with Servicenow incident creation, there are two parts that needs to be setup.

Configure a specific rule for Servicenow

Before we start we assume that the email extension is configured and is tested, and you are able to receive email alerts in your own inbox. There is an alert included in dbWatch, called “Test alert” that is useful for testing the alerting capabilities as the type of and frequency of alerts can be adjusted.

In order for the scripting language of Servicenow to interact with the emails in the best possible way, we need to setup the email extension with a specific rule for the Servicenow integration.
This is an example of such a setup. Its important that this rule is given a name, such as “SERVICENOW” (without the “) , that also corresponds with some formatting file names we will create later. In all the examples here we use the string “SERVICENOW”. This is also case sensitive.



What is important is that the rule name is set to “SERVICENOW” and that the receivers is set to the incoming email address of your Servicenow environment. You can apply filters if you dont want some groups, platforms etc creating incidents in Servicenow.

In addition to this, we need to setup rule specific formatting files.

They must be placed in the edited_configurations directory, example placement based on default install is: C:\Program Files\dbWatch\12.8.2\server\extensions\edited_configurations\

The files:
alert_format_subject[SERVICENOW].txt
lost_connection_format_subject[SERVICENOW].txt
reconnect_format_subject[SERVICENOW].txt

Needs to have this line of content (in bold):
*%STATUS% for ALERT on INSTANCE *

The files:
alert_format_body[SERVICENOW].txt
lost_connection_format_body[SERVICENOW].txt
reconnect_format_body[SERVICENOW].txt

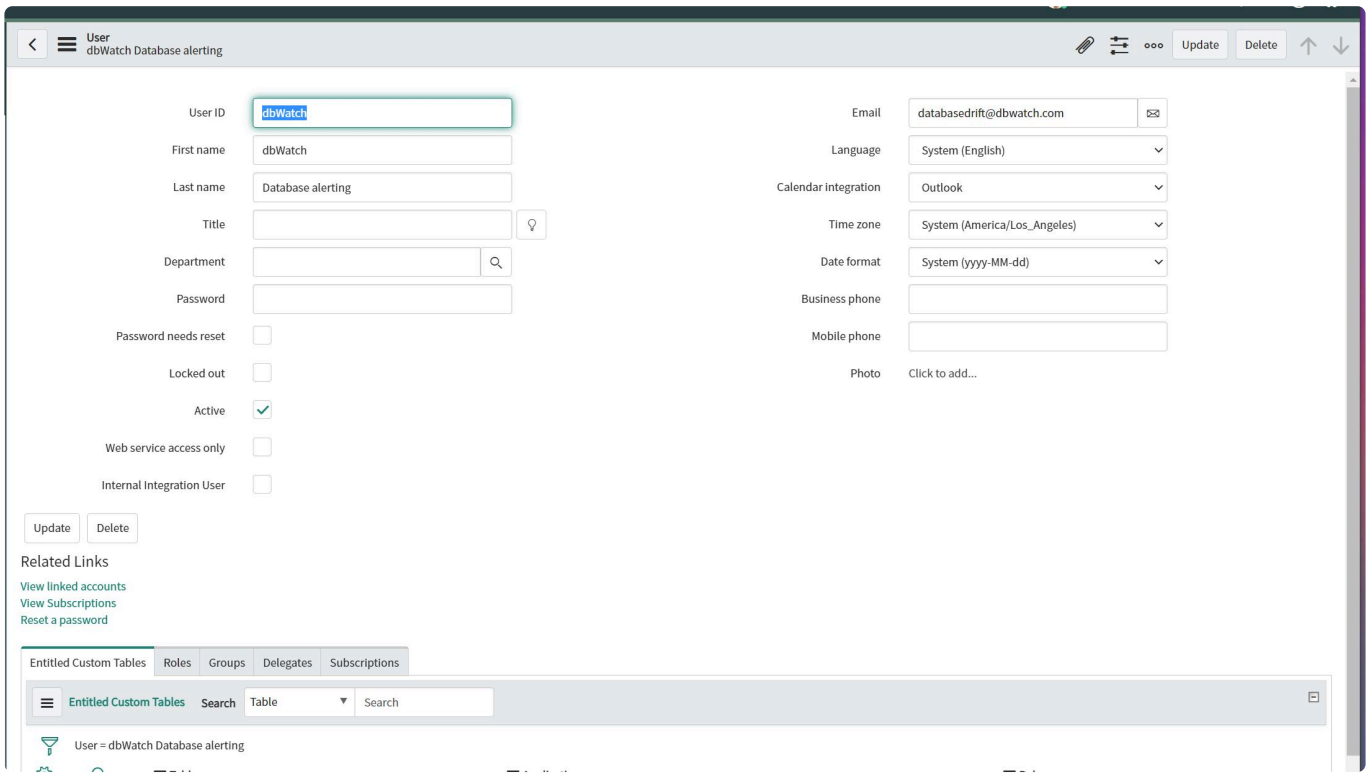
Needs to have the following content (in bold):

*STATUS:STATUS
ALERTNAME:ALERT
INSTANCE_NAME:INSTANCE
GROUP:GROUP
DETAILS:%ALERT_DETAILS%*

This formats the emails so its easy to pick up the information in the scripting language in Servicenow.

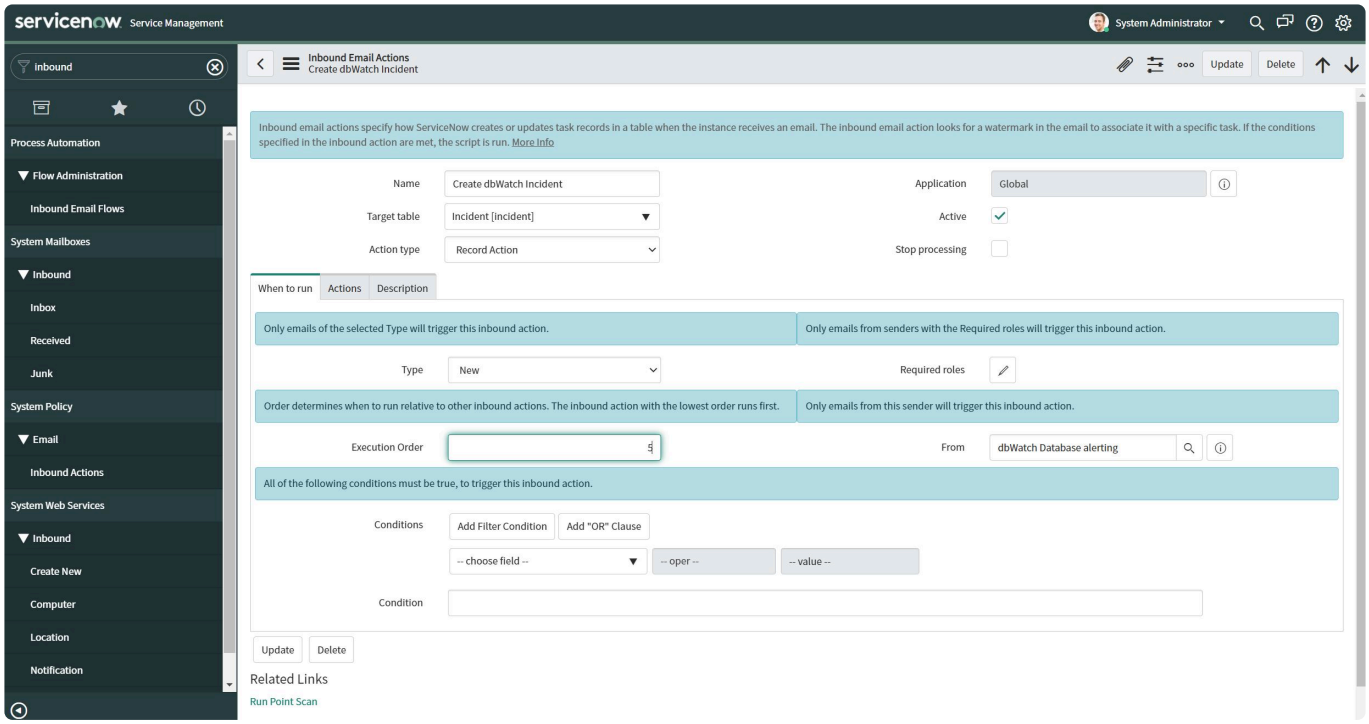
Setup in Servicenow

Create a service user that can be used to map the caller:



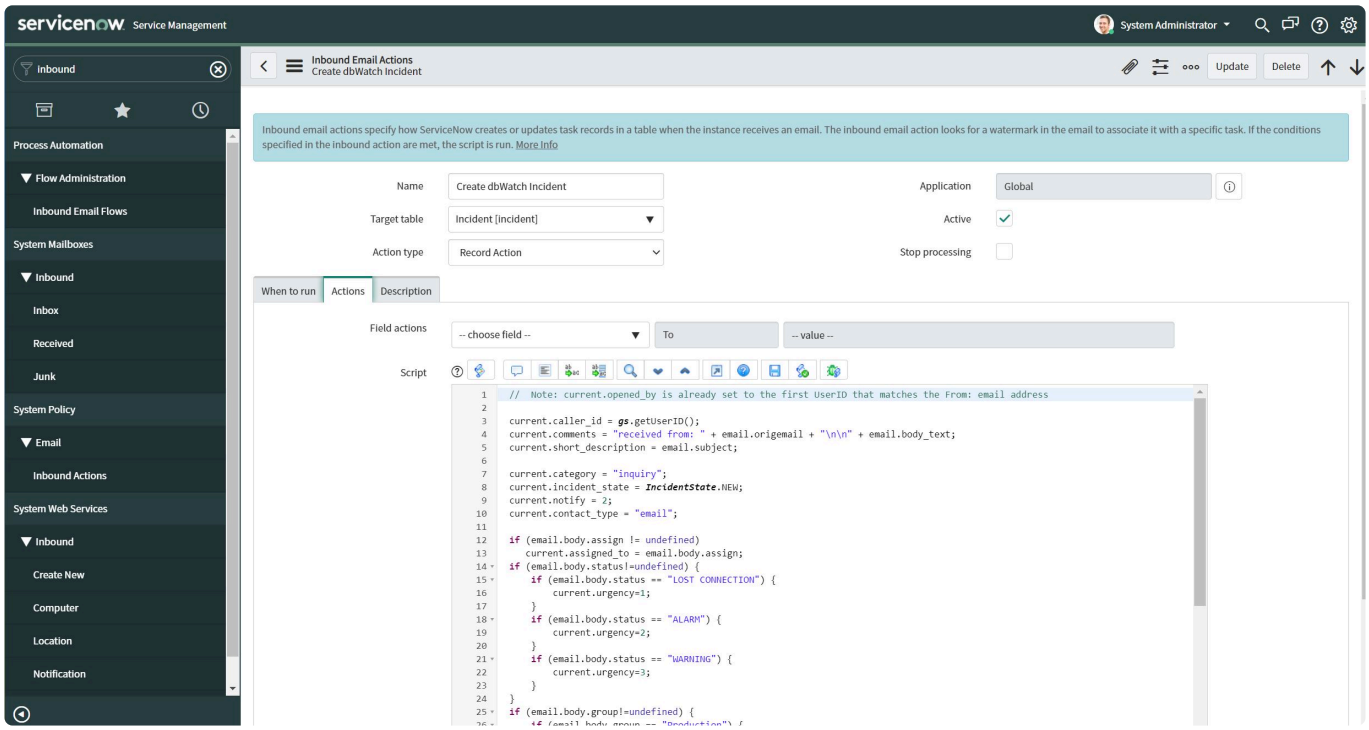
The email address of this user should map to the email used to send dbWatch alerts, as you configured when you configured the email extension.

Then create a new inbound email action for dbWatch emails:



Note that it will only run on emails from the user we configured earlier. It also need to have an execution order lower than the common inbound email action for all emails. The name should also reflect that it is processing dbWatch alerts.

Then go to actions and insert the script for processing the data:



The updated file for formatting for email extension, and the last script for inbound email action processing is available from Github.

[Github link]

Web server

Control Center can be configured to serve dynamic web content over http.

This is intended for dashboards or live reports, and for data export used for integration purposes.

Webserver configuration

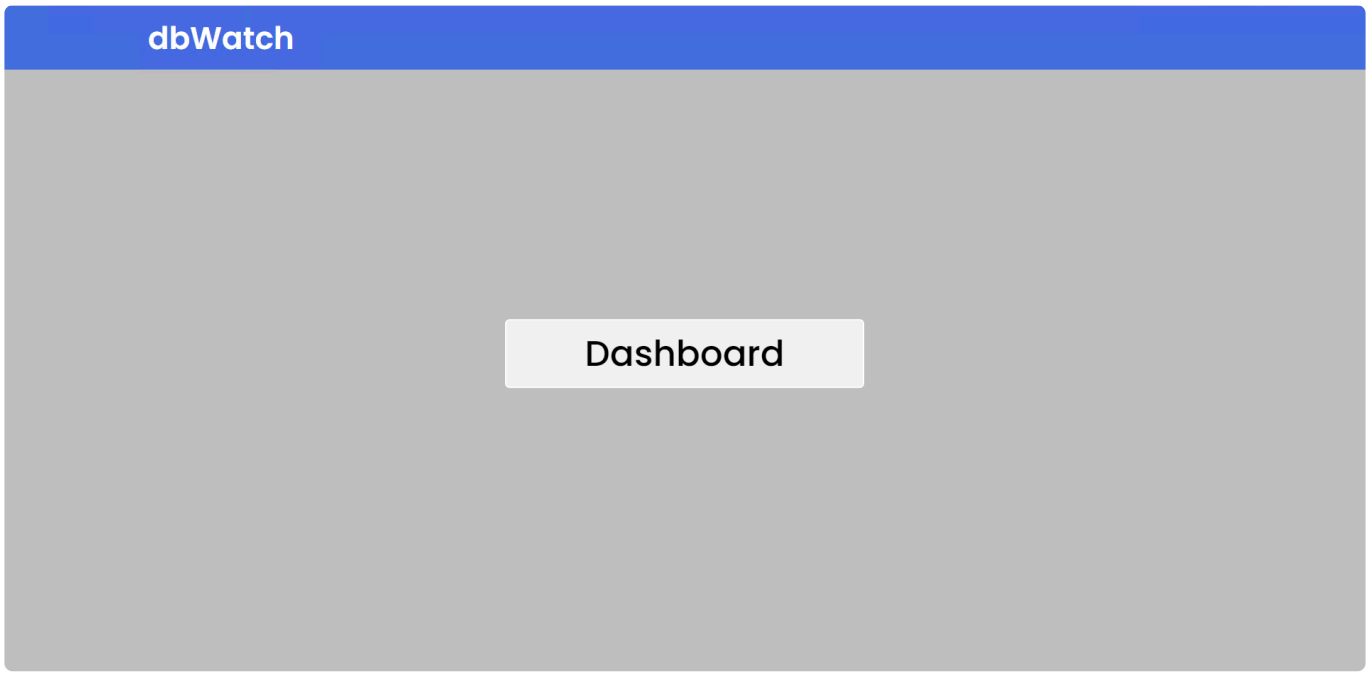
- Webserver setup
- Configuring anonymous webaccess

Mountable web pages

The following pages can be mounted on an url in dbWatch.

- Dashboard
- Data export

When you go to the root URL, there is welcome screen. Click on dashboard to see the dashboards available. It looks a bit empty and alone at the moment, but this is because new functionality is coming on this front page.



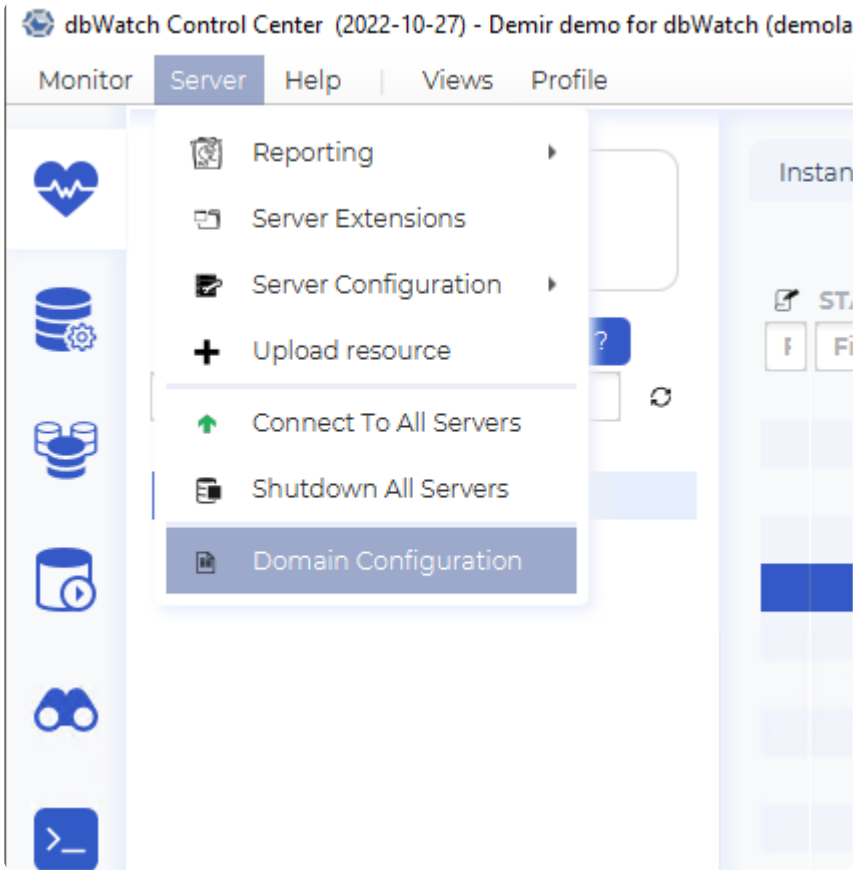
Supported browsers

Supported browsers

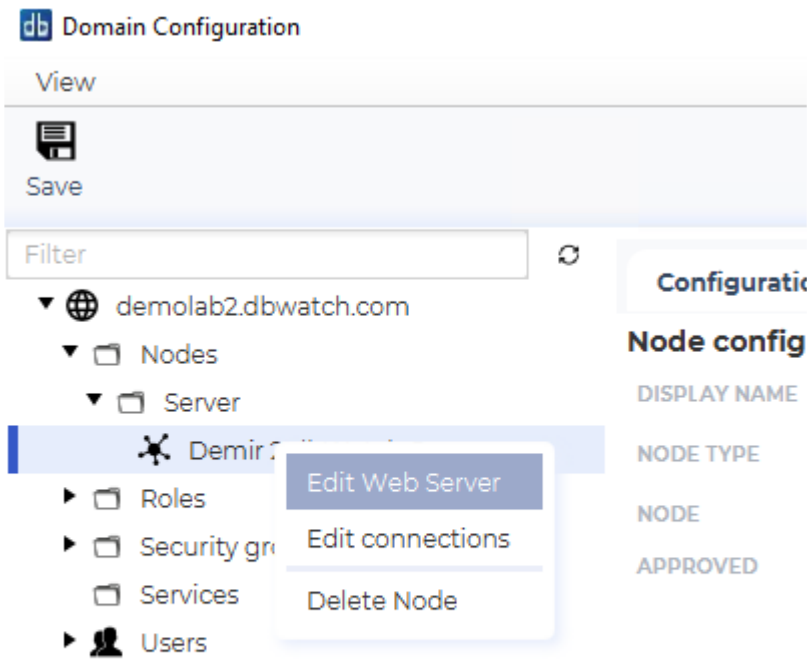
<< Configuring email extension for Service Now / Web configuration >>

Web configuration

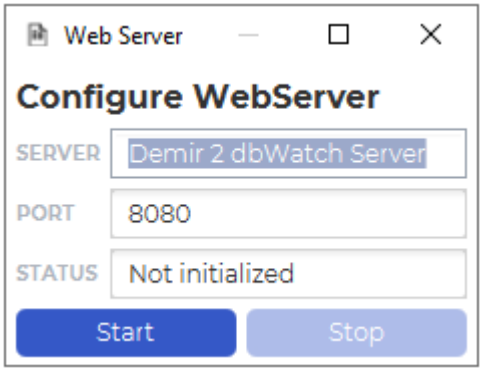
To enable the Web server, go to the “Server” -> “Domain Configuration” menu.



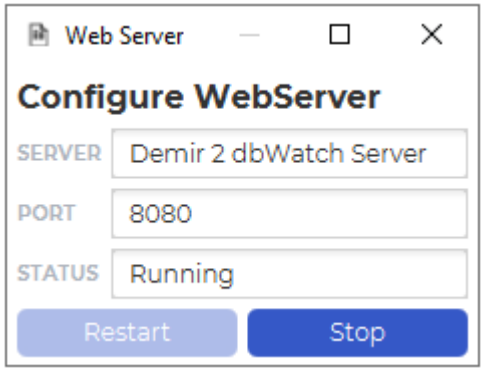
Expand the domain, Nodes, Server, and right-click on the server you want to run the webserver on. Select “Edit webserver”



In the Configure webserver dialog, it will indicate if its already running or not. You can also change the port. Click start to run.



When you have started it, it should look like this:



The example will create a web server on **http://localhost:8080**

Web pages

The following pages can be mounted on an url in dbWatch.

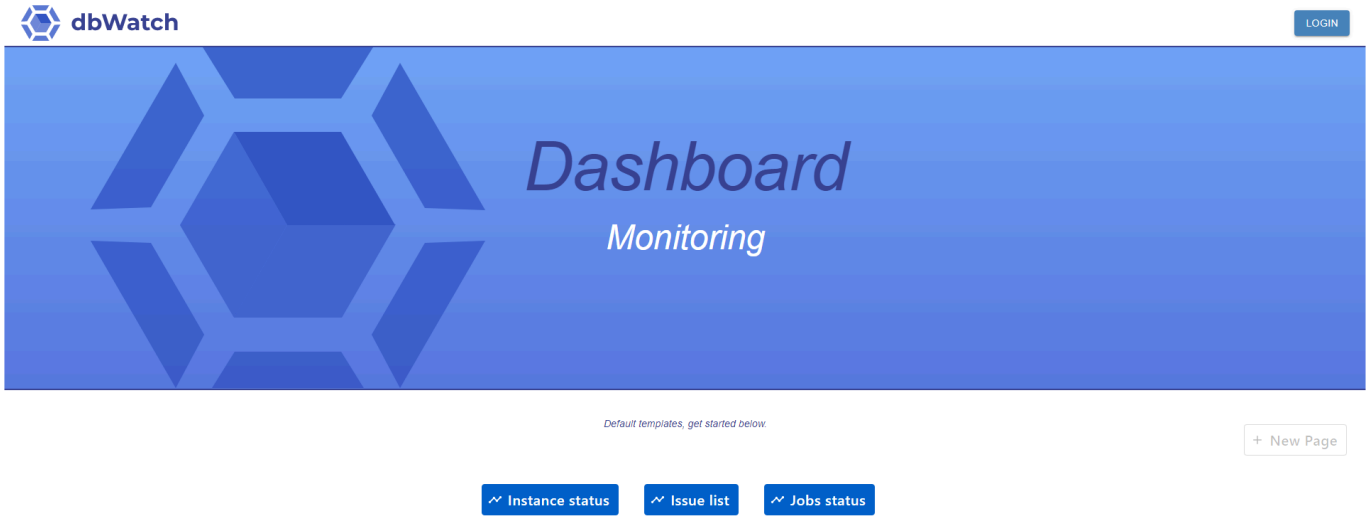
- Dashboard
- Data export

<< Web server / Dashboards >>

Dashboards

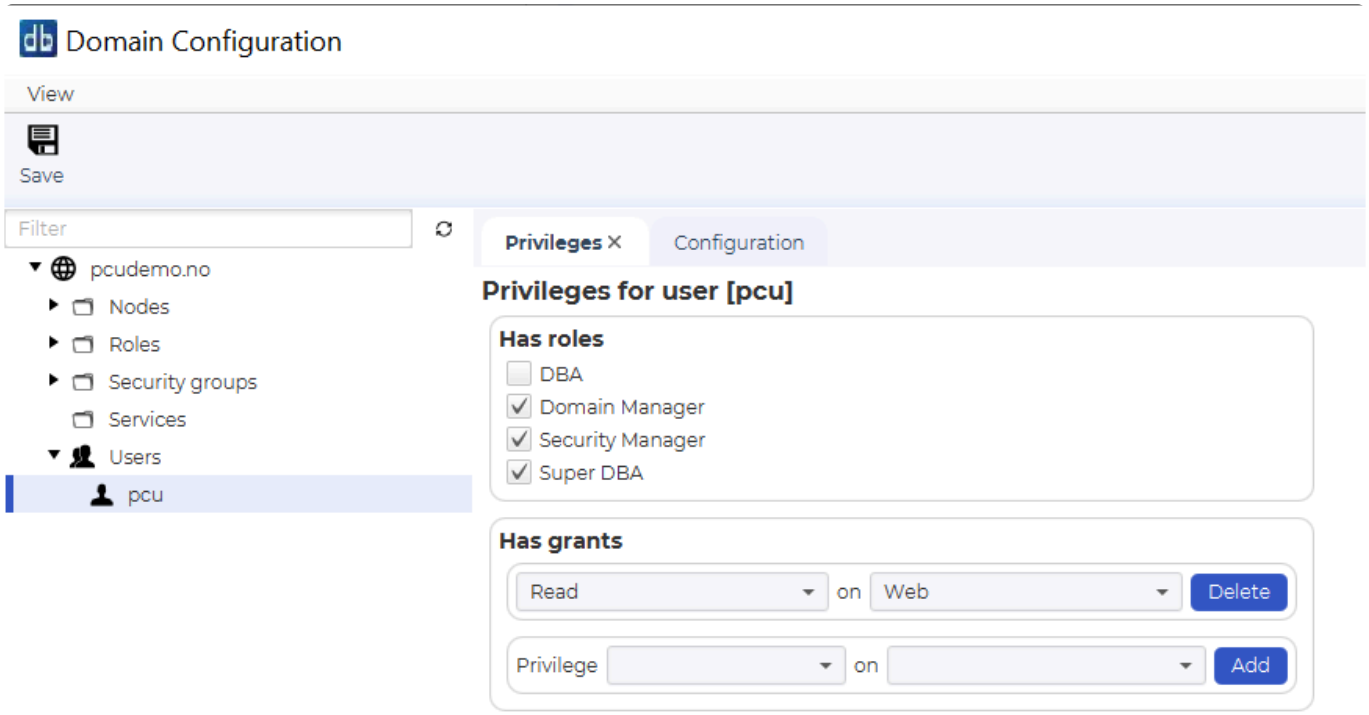
Dashboard overview

The main page for the dashboards:



User and login

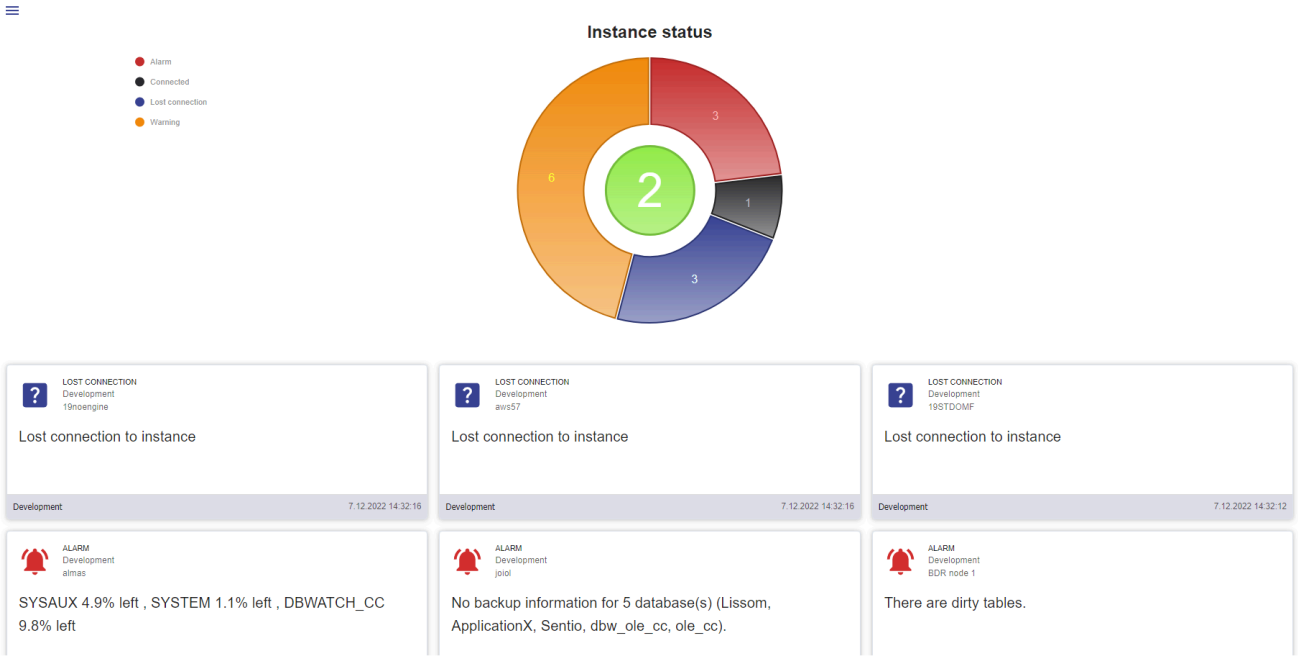
To view dashboards, you need to login with a user that has at least read privileges on web.



Default dashboards

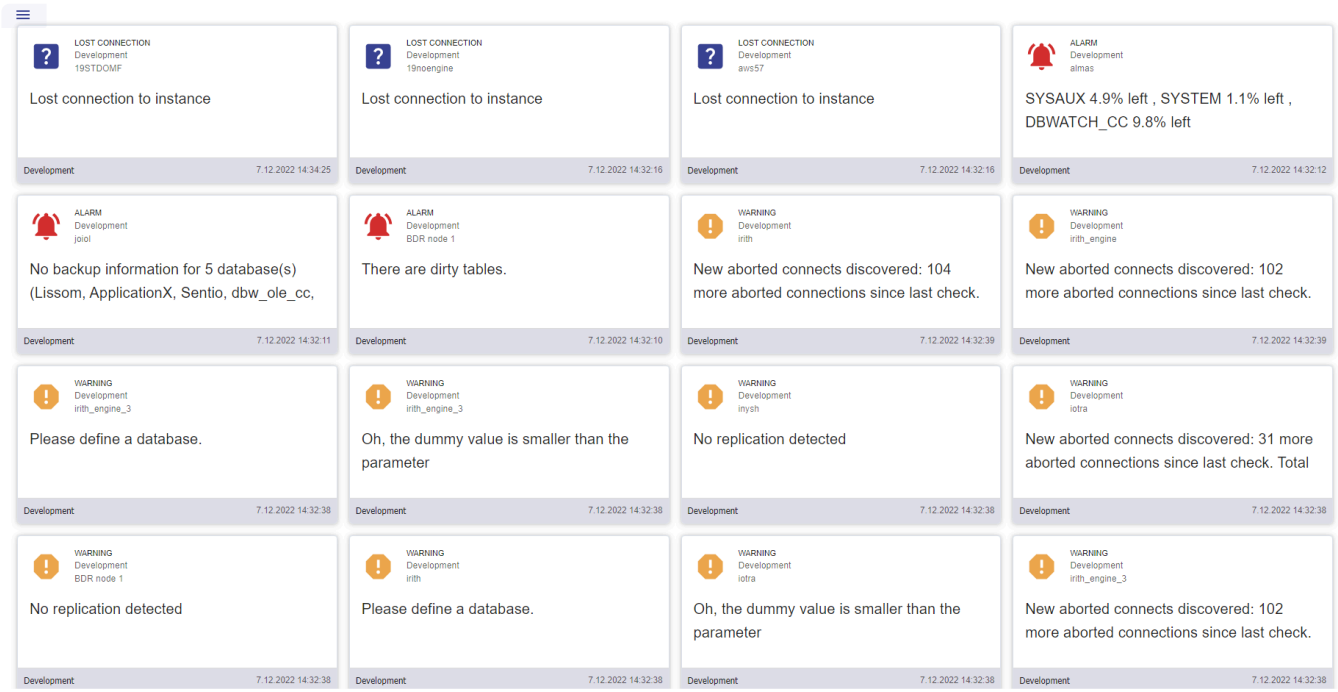
These are the default dashboards:

Instance status:



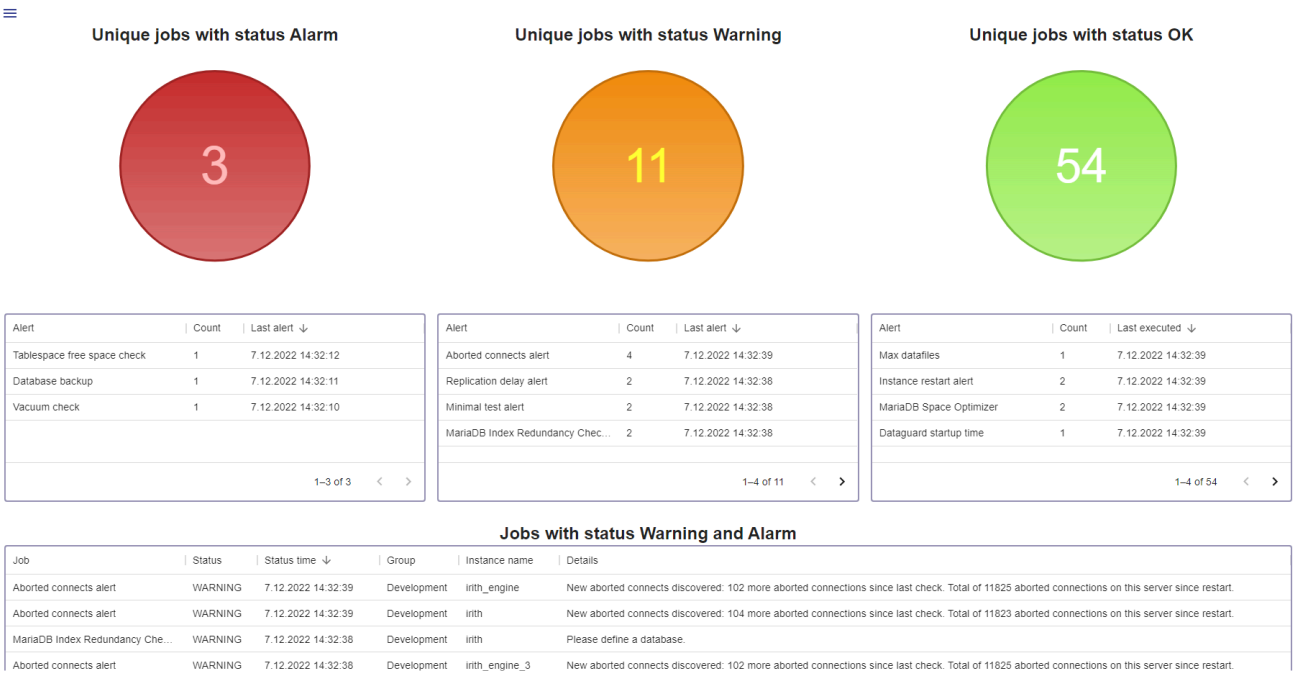
This dashboard visualizes the number of instances in different status, and also an issue list from all instances on this server.

Issue list:



This dashboard visualizes the list of current issues (alarms or warnings from jobs) on all instances on this server.

Job status:



Jobs with status Warning and Alarm

Job	Status	Status time	Group	Instance name	Details
Aborted connects alert	WARNING	7.12.2022 14:32:39	Development	irith_engine	New aborted connects discovered: 102 more aborted connections since last check. Total of 11825 aborted connections on this server since restart.
Aborted connects alert	WARNING	7.12.2022 14:32:39	Development	irith	New aborted connects discovered: 104 more aborted connections since last check. Total of 11823 aborted connections on this server since restart.
MariaDB Index Redundancy Che...	WARNING	7.12.2022 14:32:38	Development	irith	Please define a database.
Aborted connects alert	WARNING	7.12.2022 14:32:38	Development	irith_engine_3	New aborted connects discovered: 102 more aborted connections since last check. Total of 11825 aborted connections on this server since restart.

This dashboard visualizes the number of unique jobs in different status on this server.

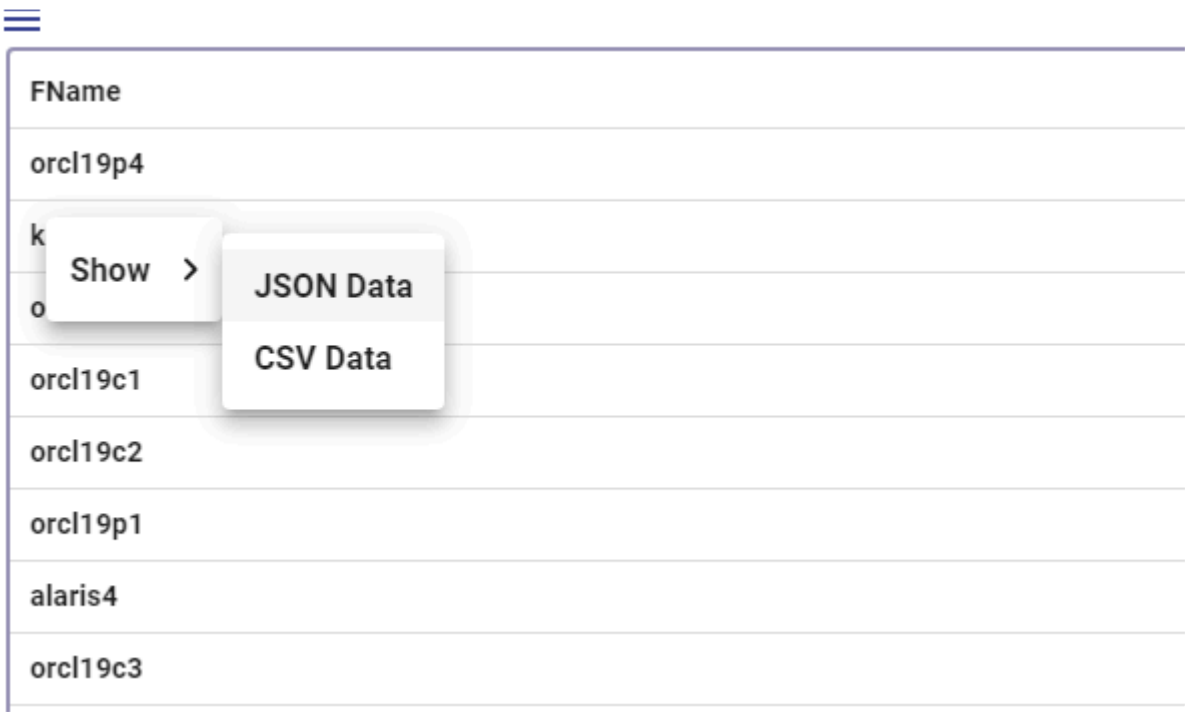
<< Web configuration / Web data export >>

Web data export

You can export data from dbWatch through http for easy access by others (tools or humans). Data can be made available formatted as json or csv.

Data export might require anonymous webaccess configured.

Any data formatted as a table in the dashboard will support data export, just right-click on the table, select show and the format you want.



If you select JSON Data you get something like this:

```
{
    "title":"",
    "query":"instance->i/$i/name{}",
    "tags":[[]],
    "columns":["FName"],
    "data":[
        ["kethor"],
        ["orcl19c2"],
        ["orcl19p1"],
        ["NODE1_AD1"],
        ["lodos"],
        ["orcl19p2"],
        ["NODE2_AD1"],
        ["NODE3_AD1"],
        ["alaris3"],
        ["orcl19p3"],
        ["alaris2"],
        ["orcl19c4"],
        ["culain"],
        ["alaris1"],
        ["gerrard"],
        ["korriban"],
        ["antar"],
        ["jante"],
        ["onderon"],
        ["orcl19p4"],
        ["kohlma"],
        ["alaris4"],
        ["orcl19c3"],
        ["orcl19c1"]
    ]
}
```


If you select CSV Data you get something like this:

The URL generated, can then be used to integrate the data, and each time the URL is hit, fresh data is produced backed by the FDL query you were running.

There is a possibility to create custom formats in addition to the csv and json format provided through the graphical interface.

Additional formats and custom formats can be made available on request.

<< Dashboards / Web dashboard editor (beta) >>

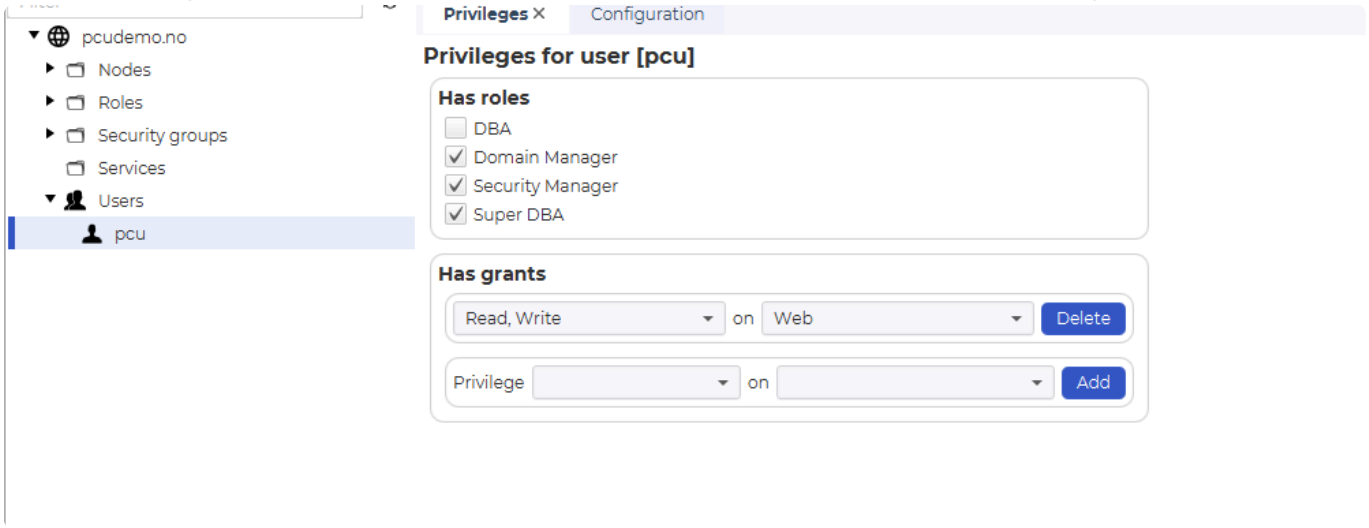
Web dashboard editor (beta)

Web dashboard editor (currently in beta)

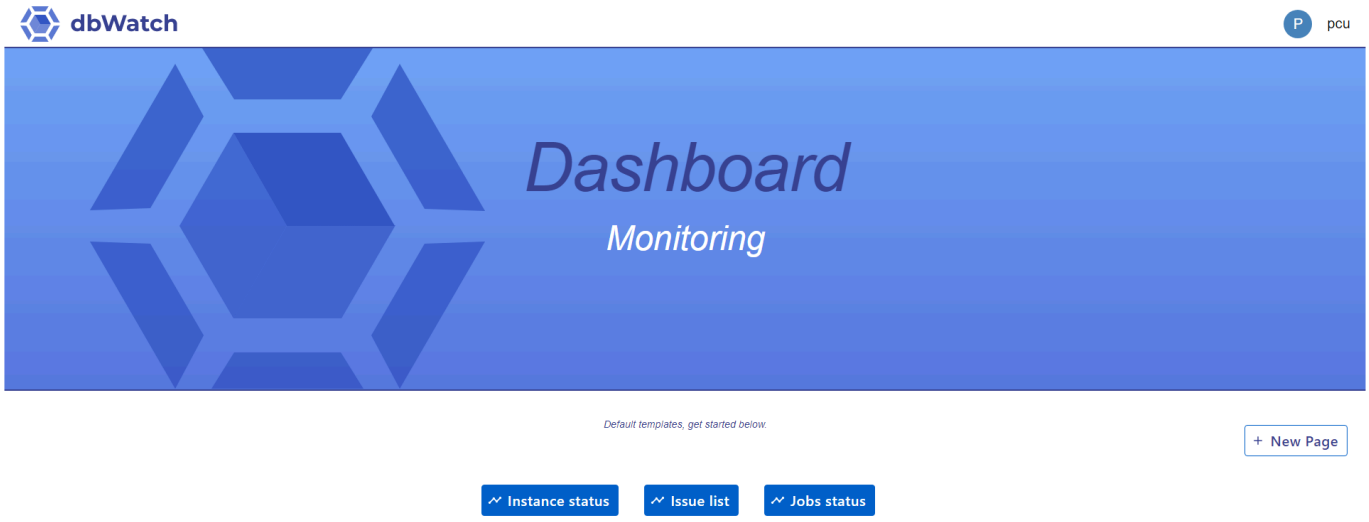
Web dashboard editor is an additional addon module to dbWatch Control Center, that allows the customer to create their own dashboards displaying data about their database environment.

While it is in beta, it is open for customers to try out.

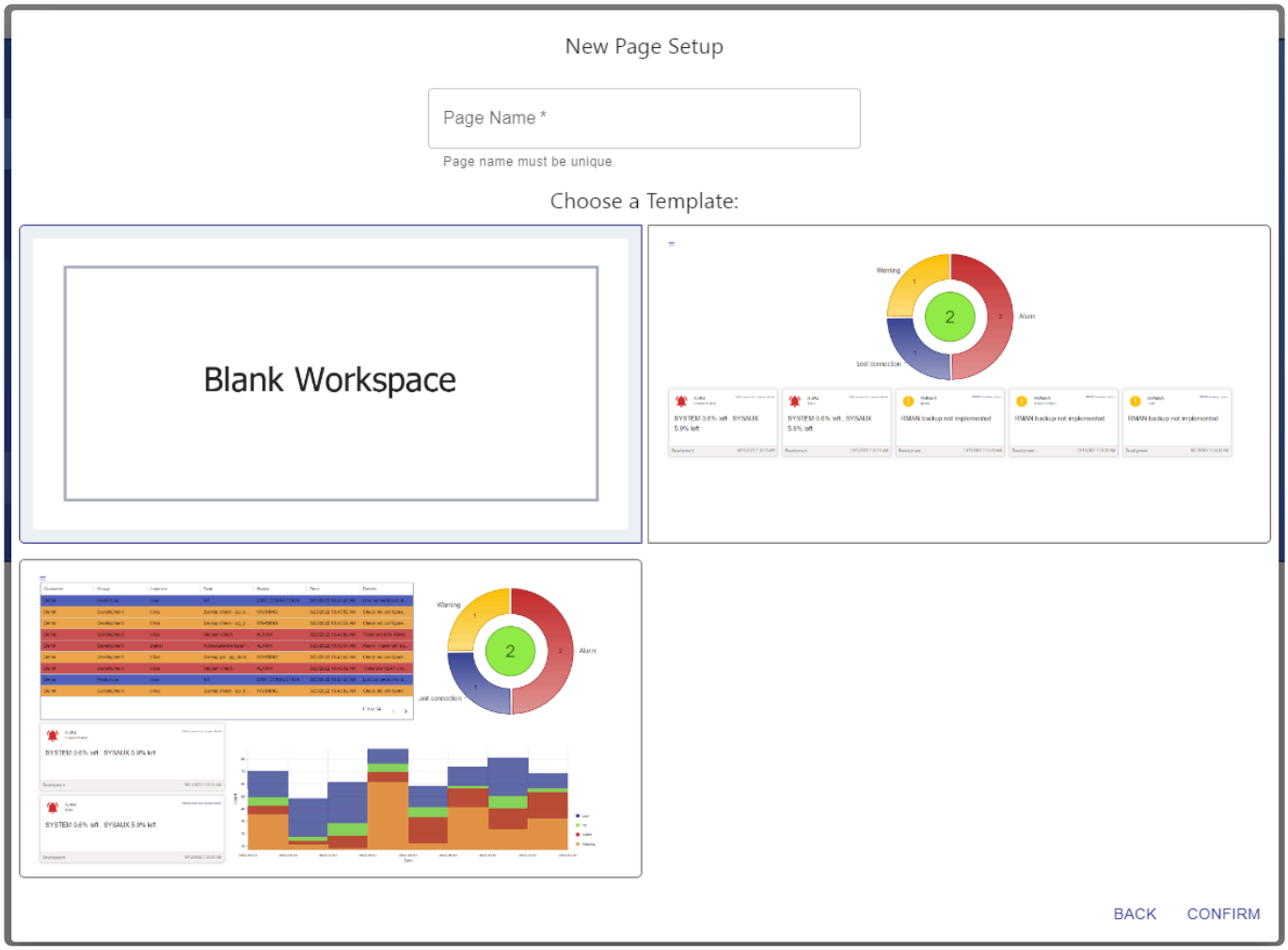
The web dashboard editor is available from the web frontend, but to use the web dashboard editor you will need to login with a dbWatch Control Center user that has read and write privileges



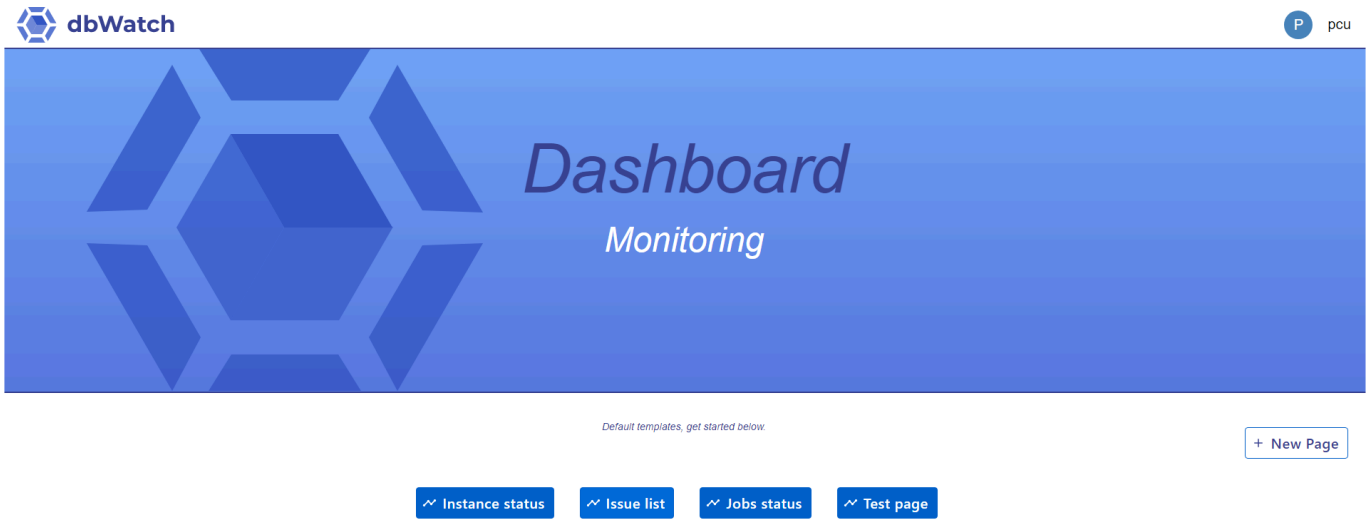
When you login with a user with the correct privileges the “+ New Page” icon will be possible to use.



If you click on “+ New Page” you open the “New Page Setup” window. Here you can name your new page and choose a template to start of with, if you want.

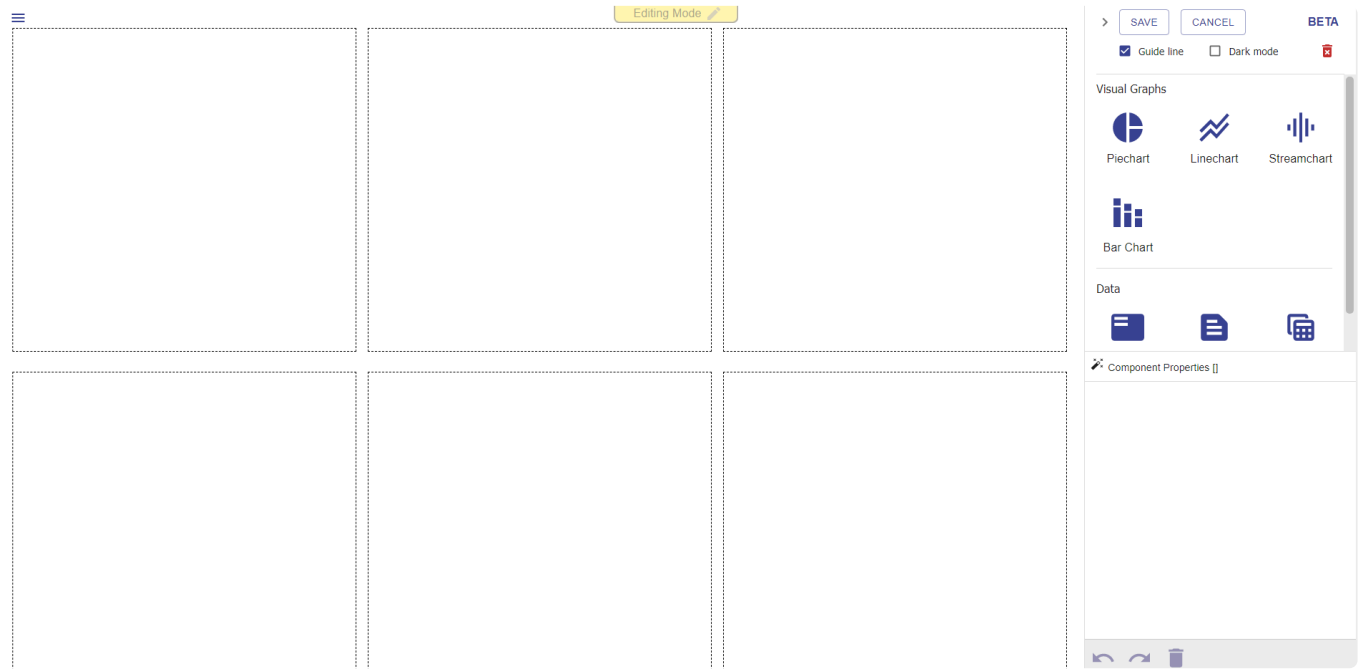


If we try to create a new page, called “Test page” it will show up in the dashboard:



If you click into the new “Test page”, you will find the editor if you move the mouse pointer to the right side and click to expand it.

When you expand it, you will be in editing mode, and you can drag and drop visual components.



Anonymous webaccess

Configuring Anonymous Web Access in dbWatch

This guide outlines the updated process for configuring anonymous web access in dbWatch. The configuration is now managed through the ‘tuning.properties’ file, which simplifies the setup process.

Step 1: Create a User with Web Access

First, create a dedicated user for anonymous web access. Ensure this user has the **read on web** privilege.

- For instructions on assigning ‘read on web’ privileges, visit How to give users read on web

Step 2: Configure the tuning.properties File

The anonymous web access settings are now located in the ‘tuning.properties’ file. To locate this file:

- Find the Web Server’s Configuration Directory:
  You can find it within the server’s config directory, as named during the web server setup.
- Check for the tuning.properties File:
  If ‘tuning.properties’ does not exist, it will be automatically generated approximately one minute after starting the server.
- Edit the File:
  Open the ‘tuning.properties’ file, which should initially appear as follows:

```
# anonymous
# anonymous.token = password:<set password>@user:<set user>@domain:domain.my
```

1. Configure the Settings:

- Uncomment the ‘anonymous.token’ line by removing the ‘#’.
- Set the Password and User: Replace and with the appropriate credentials.
- Set the Domain: Replace domain.my with your server’s governing domain.

Step 3: Apply the Configuration

After saving changes to ‘tuning.properties’, dbWatch will automatically apply the new configuration within 10 to 20 seconds.

Supported Web Browsers for dbWatch Dashboard

As of the current version, dbWatch Dashboard supports the following web browsers:

- **Google Chrome**
- **Microsoft Edge**
- **Firefox**

Why These Browsers?

Our focus on Google Chrome and Microsoft Edge is driven by several factors:

- **Advanced Features and Standards**
  These browsers adherence to web standards, ensuring compatibility with the latest web technologies used in dbWatch Dashboard.
- **Security**
  Chrome and Edge offer strong security features, which is crucial for protecting sensitive database information accessed through our dashboard.

Why Not Internet Explorer?

Internet Explorer lacks support for many modern web standards and technologies used in our dashboard. Security Risks: Internet Explorer receives limited security updates, making it more vulnerable to security threats.

<< Anonymous webaccess / Advanced Topics >>

Advanced Topics

For this section, we'll be focusing on Farm Data Language and other topics you may find interesting.

<< Supported Web Browsers for dbWatch Dashboard / FDL – Farm Data Language >>

FDL – Farm Data Language

Introduction to Farm Data Language

Farm Data Language or FDL for short is the query language used in dbWatch to find information about objects. FDL is formerly known as DBWQL.

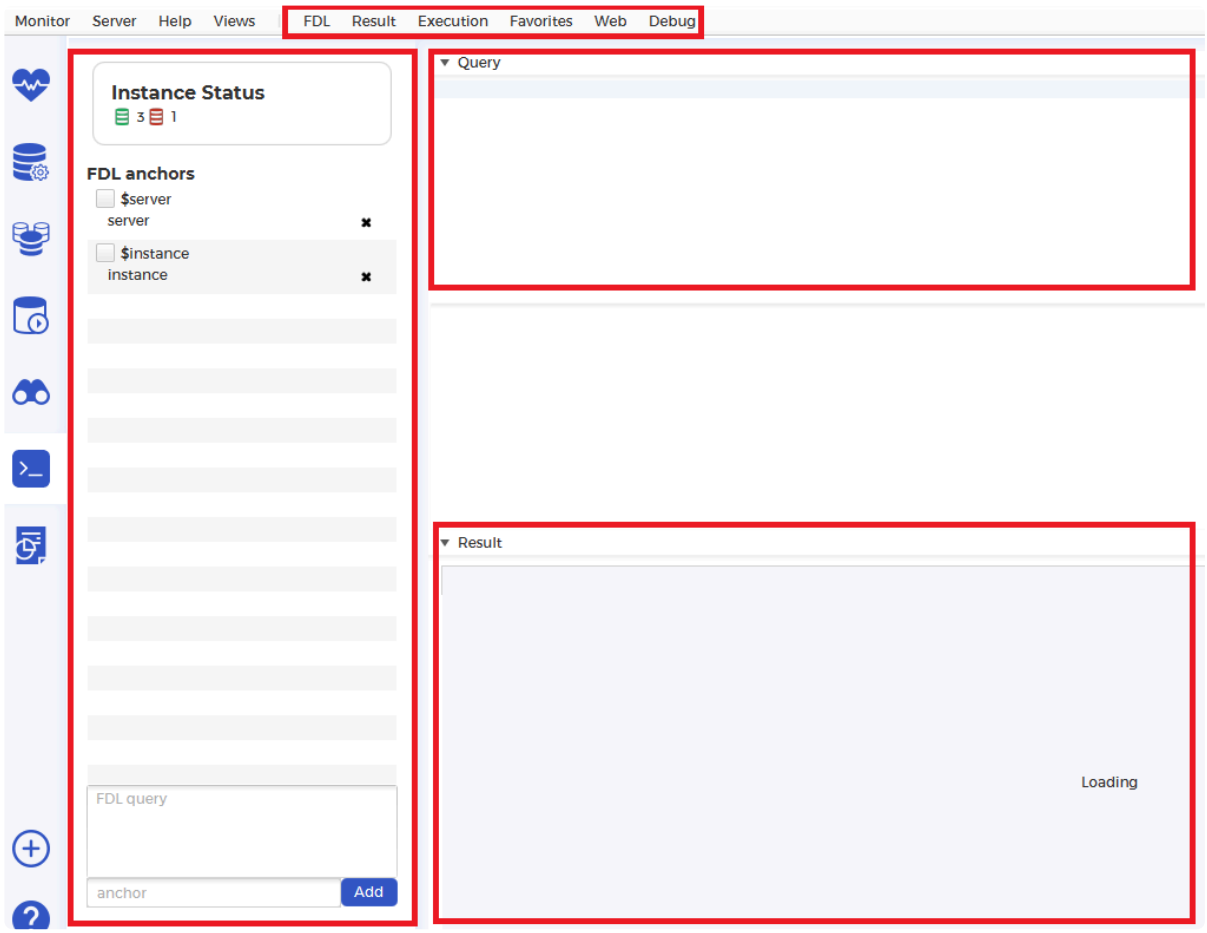
This topic should be considered as the advanced part of dbWatch. It is also the central part from which many features in dbWatch are based and should be learned by anyone who wants to get the most out of their dbWatch installation.

FDL highlights

- FDL uses path expressions to navigate the dbWatch state.
- FDL uses filter expressions to filter a set of data.
- FDL is modeled on the XPath language.
- FDL is designed to fetch data asynchronously to efficiently fetch data from multiple servers.

Introduction

The best way to get familiar with FDL is by using the FDL console, accessible as a tab on the left side of the dbWatch Monitor.



The console consists of 4 main areas and a command bar.

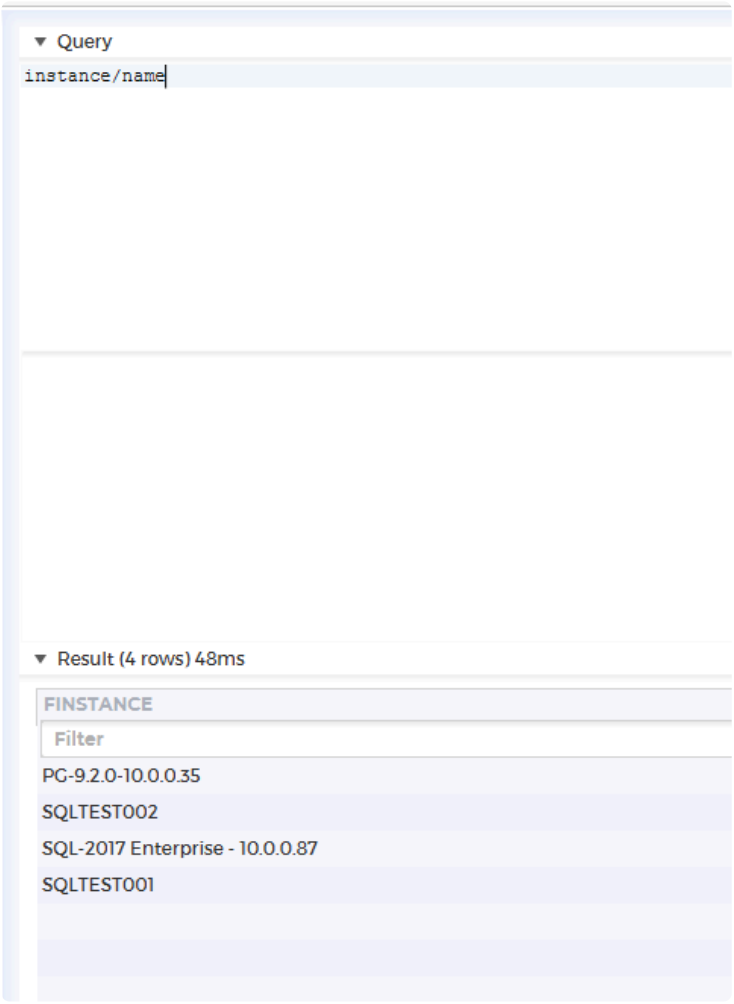
1. On the left-hand side, you have the predefined anchors to use in your queries. In Enterprise Manager, it is similar to the “Context area”.
2. The topmost part holds all your settings and additional options when executing your FDL. Here you can set the execution as either “Continuous” (automatically triggers written FDL queries) or “Triggered” (manually triggers written FDL queries). You can also open a new FDL Console, change the result page to Table format or Tree format, and debugging.
3. Below it is the “Query area”. This is where you enter your FDL queries.
4. Underneath the “Query area”, the non-boxed space is where it displays suggestions if you hit the tab key while typing queries or it will display error messages. This is commonly known as “Information Area”.
5. The bottom-most area is the “Result area’. This is where the results of FDL queries are displayed.

By default, the FDL console operates in “Continuous”, which means that queries are executed on the fly as you type. You can change this to “Triggered” on the “Execution” menu. Then queries will only be executed when you press “Ctrl + E”.

dbWatch defines a graph of **property** that we navigate with FDL. Each property can contain other properties or values (values do not contain other properties). We move from one property to the other with the “/” sign.

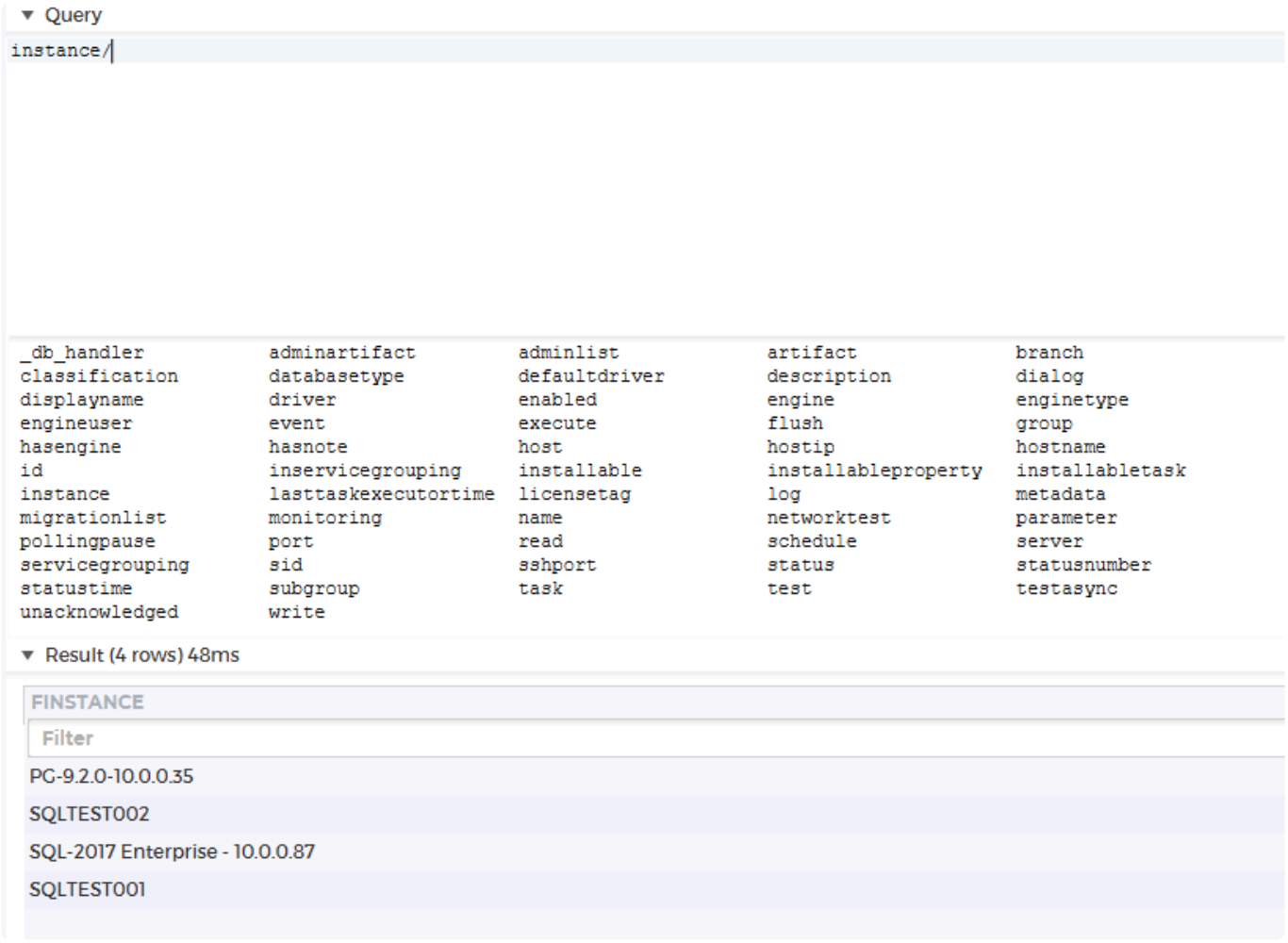
The **start** property is the property representing (all) the dbWatch Server(s).

If you type **instance** in the debug console, you will see a list of all the instances that are registered on your dbWatch Servers.
In the example below, I typed *Instance/name* to return the given names of each instance. If I typed *instance* instead, it will return the registered id of each instance.

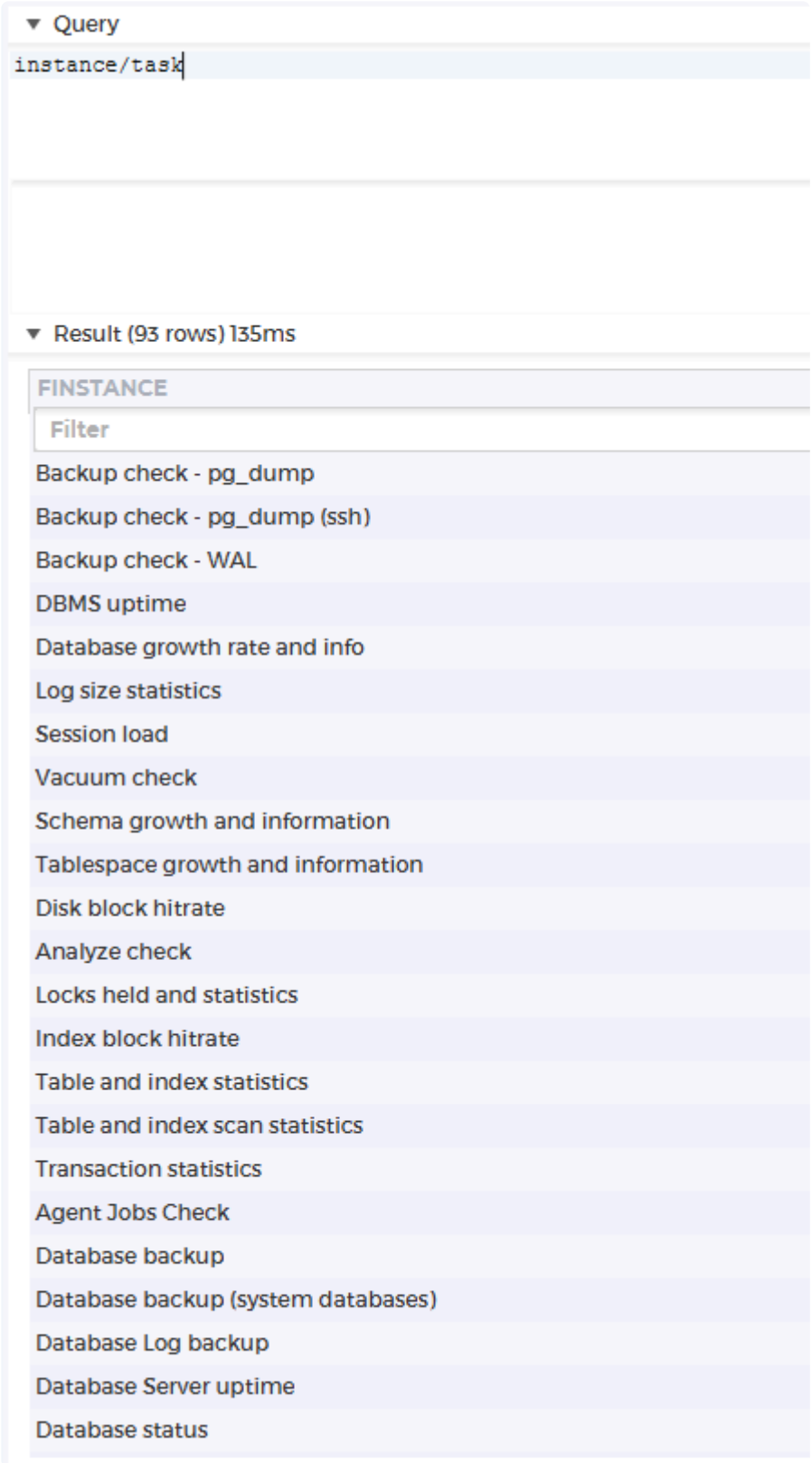


If you add a "/" and hit the **tab** key, you will see all the entities and values that are available from the instances. Going back to the example below, I used *instance* instead of the previously written query *Instance/name*. There are two main reasons for doing so:

1. Think of **properties** as directories. By accessing one directory, you limit or expand the subdirectories associated with that directory. Similarly, properties can only be attributed to another property if there is an association, to begin with.
2. Next, the backslash ("/") and tab key combination is some sort of suggested paths. As mentioned previously, if there is no association with any property then it won't return any value. Since *Instance/name* has no property connected to it then it won't return any result.



Expand the query again by typing “instance/task”. The result will then show a list of all installed tasks (or alerts) on all instances.



As you will observe, the default is that only the last “column” in the query is shown in the result set (in this case task is shown, but not instance). In order to output other “columns”, you can add curly brackets “{}” to the property you want to show. Don’t mind the arrows first. This will be discussed in depth in **Anchors**

```
instance->i/name?{"Name"}/$i/task{}
```

Result (93 rows) 135ms

NAME	FTASK
Filter	Filter
PG-9.2.0-10.0.0.35	Tablespace growth and information
PG-9.2.0-10.0.0.35	Disk block hitrate
PG-9.2.0-10.0.0.35	Analyze check
PG-9.2.0-10.0.0.35	Locks held and statistics
PG-9.2.0-10.0.0.35	Index block hitrate
PG-9.2.0-10.0.0.35	Table and index statistics
PG-9.2.0-10.0.0.35	Table and index scan statistics
PG-9.2.0-10.0.0.35	Transaction statistics
SQLTEST002	Agent Jobs Check
SQLTEST002	Database backup
SQLTEST002	Database backup (system databases)
SQLTEST002	Database Log backup
SQLTEST002	Database Server uptime
SQLTEST002	Database status

As you can see, the last column (task in this case) is always shown. You don't need to add a {} at the trailing column (as seen above) but for future edits of your FDL query, it makes it easier to append additional columns. Inside the curly brackets, you can rename any column using aliases. Similar to any query, all alphanumeric characters inside the open and closed brackets will substitute as their column name.

Filters

You can apply filters to the query by adding an expression that evaluates to true/false in square brackets.

In the previous example, we queried for the list of tasks of all registered instances. If we want the tasks of only some of the instances, we can filter on properties of the instances. For example, if we want only PostgreSQL instances that have a name that contains "2.0" in between its name, we could write as follows:

Query

```
instance[databasetype = 'postgres' & name like '%2.0%']->i/name?{}/$i/task{}
```

Result (18 rows) 63ms

?(FNAME)	FTASK
Filter	Filter
PG-9.2.0-10.0.0.35	Backup check - pg_dump
PG-9.2.0-10.0.0.35	Backup check - pg_dump (ssh)
PG-9.2.0-10.0.0.35	Backup check - WAL
PG-9.2.0-10.0.0.35	DBMS uptime
PG-9.2.0-10.0.0.35	Database growth rate and info
PG-9.2.0-10.0.0.35	Log size statistics
PG-9.2.0-10.0.0.35	Session load
PG-9.2.0-10.0.0.35	Vacuum check
PG-9.2.0-10.0.0.35	Schema growth and information
PG-9.2.0-10.0.0.35	Tablespace growth and information
PG-9.2.0-10.0.0.35	Disk block hitrate
PG-9.2.0-10.0.0.35	Analyze check
PG-9.2.0-10.0.0.35	Locks held and statistics
PG-9.2.0-10.0.0.35	Index block hitrate
PG-9.2.0-10.0.0.35	Table and index statistics
PG-9.2.0-10.0.0.35	Table and index scan statistics
PG-9.2.0-10.0.0.35	Transaction statistics
PG-9.2.0-10.0.0.35	_job_max_connections

You can add filters to all the columns. So if we want to limit the result further to only tasks that have a name starting with “Backup”, we could do the following

Query

```
instance[databasetype = 'postgres' & name like '%2.0%']->i/name?{}/$i/task[name like 'Backup%']
```

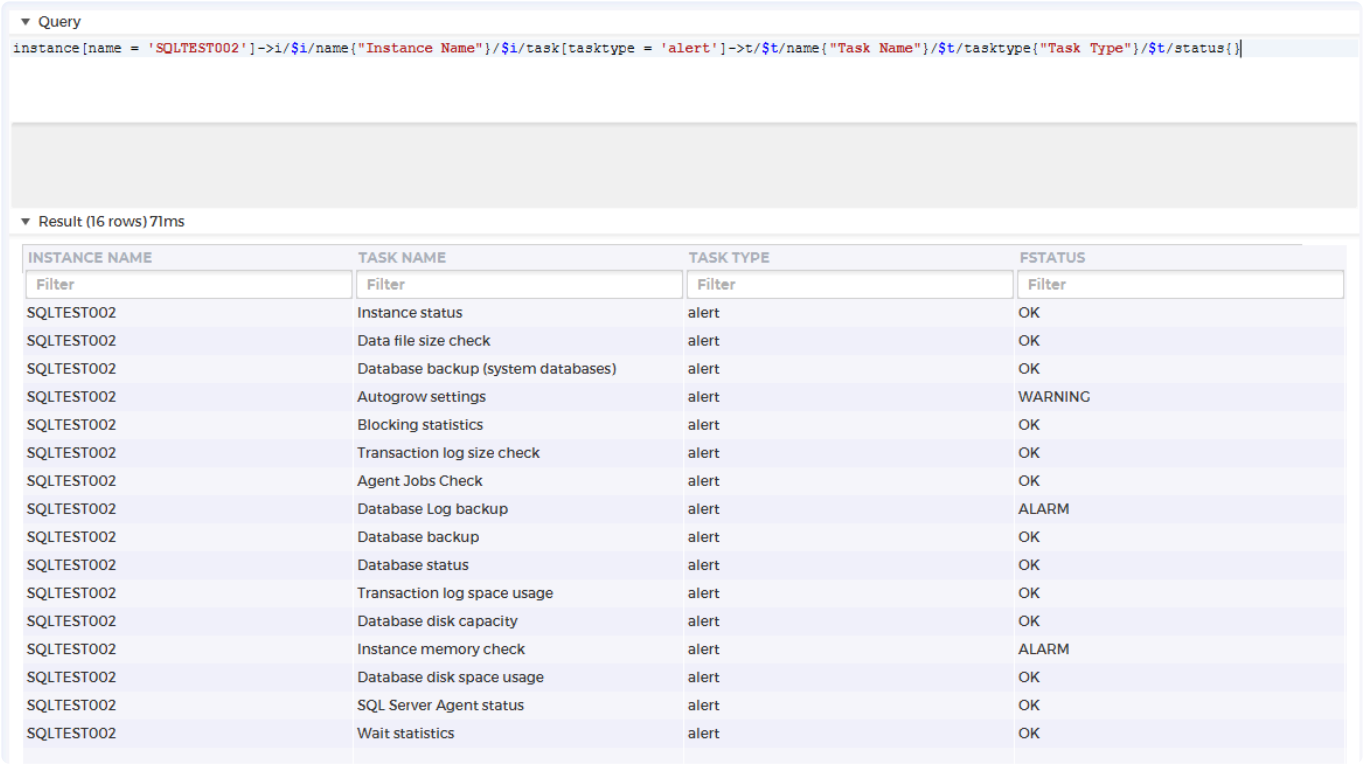
Result (3 rows) 51ms

?(FNAME)	FTASK
Filter	Filter
PG-9.2.0-10.0.0.35	Backup check - pg_dump
PG-9.2.0-10.0.0.35	Backup check - pg_dump (ssh)
PG-9.2.0-10.0.0.35	Backup check - WAL

Anchor

What if you want to list several properties from a property? Then we use an anchor. An anchor is defined by “->” and accessed by “$”.

For example, if we want to list the name and status for all alerts on a specific instance we could do the following



Query

```
instance[name = 'SQLTEST002']->i/$i/name{"Instance Name"}/$i/task[tasktype = 'alert']->t/$t/name{"Task Name"}/$t/tasktype{"Task Type"}/$t/status{}
```

Result (16 rows) 71ms

INSTANCE NAME	TASK NAME	TASK TYPE	FSTATUS
Filter	Filter	Filter	Filter
SQLTEST002	Instance status	alert	OK
SQLTEST002	Data file size check	alert	OK
SQLTEST002	Database backup (system databases)	alert	OK
SQLTEST002	Autogrow settings	alert	WARNING
SQLTEST002	Blocking statistics	alert	OK
SQLTEST002	Transaction log size check	alert	OK
SQLTEST002	Agent Jobs Check	alert	OK
SQLTEST002	Database Log backup	alert	ALARM
SQLTEST002	Database backup	alert	OK
SQLTEST002	Database status	alert	OK
SQLTEST002	Transaction log space usage	alert	OK
SQLTEST002	Database disk capacity	alert	OK
SQLTEST002	Instance memory check	alert	ALARM
SQLTEST002	Database disk space usage	alert	OK
SQLTEST002	SQL Server Agent status	alert	OK
SQLTEST002	Wait statistics	alert	OK

As you can see above, we place all the task (of type alert) belonging to the instance called “SQLTEST02” in an anchor called **t**, then we can use **$t** later in the query to reference the task name and status. As seen in previous sections, we also used **i** as our anchor. It shows how we can use any letter or string of characters as our anchors. For more simplicity, it’s better to avoid very long strings of character, as anchors are supposed to make it easier to refer to a property.

Operators

The following operators are available in the FDL language.

Operator	Meaning	Example	Description
=	equals	instance[port = 3306]	Can be used in the filter part of a FDL.
!=	not equals	instance[port != 3306]	Can be used in the filter part of a FDL.
<	less than	instance[total_session_count < 50]	Can be used in the filter part of a FDL.
<=	less than or equal	instance[total_session_count <= 50	Can be used in the filter part of a FDL.
>	greater than	instance[memory_usage > 300	Can be used in the filter part of a FDL.
>=	greater than or equal	instance[memory_usage >= 300]	Can be used in the filter part of a FDL.
!	inverts a boolean value	instance[!(host like	Can be used in the filter part of a

		’192.168.%’)]	FDL.
&	and	instance[total_session_count < 50 & memory_usage < 100]	Can be used in the filter part of a FDL.
|	or	instance[total_session_count < 50 | memory_usage < 100]	Can be used in the filter part of a FDL
like	string matching with ‘%’ wildcard	instance[host like ’192.168.%’]	Can be used in the filter part of a FDL.
matches	regular expression matching	instance[name matches ‘[O-T].*’]	Can be used in the filter part of a FDL.
->	create anchor	instance->i/$i/name{}/$i/status{}	Creates an anchor of a propery that can be referenced later by prefixing the assigned name with $.
{x}	assign column number	instance->i/$i/name{ 0 }/$i/status{ 2 }/$i/host{ 1 }	Postfixing a value with a number between curly braces causes that value to be in the resultset at the column number specified. By default only one column will be produced, containing the last value in the query.

Examples of FDL queries

Fetch the names of all instances.

```
instance/name
```

Fetch all tasks on the instance named “Instance one”.

```
instance[name="Instance one"]/task
```

Fetch all tasks with status Alarm and the corresponding instance name

```
instance->inst/task[status="ALARM"]{ 0 }/$inst/name
```

Fetch the name of all instances of with a total_session_count above 10 and that are of database type oracle or SQL server.

```
instance[total_session_count > 10 & (databasetype='oracle' | databasetype = 'sqlserver')]/name
```

Result rendering

The Debug console can render the query results based on hashtags in the query. These hashtags work as a hint to the GUI and are not part of the FDL language. They will currently have no effect on queries executed in other contexts (f.ex through the CLI.)

The currently available render hashtags are:

Hashtag	Example	Description
#rendertext	instance/name{#rendertext}	This is the default rendering.
#renderdate	instance/task/ lastruntimestamp{#renderdate}	Will attempt to render the value as a time if the value is today, otherwise as datetime (and it will be sorted as an datetime, not string).
#rendershortdate	server/started{#rendershortdate}	Will attempt to render the value as a date.
#renderfulldate	instance/task/ lastruntimestamp{#renderfulldate}	Will attempt to render the value as a datetime (and it will be sorted as an datetime, not string).
#renderstatus	instance/task{#renderstatus}	Will attempt to render the value as a status icon.
#renderstatusraw	instance/status{#renderstatusraw}	Will attempt to render the value as an untyped status icon.
#renderint	instance/db_count{#renderint}	Will attempt to render the value as an integer (and it will be sorted as an int, not string).
#renderdecimal	instance/db_count{#renderdecimal}	Will attempt to render the value as a decimal (and it will be sorted as an decimal, not string).
#renderpercent	instance/cpuload{#renderpercent(100)}	Will attempt to render the value as percent of the argument (Default 100). (So if your values are in the range 0..1, you could specify #renderpercent(1))
#renderbar	instance{}/buffer_miss_pct{#renderbar(100) #warning(70) #alarm(90)} instance{}/buffer_miss_pct{#renderpercent #renderbar(100) #labelwidth(100)}	Will attempt to render the value as a bar. The argument is max value (default 100), with no warning/alarm threshold set. Adding #warning and/or #alarm with arguments in percent will cause the bar to be colored red,orange,green based on theese thresholds. Can be combined with renderint/decimal/percent and an optional labelwidth

#renderboolean	instance{}/hasnote{#renderboolean #iconfalse #icontrue(document.png)}	Excpects a true/false value and will render the values as icons. The default icons can be overridden by specifying #iconfalse/#icontrue. Specifying no argument (like #iconfalse in the example), will result in no icon being displayed. Specifying an icon file (like #icontrue(document.png), will result in that icon being used. (Feature available from version 12.1.5)
#headericon	instance/ hasnote{#headericon(editscript.png)}	Will display the specified icon as the column header
#renderraw	instance{#renderraw}/db_count	Will show the instance column as rawdata and not a “nice” string.
#sort	instance{#sort(asc)}	Will sort the result on this column. Default is descending. Legal arguments are ‘asc’ and ‘desc’.
#width	instance/displayname{#width(120)}	Will cause the displayname column to be restricted to a width of 120px.
#fit	instance/started{#fit(xx.xx.xxx xx xx xx)}	Will cause the instance column to be wide enough to display the specified text. Normally the column is wide enough to display the column header text. The #fit tag makes room for the specified text instead.
#hide	instance{#hide}/name	Will hide the column from the result.
#owner	instance{#owner}/name	This value will be used to create the correct menu when right clicking on a row.
#id	instance{#id}/name	This value will be used as an index when optimizing the updating of the result table.

FDL terminology

property

A dbwatch property identifies a resource. Dbwatch properties consist of a property type, id, and context.

Represented as a string property can look like this;

- “instance:My database@server:My server” or
- “user:Jack@securitycontext:default@server:My server”

Topic

A topic is a semantic description of an edge in the semantic graph. Dbwatch comes with a set of predefined topics. It is also possible to define new topics to fit your enterprise needs.

<< Advanced Topics / Functions >>

Functions

Functions in FDL

Numeric functions

Function name	Description
add	Adds a number to a base value
sub	Substracts a number from a base value
div	Diveds a number
mult	Multiplies a number
abs	Returns the absolute value of a number
until	Return all numbers from a base number until an end number
round	Rounds a number to a specified precision

Aggregate functions

Function name	Description
groupby	Groups a set on some value(s)
count	Counts the number of rows
sum	Sums all values in a set
avg	Computes the average of set
max	Finds the max value in a set
min	Finds the minimum value in a set
orderby	Orders a set

String functions

Function name	Description
upper	Converts a string to uppercase
lower	Converts a string to lowercase
trim	Strips empty spaces from the beginning and end of a string
concat	Concats a number of strings
capitalize	Capitalizes a string
mkstring	Makes a string from a set

nicedb	Convert databasetype to nice string
replace	Replaces all occurences of string A with string B in a string C

Set functions

Function name	Description
distinct	Removes duplicates from a set of values
union	The union of two sets
intersect	The intersection of two sets
difference	The difference of two sets
empty	If a set is empty
nonempty	If a set is nonempty
containsall	Checks if two sets are equal
containssome	Checks if two sets have overlapping elements
containsnone	Checks if two sets have no overlapping elements

Time functions

Function name	Description
todate	Converts a value to datestring
tomillis	Converts a datetime string to milliseconds
toperiod	Converts a number to a period string

add function

Description

Adds a number to a base value.

Example

Find the port number above the current port number for an instance

```
instance[name = 'A']/port/add(1)
```

Syntax

add(number)

Parameters

Type	Description
Number	The number to add

[<< Functions](#) / [sub function >>](#)

sub function

Description

Substracts a number from a base value.

Example

Find the port number below the current port number for an instance

```
instance[name = 'A']/port/sub(1)
```

Syntax

sub(number)

Parameters

Type	Description
Number	The number to substract

<< add function / abs function >>

abs function

Description

Returns the absolute value of a number

Example

Find the absolute value for -3

```
abs(-3)
```

Syntax

abs(number)

Parameters

Type	Description
Number	The number to find the absolute value for

<< sub function / until function >>

until function

Description

Return all numbers from a base number until an end number

Example

Return the numbers in the range 1 to 5.

```
1/until(5)
```

Syntax

until(number)

Parameters

Type	Description
Number	The endpoint for the range

<< abs function / groupby function >>

groupby function

Description

Groups a set on some value(s). Used in combination with aggregate function(s).

Example

Group the instances by database type, and count the number of instances in each group

```
instance->i/groupby(databasetype)->g/$g/databasetype{}/count($i)
```

```
instance->i/groupby(databasetype)->g/$g/_1{}/count($i)
```

```
instance->i/groupby(databasetype, version)->g/$g/_1{}/$g/_2{}/count($i)
```

Syntax

groupby(topic1, topic2, ..)/aggregatefunction(..)

Parameters

Type	Description
Topic	The topic(s) to group by

Discussion

When using the groupby function, a group is created. This group can be anchored with the normal “->” syntax for later reference.
The topics used to create the group can be referenced by name (like “databasetype” in the first example), or by index (like _1 in the second example).

[<< until function](#) / [count function >>](#)

count function

Description

Counts the number of rows in a set

Example

Find the number of instances with status warning.

```
count(instance[status='WARNING'])
```

Syntax

count(expression)

Parameters

Type	Description
Expression	An expression that gives a number of rows

<< groupby function / sum function >>

sum function

Description

Sums the numbers in a set

Example

Find the total number of sessions for all your oracle instances

```
sum(instance[databasetype = 'oracle']/total_session_count)
```

Syntax

sum(expression)

Parameters

Type	Description
Expresssion	Expression that gives a set of numbers

<< count function / avg function >>

avg function

Description

Calculates the average of a set of numbers

Example

Find the average number of sessions for all your oracle instances

```
avg(instance[databasetype = 'oracle']/total_session_count)
```

```
avg(instance[databasetype = 'oracle']/total_session_count, 2)
```

Syntax

avg(expression)
avg(expression, precision)

Parameters

Type	Description
Expresssion	Expression that gives a set of numbers
Number	Optional argument specifying the number of decimals in the result

<< sum function / max function >>

max function

Description

Finds the max value from a set, or from a number of values.

Example

Find the max number of sessions for all your oracle instances

```
max(instance[databasetype = 'oracle']/total_session_count)
```

Returns the largest of the arguments (average session_count or 10)

```
max(avg(instance[databasetype = 'oracle']/total_session_count), 10)
```

Syntax

max(expression)
max(expression1, expression2, ..)

Parameters

Type	Description
Expresssion	Expression that gives a set of numbers

|_. Type |_. Description | | Expresssion | Expression that gives a number | | Expresssion | Expression that gives a number | | Expresssion | Optional expression that gives a number | | Expresssion | Optional expression that gives a number | | … | … |
p=. [<< avg function] / [min function >>]

min function

Description

Finds the minvalue from a set, or from a number of values.

Example

Find the minnumber of sessions for all your oracle instances

```
min(instance[databasetype = 'oracle']/total_session_count)
```

Returns the smallest of the arguments (average session_count or 10)

```
min(avg(instance[databasetype = 'oracle']/total_session_count), 10)
```

Syntax

min(expression)
min(expression1, expression2, ..)

Parameters

Type	Description
Expresssion	Expression that gives a set of numbers

Type	Description
Expresssion	Expression that gives a number
Expresssion	Optional expression that gives a number

<< max function / upper function >>

upper function

Description

Converts a string to uppercase

Example

Get all hostnames as uppercase

```
instance/hostname/upper()
```

```
instance/upper(hostname)
```

Syntax

upper(expression)

value/upper()

Parameters

Type	Description
Expression	Expression giving a (set of) strings

<< min function / lower function >>

lower function

Description

Converts a string to lowercase

Example

Get all hostnames as lowercase

```
instance/hostname/lower()
```

```
instance/lower(hostname)
```

Syntax

lower(expression)

value/lower()

Parameters

Type	Description
Expression	Expression giving a (set of) string(s)

<< upper function / trim function >>

trim function

Description

Strips empty spaces from the beginning and end of a string

Example

Get all hosts without leading or trailing spaces

```
instance/host/trim()
```

```
instance/trim(host)
```

Syntax

trim(expression)

value/trim()

Parameters

Type	Description
Expression	Expression giving a (set of) string(s)

[<< lower function](#) / [concat function >>](#)

concat function

Description

Concats a number of strings

Example

Get all hosts and append the domain “.dbwatch.com”

```
instance/concat(host, '.dbwatch.com')
```

Syntax

concat(string1, string2, …)

<< trim function / capitalize function >>

capitalize function

Description

Capitalizes a string

Example

Get and capitalize all instance names

```
instance/name/capitalize()
```

```
instance/capitalize(name)
```

Syntax

capitalize(expression)

value/capitalize()

<< concat function / distinct function >>

distinct function

Description

Removes duplicates from a set of values

Example

Get the set of versions for all instances

```
distinct(instance/version)
```

```
instance/distinct(version)
```

<< capitalize function / mkstring function >>

mkstring function

The mkstring function makes a string from a set of values

Description

Creates a string from a set of values

Example

Make a comma separated string of all instance names

```
mkstring(instance/displayname, ", ")
```

Syntax

mkstring(expression)
mkstring(expression, separator)

Parameters

Type	Description
Expresssion	Expression that gives a set of values
Separator	Optional argument specifying the separator to use, default is “, “

<< distinct function / nicedb function >>

nicedb function

Description

Converts a databasetype to a nice string representation

Example

List the database types in your environment

```
instance/databasetype/nicedb
```

```
instance/nicedb(databasetype)
```

Syntax

nicedb(databasetype)

databasetype/nicedb()

<< mkstring function / union function >>

union function

The union function finds the union of sets

Description

The union function finds the union of sets

Example

Find the disk space task from all test instances and disk capacity task from all production instances, and their version

```
union(
   instance[name like '%test%']/task[name like '%disk space%'],
   instance[name like '%prod%']/task[name like '%disk cap%']
)->t{}/$t/version
```

Syntax

union(expression, expression, ..)

Parameters

Type	Description
Expresssion	Expression that gives a set of values

<< nicedb function / intersect function >>

intersect function

The intersect function finds the values that are in both set A and set B

Description

Finds the intersection of two sets

Example

Find tasks from the ‘sla_higs_tasks’ list that are installed on instances tagged with sla=low

```
intersect(
      instance[sla='low']/task/name,
      list[name='sla_high_tasks']/content/name
)
```

Syntax

intersect(expression, expression)

Parameters

Type	Description
Expresssion	Expression that gives a set of values
Expresssion	Expression that gives a set of values

<< union function / difference function >>

difference function

(Available from september 2023 release)

The difference function finds the values that are in set A but not in set B

Description

Finds the difference of two sets

Example

Finds the elements in the first set that are not part of the second set.

```
difference(
  union("A", "B"),
  union("B", "C", "D")
)
```

Will give result “A”

Example

Find jobs that are installed on instance ‘test-1’ but not on ‘test-2’.

```
difference(
    instance[name = 'test-1']/task/id,
    instance[name = 'test-2']/task/id
)
```

Syntax

difference(expression, expression)

Parameters

Type	Description
Expresssion	Expression that gives a set of values
Expresssion	Expression that gives a set of values

<< intersect function / round function >>

round function

The round function rounds a number to a specified precision

Description

Rounds a number

Example

Find tasks from the ‘sla_higs_tasks’ list that are installed on instances tagged with sla=low

```
instance/disk_space/round(2)
round(instance/disk_space, 2)
```

Syntax

expression/round(precision)
round(expression, precision)

Parameters

Type	Description
Expresssion	Expression that gives a set of numeric values
Precision	The number of decimal points to round to

<< difference function / div function >>

div function

Description

Diveds a number

Example

Divide the number of instances by 7

```
count(instance)/div(7)
div(count(instance), 7)
```

Syntax

expression1/div(expression2)
div(expression1, expression2)

Parameters

Type	Description
Expression1	Expression that gives a (set of) number(s)
Expression2	Expression that gives a number to divede by

<< round function / mult function >>

mult function

Description

Multiples a number

Example

Multiply the number of instances by 2

```
count(instance)/mult(2)
mult(count(instance), 2)
```

Syntax

expression1/mult(expression2)
mult(expression1, expression2)

Parameters

Type	Description
Expression1	Expression that gives a (set of) number(s)
Expression2	Expression that gives a number to multiply by

<< div function / replace function >>

replace function

Description

Replaces all occurences of string A with string B in a string.

Example

Replace all occurences of “abc” with “def” in instance names.

```
instance/name/replace("abc", "def")
```

Syntax

expression1/replace(expression2, expression2)

Parameters

Type	Description
Expression1	Expression that gives (a set of) text(s)
Expression2	Expression that gives a text to replace
Expression3	Expression that gives a text to replace with

<< mult function / orderby function >>

orderby function

Description

Orders the rows in a set, with an optional limit on the number of rows to return

Example

Find the 10 instances with the highest number of sessions

```
instance->i{}/total_session_count/orderby('desc', 10)
```

Order the instances based on memory_usage

```
instance->i{}/memory_usage/orderby()
```

Syntax

expression/orderby(sortorder, limit)

Parameters

Type	Description
Expresssion	Expression that gives the rows to sort
sortorder	‘desc’ or ‘asc’, default is ‘desc’
limit	The number of rows to return, default is no limit

<< replace function / todate function >>

todate function

Description

Converts a value to a datetime string

Example

```
server/time/todate

Result: 10.sep.2024 10:44:47
```

```
server->s/todate($s/time)

Result: 10.sep.2024 10:44:47
```

```
server->s/todate($s/time, 'dd-MM-yyyy')

Result: 10-09-2024
```

Syntax

number/todate

todate(number, format)

Parameters

Type	Description
Number	The number to convert to a datettime
Format	An optional format

<< orderby function / empty function >>

empty function

Description

Returns true if the specified set is empty, otherwise false

Example

Find what instances has no open tickets

```
instance->i/name{"Instance"}/$i/empty(tickets[status = 'OPEN']){"All tickets c
losed"}
```

Syntax

empty(set)

Parameters

Type	Description
Set	A set that can be empty

<< todate function / nonempty function >>

nonempty function

Description

Returns true if the specified set is not empty, otherwise false

Example

Find what instances has open tickets

```
instance->i/name{"Instance"}/$i/nonempty(tickets[status = 'OPEN']){"Open tickets"}
```

Syntax

nonempty(set)

Parameters

Type	Description
Set	A set that can be empty

[<< empty function](#) / [tomillis function >>](#)

tomillis function

Description

Converts a datetime to milliseconds since 1970

Example

```
tomillis('01-01-2024', 'dd-MM-yyyy')

Result: 1704063600000
```

Syntax

tomillis(datetime, format)

Parameters

Type	Description
Datetime	A string representing a datetime
Format	The format of the datetime string

<< nonempty function / toperiod function >>

toperiod function

Description

Converts a number to a period string.

Example

```
toperiod(1000, 's')

Result: 16m 40s 0ms
```

Syntax

toperiod(number, numbertype)

Parameters

Type	Description
Number	The number to convert
Numbertype	How to enterprit the number (ms=milliseconds, s=seconds, m=minutes, h=hours, d=days)

<< tomillis function / containsall function >>

containsall function

Description

Checks if two sets are equal

Example

Check if the set of all instances is equal to the set of all instances with status OK.

```
server[containsall(instance, instance[status = 'OK'])]/name
```

Syntax

containsall(set1, set2)

Parameters

Type	Description
Set	A set to compare
Set	A set to compare

[<< toperiod function](#) / [containsnone function >>](#)

containsnone function

Description

Checks if two sets have no overlapping elements

Example

Syntax

containsnone(set1, set2)

Parameters

Type	Description
Set	A set to compare
Set	A set to compare

<< containsall function / containssome function >>

containssome function

Description

Checks if two sets have anyoverlapping elements

Example

Syntax

containssome(set1, set2)

Parameters

Type	Description
Set	A set to compare
Set	A set to compare

<< containsnone function / Available properties >>

Available properties

Properties available on dbWatch Servers

Properties on instances

Properties available on jobs

Properties available on internal

<< containssome function / Properties on dbWatch Servers >>

Properties on dbWatch Servers

The available properties for server are:

Name	Description	Return type	Typical values	Extra
name, displayname	The name of the dbWatch Server	Entity	dbWatch Control Center	
instance	All the instances that this dbWatch Server is responsible for	List of Entities	838791624072954182, 587324160434844496, 531224945097604289, …	
instancecount	The number of instances this dbWatch Server is responsible for	Number	74	
target	All the targets that this dbWatch Server is responsible for. Currently this is the same as **instance**	List of Entities	838791624072954182, 587324160434844496, 531224945097604289, …	
host	The IP address of the dbWatch Server	String	192.168.7.27	
hostname	The host name for this Servers IP address. A reverse name lookup is performed.	String	Libra	
port	The port number this	Number	7100	

	dbWatch Server accepts connections on.			
time	The current time on the server as milliseconds since 1970	Number	1696925602129	
currenttime	A text representation of the current time on the server	String	Tue Oct 10 10:13:22 CEST 2023	
uptime	How long the dbWatch Server has been running	String	17 days 19 hours 44 minutes 1 seconds	
application	The name of the dbWatch application	Entity	dbWatch	
version	The raw version of the dbWatch Server	Number	1695376323674	
versioninfo	The nice version of the dbWatch Server	String	Control Center Server (2023-09-22)	
started	The start time of the dbWatch Server	String	09.okt.2023 10:45:28	
ostype	The operating system the dbWatch Server is running on.	String	Windows Server 2016/10.0/amd64	
osuser	The operating system user the dbWatch	String	MASSOSS$	

	Server is running under.			
group	The groups that are defined on this dbWatch Server.	List of Entities	Development, Production, Test	
networkrange	The network range that this dbWatch Server thinks its on. Used by the autodiscover process when scanning for unregistered instances	String	192.168.7.0/24	
installable	List of all the jobs that can be installed on instances	List of Entities	DBMS uptime, Lock statistics, Session load	
flush	Flushes the caches on the dbWatch Server	String	Index flushed	
domaincontroller	The domains this dbWatch Server is domain controller for	String	demo.3826.dbwatch.com	
discoveredinstance	List of instances that have been discovered by the autodiscover process	List of Entities	mysql/192.168.6.234/3306, postgres/192.168.6.223/5432, …	
autodiscoverscan	List of network ranges that should be	List of Entities	192.168.6.0/24	

	scanned for instances			
license	The dbWatch license for this Server.	Entity	license	
user	List of users defined in the domain. Only the Domain Controller will return values	List of Entities	admin, bob, mark, lisa	
cpuload	An approximation of the current cpu load on the Server	Number	12.76	
iospeed	A number of milliseconds it takes to create a small temp file and write a short text to it. The lower the better.	Number	5	
speed	The number of milliseconds it takes to perform a certain calculation. The lower the better	Number	92	
socketinfo	Debug info about the active connections on this server (dbWatch node connections, not database connections).	List of String	10.0.5.218:53410 Connected (id: {8583312666762166566,-4679215439455640848}, version: 22.sep.2023 11:52:03)	

socket	The active connections on this server. (dbWatch node connections, not database connections).	List of Entities	6898177628126849188, 6047584236597675033, …	
extension	The available extensions on this dbWatch Server (currently only E-Mail extension)	List of Entities	E-Mail extension	
file	The management, job and report xml files available to this dbWatch Server	List of Entities	SQL_Server_menuitem_disable_db_obj_sepec.xml, SQL_Server_menuitem_databases_restore_snapshot.xml …	
artifact	The management, job and report artifact available to this dbWatch Server.	List of Entities	com.dbwatch.management:sql_server_node_backups:1.3, com.dbwatch.management:mysql_result_cpu-pct-graph:1.0, …	
scheduledreport	The global reports that are scheduled on this dbWatch Server	List of Entities	Scheduled Report [Database backup report] – 3	
jobtemplate	The job templates that are defined on this dbWatch Server	List of Entities	To31ECbvuhHj8Kg6, Z60jdT3HkULk12Mh, …	
dictionary	The dictionaries that are	List of Entities	Default	

	defined on this dbWatch Server			

<< Available properties / Properties on Instances >>

Properties on Instances

Instances are available under server through the instance property

```
server/instance
```

All instances have the following properties:

Name	Description	Return type	Typical values	Extra
name	The given name of the instance	String	My Dev Instance	
displayname	The given name of the instance as an Entity	Entity	My Dev Instance	
databasetype	The database type of the instance	String	oracle, sqlserver, …	
host	The hostip/name that was supplied when registering this instance	String	10.0.0.83, localhost, ..	
hostip	The IP address of this instance	String	10.0.0.46	
hostname	The host name of this instance, fallback to **host** if the name is not resolvable	String	test1.mydomain.com	
port	The port dbWatch uses to connect to this instance	Number	1433, 1521, 3306, …	
status	The current status for this instance	Entity	OK, ALARM, WARNING, LOST CONNECTION, …	
statusnumber	The current status for this instance as a number	Number	1, 2, 4, …	
statustime	The time that the instance got its current status, as milliseconds since 1970	Number	1696935522503	
enginetype	The engine types defined for this instance (currently sql and ssh)	List of String	sql, ssh	
hasengine	YES/NO value indicating if this instance has a	String	YES/NO	

	dbwatch engine framework.			
engine	The name of the database/schema, if any, where the dbwatch engine framework is stored	String	dbwatch	
engineuser	The user dbWatch uses (if any) to connect to this instance to run engine procedures.	String	dbwatch	
propuser	The user dbWatch uses to connect to this instance in order to resolve dynamic properties.	String	sa, sys	
group	The group this instance is a part of	String	Development, Production, …	
subgroup	The sugroup this instance is a part of (if any)	String	Project A	
connected	Value indicating if dbWatch currently has a valid connection to this instance	String	true, false	
unacknowledged	Value indicating if this instance has jobs with unacknowledged warning/ alarms	String	true, false	
flush	Flushes the cache (on the dbWatch Server) for this instance	String	Flushed instance	
driver	The jdbc driver that is used to connect to this instance	Entity	com.microsoft.sqlserver:mssql-jdbc:10.2.0.jre8	
task	A list of all the dbWatch jobs that are installed on this instance	List of Entities	Query cache hitrate, Thread cache hitrate, …	
installabletask	A list of all engine jobs that are compatible with this instance	List of Entities	Transaction log size check, Transactions load, …	
installableproperty	A list of all non engine jobs that are compatible with this instance	List of Entities	instance_restart_alert, instance_connection_alert, …	

installable	A list of all jobs (engine and non engine) that are compatible with this instance	List of Entities	Transactions load, instance_connection_alert, …	
pause	If this instance currently has a monitoring pause	String	YES, NO	
alerting	If this instance currently has an alerting pause	String	YES, NO	
propauthtest	Performs a test connection to the instance using the defined property authentication	String	OK, No Auth, Connection to 10.0.1.192:5432 refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.	
monauthtest	Performs a test connection to the instance using the defined monitoring engine authentication (if any)	String	OK, No Auth, Connection to 10.0.1.192:5432 refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.	

In addition most instances have the following properties:

Name	Description	Return type	Typical values	Extra
platform_info	Platform and version info	Entity	Oracle 19.3.0.0.0, SQL Server 2016 (SP3)	
edition	The edition of the instance	Entity	Standard Edition, Enterprise Edition	
started	A text representation of the start time for the instance	Entity	21.10.2020 11:39:16, 11.10.2023 11:47:08	
total_session_count	The total number of sessions on the instance	Entity	12, 432	
disk_usage	The approximate disk usage (in GB) for the instance	Entity	445	
memory_usage	The approximate memory usage (in MB) for the instance	Entity	2450	
db_count	The number of databases on the instance	Entity	23	

<< Properties on dbWatch Servers / Properties on Jobs >>

Properties on Jobs

Jobs are available under instance through the task property

```
server/instance[name = '...']/task
```

The available properties are:

Name	Description	Return type	Typical values	Extra
name	The name of the job	String	Blocking statistics, Data file size check, …	
displayname	The name of the job as an Entity	Entity	Blocking statistics, Data file size check, …	
description	A short description explaining what this job does	String	Checks free space on drives where data and transaction ….	
longdescription	A longer description explaining what this job does	String	Checks free space on drives where data and transaction ….	
status	The current status of this job	Entity	OK, WARNING, ALARM	
statusnumber	The current status of this job representet by a number	Number	0, 1,2.	
statustime	The time the job got its current status in milliseconds since 1970	Number	1696935690159	
details	A text describing the current state after the last execution of the job	String	No backup information for 2 database(s) (master, msdb).	
schedule	The current execution schedule of the	Entity	5m, 30 * * *	

	job			
enabled	If this job is enabled	String	YES, NO	
version	The current version of the installed job	Number	1, 1.2	
maxversion	The max version available for this job	Number	2.2, NA	
upgradable	If this job is upgradable to a newer version	String	YES, NO	
category	The category for this job	Entity	Performance, Availability, Capacity, …	
nextrun	The next time the job is scheduled to execute	String	10.okt.2023 17:15:25	
nextruntimestamp	The next time the job is scheduled to execute, in milliseconds since 1970	Number	1696950925963	
lastrun	The last time the job was executed.	String	10.okt.2023 05:10:37	
lastruntimestamp	The last time the job was executed, in milliseconds since 1970	Number	1696942841019	
untilnextrun	The number of milliseconds until this job is scheduled to execute again	Number	575804	
taskid	The id used for the job in the engine framework (for	Number	12	

	engine jobs)			
company	The publisher of this job	String	dbwatch.no	
artifact	The artifact for this job	Entity	com.dbwatch.job:ms_transaction_log_size_check:2.1, com.dbwatch.job:ms_data_cache_memory_usage:1.6	
tasktype	The type of job. (alert = has a status, task = does not have a status)	String	alert, task	
unacknowledged	If this job has unacknowledged warning or alarms	String	true, false	
parameter	The available configuration parameters for this job	List of Entities	warning_threshold, alarm_threshold, …	

<< Properties on Instances / Properties on Internal >>

Properties on Internal

Internal is available under server through the internal property

```
server/internal
```

The available properties are:

Name	Description	Return type	Typical values	Extra
installdir	The dbWatch Server installation directory	Entity	C:/Program Files/ dbwatchControlCenter	
configdir	The path where the dbWatch Server puts the configuration files	Entity	C:/ProgramData/ dbWatchControlCenter/ config	
logdir	The path where the dbWatch Server puts the log files	Entity	C:/ProgramData/ dbWatchControlCenter/ log	
datadir	The path where the dbWatch Server puts data files	Entity	C:/ProgramData/ dbWatchControlCenter/ data	
javaversion	The java version used by the Server	String	1.8.0_372	
javaarch	If the architecture is 64 or 32 bit	Number	32, 64	
memoryusage	The approximate memory usage (in MB) of the dbWatch Server	Number	1282	
memoryreserved	The approximate amount of memory (in MB) that is reserved for the dbWatch Server	Number	1947	
memorymax	The approximate amount of memory (in MB) that the dbWatch Server is allowed to use	Number	5461	
threadqueue	The number of threads waiting in queue	Number	0	
threadcount	The total number of threads	Number	362	
threadtimeout	The amount of time (in seconds) a thread can be idle before it is removed from the thread pool	Number	60	
threadpoolsize	The size of the thread pool	Number	300	
workercount	The number of threads in the threadpool that are active	Number	134	

The FString format

The FString format.

FString is a format created to extract data from complex data structures in order to create well-formed files or strings.

The basic FString

- Any string is considered an FString.
- Any parts of the string starting with ‘{$’ and ending with ‘$}’ is a replacement directive.
- An FString support standard escape sequences like \n for newline.

The replacement directive

The replacement directive is any part of the string starting with ‘{$’ and ending with ‘$}’
The first part of this directive is the lookup key. This is a name referencing a value in the context in which the FString is evaluated.

So if we were to evaluate the FString “Hello {$user_name$}” in the context

```
{
  "user_name":"Bob"
}
```

We would get the string “Hello Bob”

There are some reserved lookup keys to enable different more complex replacements.

Key	Description
* or _	Treat the whole input as the result
0 1 2…	Look up the numbered element in a list
name	Look for the key name in the context
@	The name of the container for this element. For instance if the input is from a fdl query it will be the column name

For each in collections

Some times the context contains lists of values.

```
{
  "users":[
```

```
    "Bob",
    "Ben",
    "Barbra"
}
```

The special character '|' is used to repeat a string for each of the values in a list.

Key	Description
|	Repeat the folowing FString for each selected element
¤	Seperated by

So the expression {$users|{$_$}¤, $} is interpreted as look up the key "users", treat the value as a list and for each element in the list evaluate "{$_$}" meaning insert the value. Seperate these values by ", ". The expression would yeald the string "Bob, Ben, Barbra".

examples

Expression	What it does	result
{$*\|"{$_$}"¤, $}	Treat the input as a list. For every element insert the value inside ""	"Bob", "Ben", "Barbra"
Hi {$name$}	Replace with the value named "name"	Hi Bob
Hi {$1/name$}	Treat the input as a list. Replace with the name of the second value.	Hi Ben
{$*\|{"{$@$}":"{$_$}}"¤, $}	Treat the input as a list. For every element insert the column name then the value inside ""	{"name":"Bob"}, {"name":"Ben"}, {"name":"Barbra"}

<< Properties on Internal / Product Security >>

Product Security

Introduction on dbWatch Control Center regarding product security

dbWatch AS has implemented reasonable administrative, technical and physical safeguards to help protect against security incidents and privacy breaches involving a dbWatch product, provided those products are used in accordance with dbWatch instructions for use. However, as systems and threats evolve, no system can be protected against all vulnerabilities and we consider our customers the most important partner in maintaining security and privacy safeguards. If you have any concerns, we ask that you bring them to our attention and we will investigate. Where appropriate, we will address the issue with product changes, technical bulletins and/or responsible disclosures to customers and regulators. dbWatch continuously strives to improve security and privacy throughout the product lifecycle using practices such as:

- Privacy and Security by Design
- Product and Supplier Risk Assessment
- Vulnerability and Patch Management
- Secure Coding Practices and Analysis
- Vulnerability Scanning and Third-Party Testing
- Access Controls appropriate to Customer Data
- Incident Response
- Clear paths for two-way communication between customers and dbWatch.

If you would like to report a potential product related privacy or security issue (incident, breach or vulnerability), don't hesitate to get in touch with dbWatch by support email: support@dbwatch.com.
The purpose of this document is to detail how dbWatch security and privacy practices have been applied to the Control Center, what you should know about maintaining security of this product and how we can partner with you to ensure security throughout this product's lifecycle.

Contents

- Product Description
- Hardware Specifications
- Operating System
- Network Ports and Services
- Sensitive Data Transmitted
- Sensitive Data Stored
- Certificate infrastructure
- Crypto catalog
- Network Controls
- Encryption

- Audit Logging
- Remote Connectivity
- Disclaimer

<< The FString format / Product Description >>

Product Description

Product Description

dbWatch Control Center is a database operations software intended to monitor, manage, maintain and report on database instances across multiple database platforms and physical locations.
A dbWatch Control Center environment has at its core a Control Center server that has the “Domain CA” role. This server role controls authentication and security for all Control Center server nodes and monitors. See architecture more on this.

<< Product Security / Hardware Specifications >>

Hardware Specifications

Hardware Specifications

dbWatch Server Prerequisites

- Windows Server or Linux Ubuntu Server(VMWare virtual server supported)
- 8 GB RAM
- 4 CPU cores
- 1 GB HD Space (for dbWatch)

dbWatch Engine Prerequisites

- 500 Mb free space in each database instance.
- SQL Performance module (extra cost) requires additional space, around 5 GB.
- Engine – Server communication determined by each supported database platform.
- Bulk install for large database environments.
- Installs in under 2 minutes per instance.

dbWatch Client Prerequisites

- Windows or Linux operating system with a graphical interface.
- 500 Mb hard drive space.
- 8 GB RAM
- 4 CPU cores
- Mouse.
- Java support.
- Client-Server communication requires a single port only
- Client is installed automatically under Server installation

[<< Product Description] / [Operating System >>]

Operating System

Operating System

dbWatch Control Center supports:

Windows Server 2008 and newer
Windows 10 and newer

Linux Ubuntu 20.04 LTS

It could work fine on older or newer versions. Please check the install guides for more detailed information.

Install guide on Windows.
Install guide on Linux.

<< Hardware Specifications / Network Ports and Services >>

Network Ports and Services

Network Ports and Services

Default ports and protocol

dbWatch Control Center uses by default port 7100/tcp for client to node and node to node connections. This can be configured to a different port.
If enabled the webserver can run. Default port is 8080.
On each node there is a node.connections file that both lists what ports the dbWatch Server node listens to, and what nodes it has a preconfigured connection to.

Services

Port	Protocol	Service Name	Description of Service	Encrypted	Encryption type	Open/Closed
7100	TCP	Control Center Service	Server node listening port	Yes	AES/GCM/ PKCS5Padding (256bit keys)	Open(unless closed in the Control Center firewall)
8080	TCP	Control Center Service	Webserver listening port	No	N/A	Closed, unless enabled in domain configuration

<< Operating System / Sensitive Data Transmitted >>

Sensitive Data Transmitted

Sensitive Data Transmitted

dbWatch Control Center will, by its nature log into database instances and run queries on the database instances it manages. Key data accessed is primarily data related to statistics on the behavior of the database instance, and not actual database application data.
The interfaces that allow SQL Queries (Management / SQL Worksheet) can if is allowed and DBA's want, query other data as well.

All sensitive data sent between dbWatch Control Center nodes and between nodes and clients are encrypted. Data between dbWatch Control Center instance hubs and database instances is using JDBC and rely on the encryption available in the driver and database instance, which can vary between vendors and versions.

[<< Network Ports and Services] / [Sensitive Data Stored >>]

Sensitive Data Stored

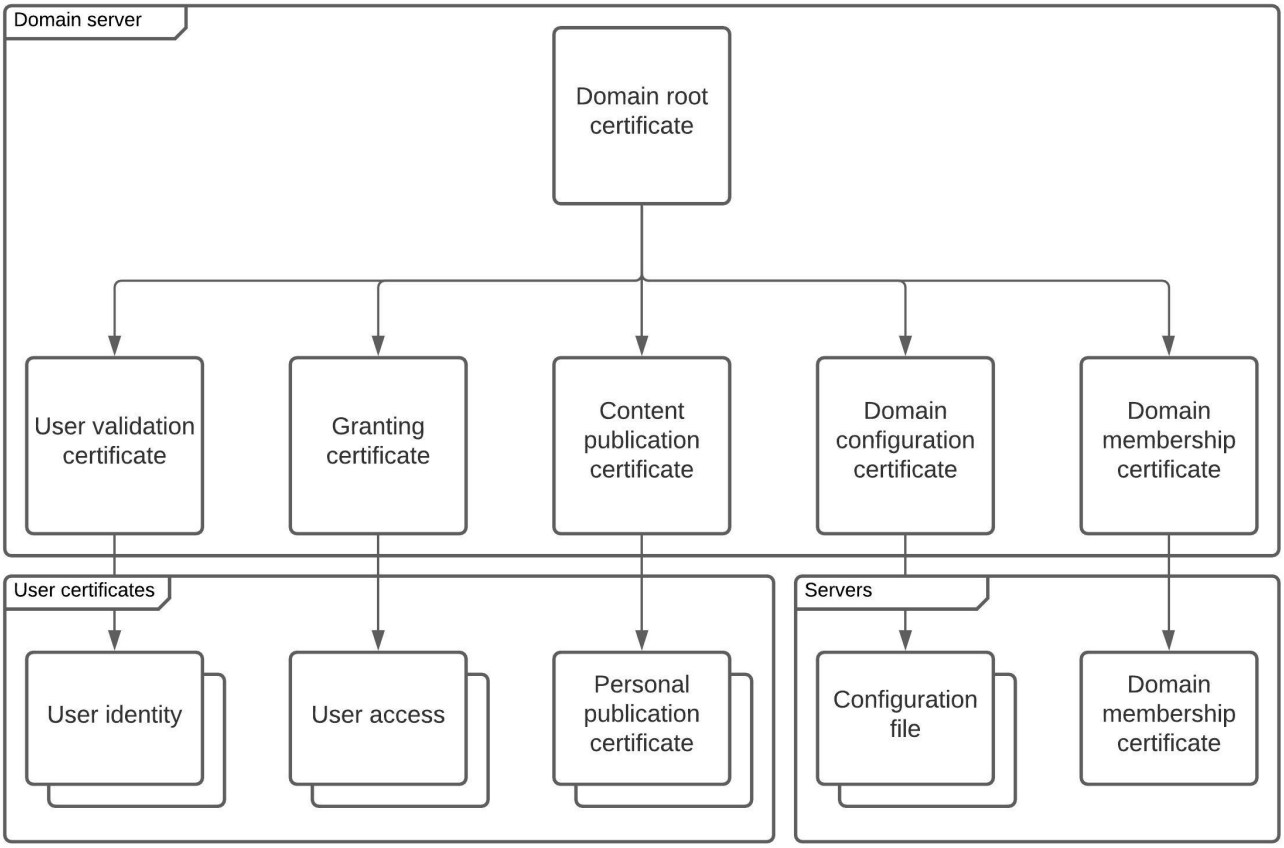
Sensitive Data Stored

dbWatch Control Center stores usernames and passwords needed to connect to database instances it monitors. The data is encrypted on the disk of the dbWatch Server.

<< Sensitive Data Transmitted / Certificate infrastructure >>

Certificate infrastructure

Certificate infrastructure



- Domain root

The root certificate of the domain. In order to be trusted by external domains this certificate must be signed by dbWatch.

- User identity certificate
  This certificate is used by the domain to sign user certificate requests.
- Granting certificate
  This certificate is used by the domain to grant rights to users.
- Configuration certificate
  This certificate is used by the domain to sign configuration files to be distributed to the domain.
- Publication certificate
  This certificate is used by the domain to sign

<< Sensitive Data Stored / Crypto catalog >>

Crypto catalog

Crypto catalog

The files in the config/crypto catalog are as follows:

On the domain controller:

File	Description
/ca/dbw_dc_cert.pem	domain controllers certificate
/ca/dbw_dc_pub.pem	domain controllers public key
/ca/dbw_dc_priv.pem	domain controllers private key
/keys/root/root_cert.pem	self signed root certificate
/keys/root/root_priv.pem	private key for self signed root
/keys/root/root_pub.pem	public key for self signed root

On the nodes:

File	Description
/keys/[domain_name]/dbw_cert.pem	the nodes certificate for this domain
/keys/dbw_pub.pem	the nodes public key
/keys/dbw_priv.pem	the nodes private key

<< Certificate infrastructure / Network Controls >>

Network Controls

Network Controls

dbWatch Control Center deploys several methods of network control.
1. Node authentication (clients and servers) using certificate infrastructure built up around the Domain CA system
2. The built-in firewall to restrict incoming connections on an IP level and encrypted package routing on a certificate domain level.
3. The actual listening on ports and connection direction can be adjusted.

Certificate infrastructure is discussed in detail here.

Control Center firewall is discussed in detail here.

Listening and connection direction is discussed in detail here.

<< Crypto catalog / Encryption >>

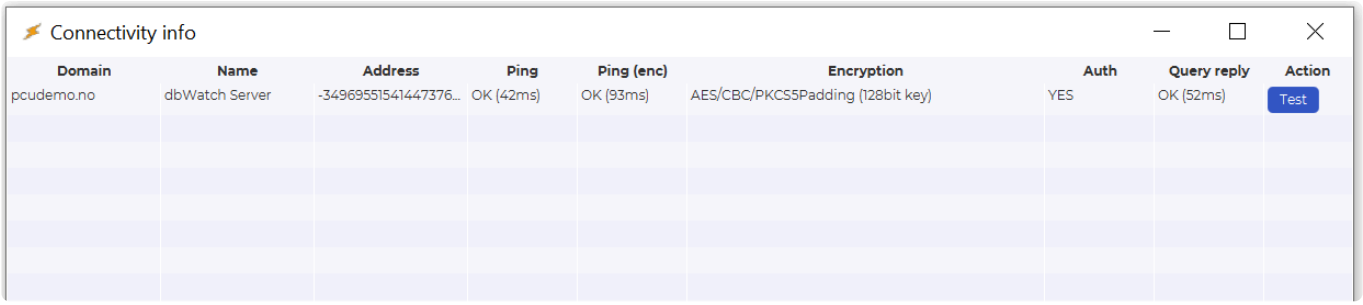
Encryption

Encryption

Data packages sent between server nodes and monitor clients are end-to-end encrypted with the node certificate.

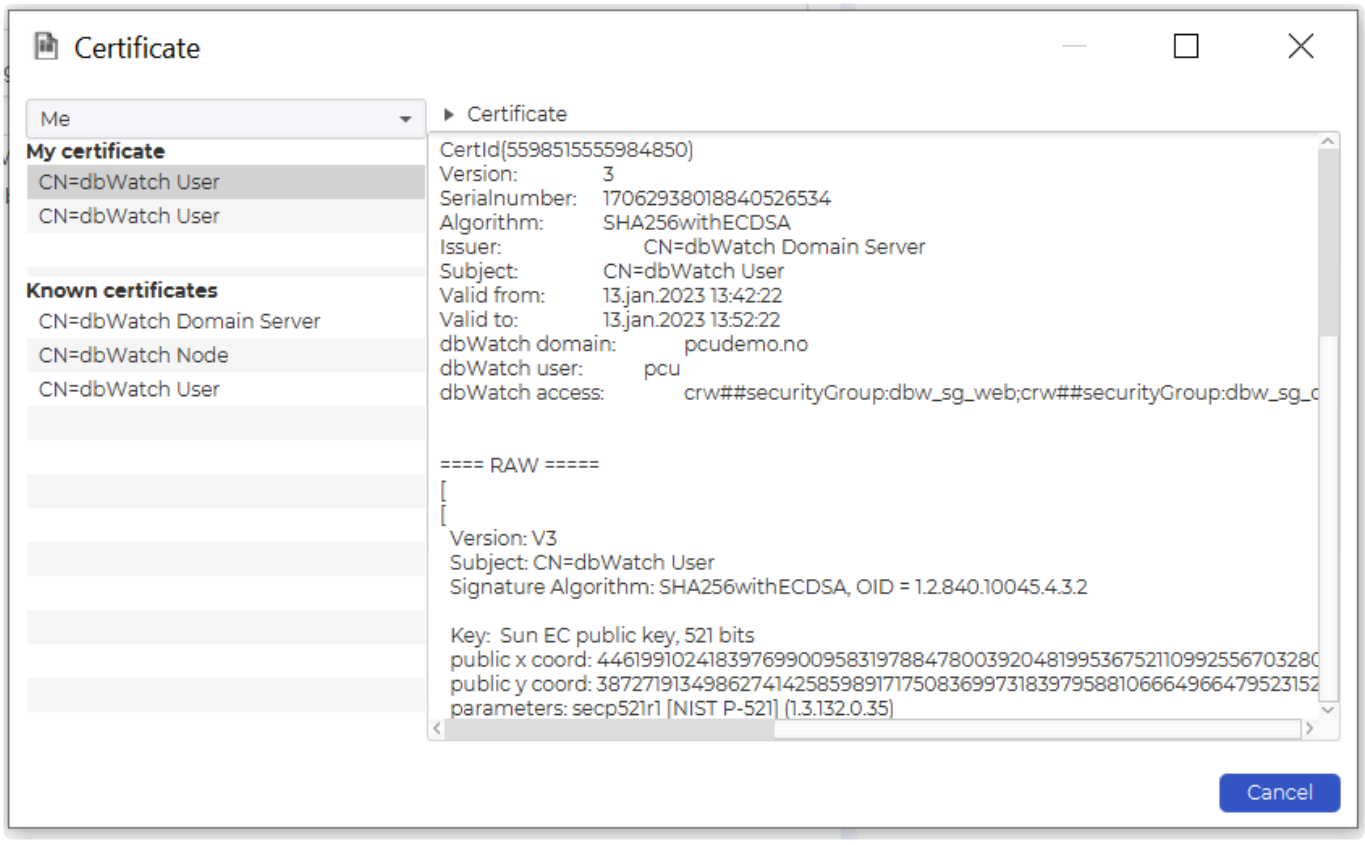
The encryption used for this is AES/GCM (256bit keys).

This is possible to verify if you click on Help->Debug->Connection info



For encryption and signing messages, certificates using SHA256withECDSA, is used, and regenerated every 10 minutes.

Details are available if you click on Help->Debug->Certificates



The negotiation of encryption uses Elliptic-curve Diffie-Hellman key agreement.

<< Network Controls / Audit Logging >>

Audit Logging

Audit Logging

Auditing in dbWatch Control Center is available to enable.

Currently, 3 log levels are available: 0=no logging, 1 = log statements triggered by user actions, 2 = log all statements by user actions and other database interactions.

Audit logs are stored on the disk of the Control Center Server.

More details are available on the wiki page for auditing.

<< Encryption / Remote Connectivity >>

Remote Connectivity

Remote Connectivity

dbWatch Control Center servers are by default configured to allow connections on port 7100/TCP.

This communication is encrypted and several network controls are available to restrict traffic.

Communication is only available to approved nodes, configured by a user with the right privileges to do so.

<< Audit Logging / Disclaimer >>

Disclaimer

Disclamer

The information contained in this Product Security Wiki is for reference purposes only. Nothing contained in this document or relayed verbally to any customer will be deemed to amend, modify or supersede the terms and conditions of any written agreement between such customer and dbWatch AS, or dbWatch AS subsidiaries or affiliates (collectively, “dbWatch”). dbWatch AS does not make any promises or guarantees to customer that any of the methods or suggestions described in this Product Security White Paper will restore customer’s systems, resolve any issues related to any malicious code or achieve any other stated or intended results. Customer exclusively assumes all risk of utilizing or not utilizing any guidance described in this Product Security White Paper.

Node to Node communication

dbWatch Control Center is composed of a set of nodes that speak to each other using a layered protocol.
This document describes these layers and how they interact to achieve security and reliability.
The dbWatch monitor is also a node and has all the behavior described here.

A brief overview of the layers

The protocol consists of these layers

1. tcp-ip – Two nodes communicate trough a tcp-ip connection.
2. tls – The traffic is wrapped inside a tls channel.
3. message layer – The nodes sends encrypted messages to each other. The messages are forwarded from node to node until it reaches the destination node.
4. Ephemeral tunnel – Most messages are sent encrypted

The tcp-ip layer

Two nodes talk to each other using tcp-ip. Any node can accept incoming connections on multiple ports and can connect to multiple ports.
This can be configured in the domain configuration dialog or in the config/node/node.connections file.
Example of a node.connections file.

```
{
        "versionCutoff":"",
        "listenTo":[
                "dbw://localhost:7100"
        ]
,        "connectTo":[
                "dbw://cloudrouter1.dbwatch.com:7100"
        ]
}
```

Transport Layer Security

Inside the tcp-ip connection we encrypt everything with TIs. Most people know this protocol from how it is used in the web(HTTPS).
There are some differences in the way dbWatch uses it. In addition to keeping the traffic between two nodes hidden the nodes use
tls to transfer proof of routing privileges to each other. Say you have 3 nodes in a chain. One Monitor a cloud router and a instance hub.
Both the Monitor and the instance hub sends certificates when negotiating the tls connection that tell the cloud router which domains they have routing privileges on. If they prove privileges on at least one common domain the cloud router will route between them.

All nodes that are part of the same domain have routing privileges within that domain. The monitor gets these privileges by authenticating to a given domain.

The message layer

The messages contain the following fields.

Field	Description	Size (bytes)
time to live	Time to live for the message	1
recipient	Recipient address (ip v 6)	16
sender	Sender address (ip v 6)	16
body category`	Category of the message body	1
send time	Time the message was sent	8
certificate id	Certificate ID used to sign the message.	8
is body encrypted	Flag if body is encrypted	1
body length	Length of the message body	4
message body	The content of the message	body length
signature length	The length of the signature	4
signature	The signature	signature length

The message layer looks at the fields and routes the message accordingly.
If the recipient is the nodes address the node validates the signature before processing the message.
The signature covers all the fields except for time to live.

Ephemeral tunnel. (Diffie-Hellman)

Since the tls layer only encrypts between two directly connected nodes we have a second encryption layer that encrypts the bodies of most messages.
Before you send a message to a node we perform a Diffie-Hellman handshake to establish ephemeral keys to secure encrypt message bodies.
This key is used to keep all communication between the nodes secret, and also works if someone on a later date gets hold of the servers keys.

Auditing

DbWatch supports auditing of all statements that are executed on the database instances from dbWatch.

There is currently no graphical way of turning this feature on, so it involves manually editing the server_configuration.xml file.

Stop the dbWatch Server and take a backup of this file before editing it.

Add the following tag to the file:

```
<audit-settings>
   <audit-setting>
       <audit-catalog>E:/dbWatch/dbw-audit</audit-catalog>
       <file-switch-interval-minutes>360</file-switch-interval-minutes>
       <audit-level>2</audit-level>
       <file-keep-for-days>7</file-keep-for-days>
   </audit-setting>
</audit-settings>
```

- **audit-catalog** The catalog where dbWatch will place the audit files. dbWatch will create a file called audit.log in this catalog. The catalog used must be created already, and be read/writeable by the user that runs the dbWatch Service.
- **file-switch-interval-minutes** How many minutes should pass before switching log files. When switching files, dbWatch creates a zip file of the log and names it with a timestamp. Then a new audit.log file is created and used.
- **file-keep-for-days** Zip files that are older than this will be deleted. A value of 0, means it will not delete any files.
- **audit-level** 0=no logging, 1 = log statements triggered by user actions, 2 = log all statements

Example log entry:

```
Entry [Feb 16, 2023 11:53:01 AM][pcu][10.0.1.112/1521][sys as sysdba]
SELECT 1 FROM DUAL
```

From this entry we can see the time, dbWatch user, the ip and port of the database we connect to, and the database credentials used, as well as the statement.

<< Node to Node communication / Control Center Commandline >>

Control Center Commandline

CCC.exe – the Control Center Commandline

CCC.exe is a command intended to run on the commandline or terminal to trigger USL scripts.
The intention of CCC is to use it for integration and automation purposes to access the capabilities of dbWatch for other 3rd party applications as part of a script.

dbWatch provides a set of scripts for CCC, currently located here

This zip contains scripts for setup and integration with dbWatch Control Center.

Files:

setup.script – Do initial setup steps on a CCC node
add_instance.script – Adds new instances to Control Center
fdl.script – Run FDL and get output
get_instance.script Retrive instance configuration data for an instance in XML format (Combines with register_instance.script for moving instances)
register_instance.script – Register instance based on a configuration file in XML format
metadata.script – Set metadata for a database instance
jobs.script – Install jobs on a database instance
instance_action.script – Connect or disconnect to an instance

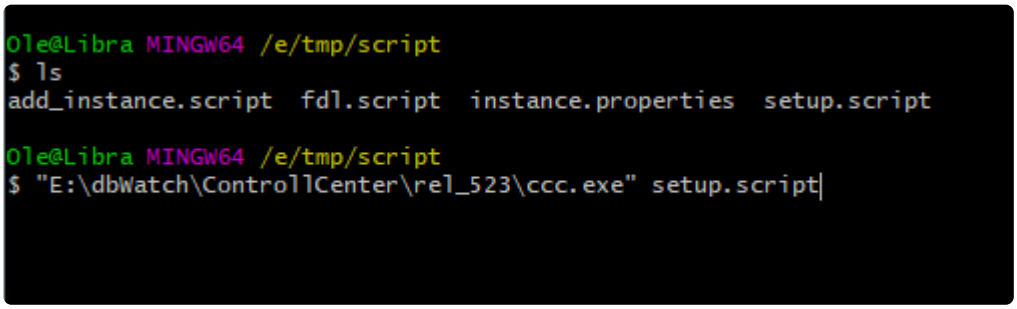
<< Auditing / Setting up a CCC node >>

Setting up a CCC node

To setup a ccc node you first need to download the ccc scripts and extract them in a catalog that you want to be the ccc workarea.

In the following example we have downloaded the scripts to “E:/tmp/scripts” and our Control Center installation is located in “E:/dbWatch/ControllCenter/rel_523”.
All ccc commands take (at minumiun) a script file as an argument.

First we run the setup.script as follows:



This initializes a dbWatch node that attempts to connect to localhost:7100 and tries to join the domain of that node.

The script has tree optional arguments.
server – specifies another host:port to connect to.
domainName – specifies a particular domain to join (relevant if the server has several domains)
name – specifies the node displayname (defaults to “ccc(hostname)”

So the command could also be:

```
"E:\dbWatch\ControllCenter\rel_523\ccc.exe" setup.script server=192.168.3.5:71
00 name="My Test CCC node" domainName=demo.dbwatch.com
```

If the connection attempt is successfull, the result will be similar to the following.

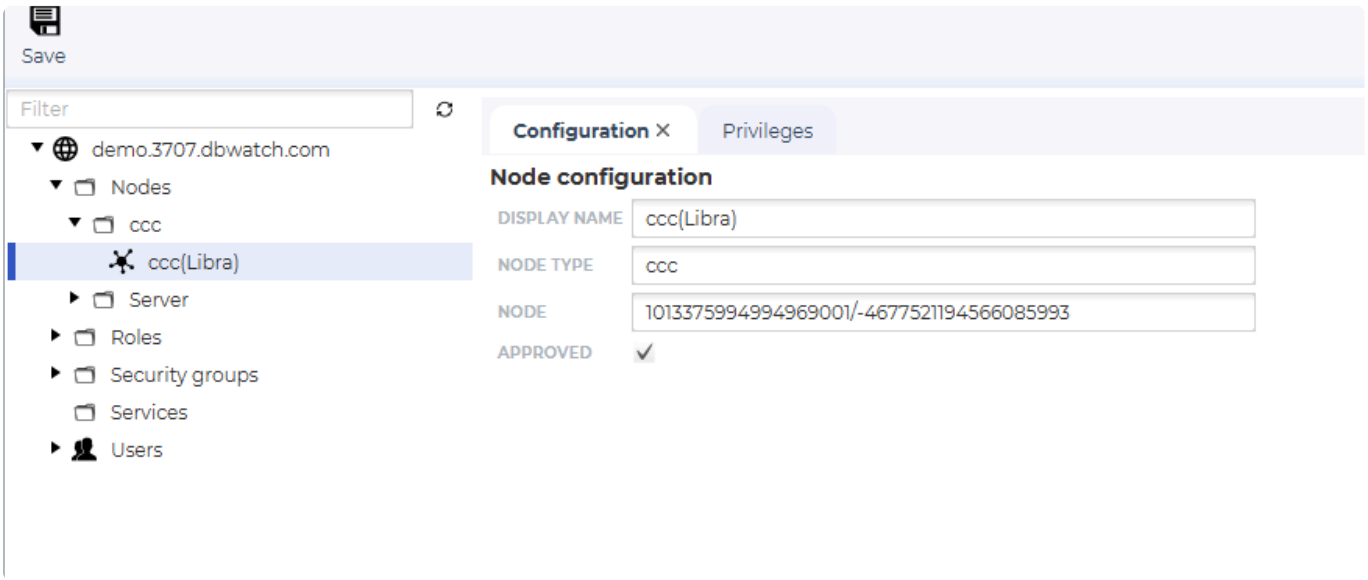
```
Ole@Libra MINGW64 /e/tmp/script
$ "E:\dbWatch\ControllCenter\rel_523\ccc.exe" setup.script
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1667227479266 / no.dbwatch.modes.ScriptRunner / main
Node address {1013375994994969001,-4677521194566085993}
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
no.dbwatch.monitoringtabs:monitoringtabs:1  -  empty dependent_services.json
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(join)
## - setup.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - setup.script.1 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
getReadyFor(localhost:7100)
**** establish (localhost:7100) ****
**** done (localhost:7100) ****
getReadyFor(localhost:7100) encryption ok
getReadyFor(localhost:7100) got domains: demo.3707.dbwatch.com (1)
getReadyFor(localhost:7100) got layout for: demo.3707.dbwatch.com
Unable to get certificate
done=true exitCode=0

Ole@Libra MINGW64 /e/tmp/script
$
```

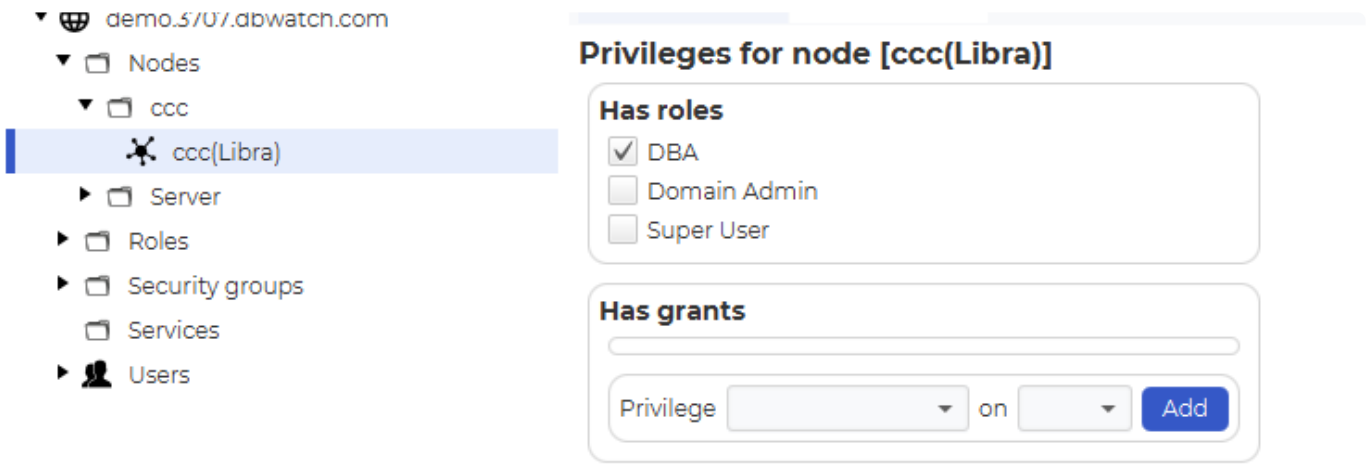
Notice the “Unable to get certificate” message.

This is because the node has not yet been given any privileges in the domain.

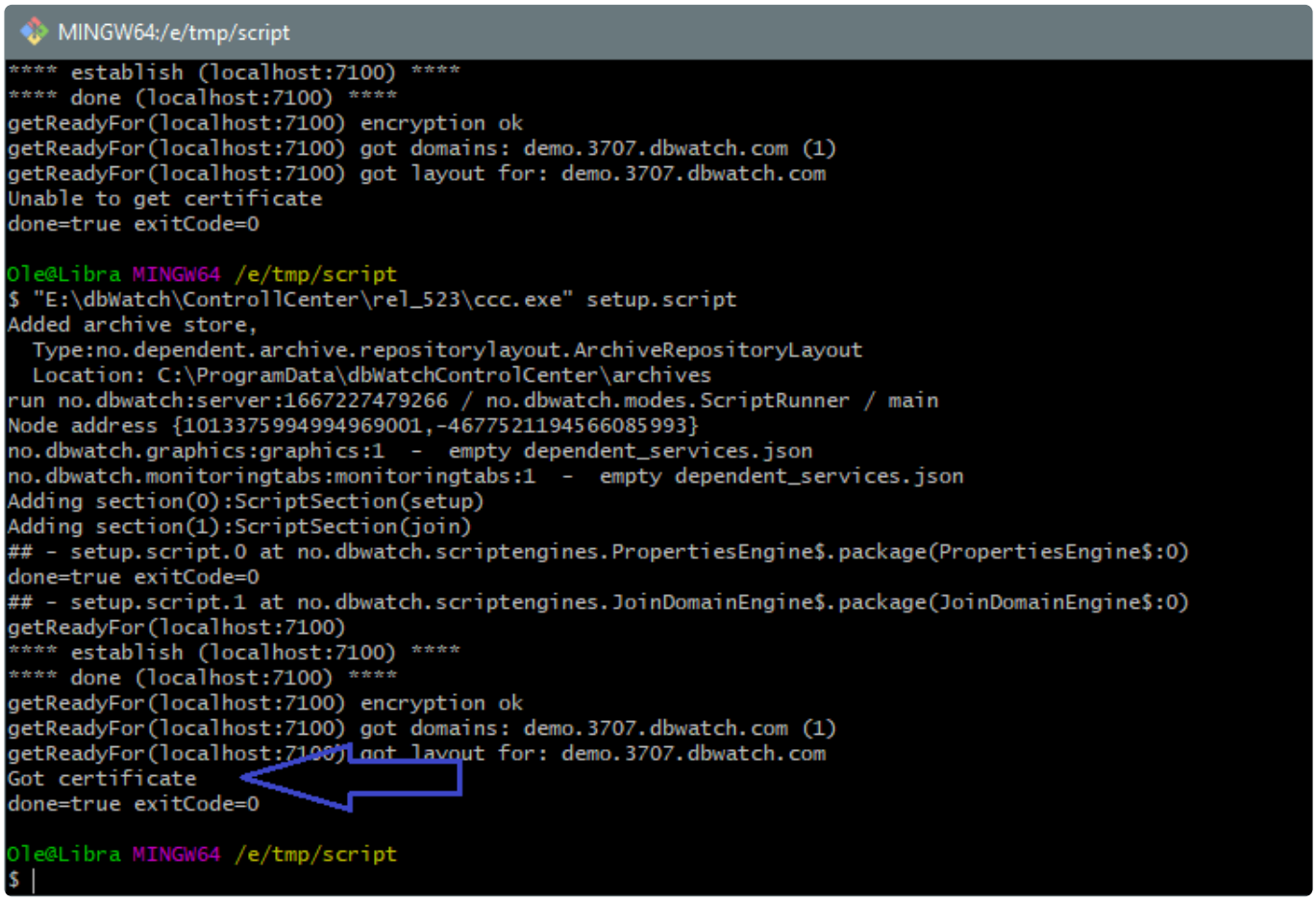
To give it privileges, go to the “Domain Configuration” view in the dbWatch Monitor and locate “ccc” under Nodes. Here the ccc node will have appeared. In our example “ccc (Libra)”. (Libra is the host name of the machine.

Check the “Approved” box, and give it the appropriate privileges.





Now run the setup script again.



Notice that we now have a certificate, and we are ready to run ccc commands.

Cryptographic keys, certificates and config files are stored under “[User home]/.dbwatch/.script_runner/”

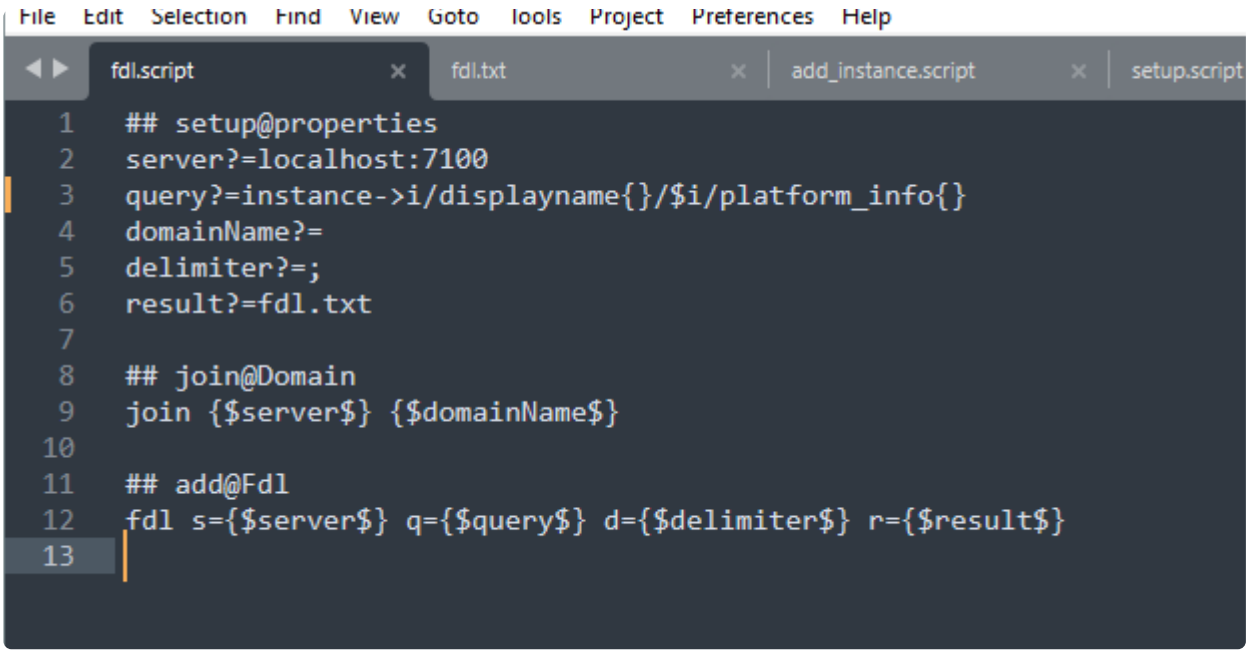
<< Control Center Commandline / FDL with CCC >>

FDL with CCC

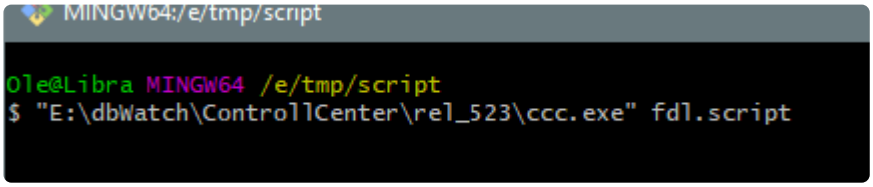
Prerequisites

Before running fdl through CCC you need to go through the steps described in Setting up a CCC node

First we need to specify the fdl query to execute. This is done by editing the “fdl.script” file.
The query is specified after the **query?=** text.



Then run the script with the following command:



The script has four optional arguments.
server – specifies another host:port to connect to.
domainName – specifies a particular domain (relevant if the server has several domains)
delimiter – specifies the delimiter to use in the result file
result – specifies the file to put the result in (default is fdl.txt)

```
$ "E:\dbWatch\ControlCenter\rel_523\ccc.exe" fdl.script
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1667227479266 / no.dbwatch.modes.ScriptRunner / main
Node address {1013375994994969001,-4677521194566085993}
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
no.dbwatch.monitoringtabs:monitoringtabs:1  -  empty dependent_services.json
Identity: demo.3707.dbwatch.com\1013375994994969001/-4677521194566085993
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(join)
Adding section(2):ScriptSection(add)
## - fdl.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - fdl.script.1 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
getReadyFor(localhost:7100)
**** establish (localhost:7100) ****
**** done (localhost:7100) ****
getReadyFor(localhost:7100) encryption ok
getReadyFor(localhost:7100) got domains: demo.3707.dbwatch.com (1)
getReadyFor(localhost:7100) got layout for: demo.3707.dbwatch.com
Already have certificate
done=true exitCode=0
## - fdl.script.2 at no.dbwatch.scriptengines.FdlEngine$.package(FdlEngine$:0)
getReadyFor(localhost:7100)
getReadyFor(localhost:7100) encryption ok
getReadyFor(localhost:7100) got domains: demo.3707.dbwatch.com (1)
getReadyFor(localhost:7100) got layout for: demo.3707.dbwatch.com
running query instance->i/displayname{}/$i/platform_info{}
done=true exitCode=0

Ole@Libra MINGW64 /e/tmp/script
$
```

If successful the result will be available in the result file (fdl.txt)

[<< Setting up a CCC node](#) / [Add instance with CCC >>](#)

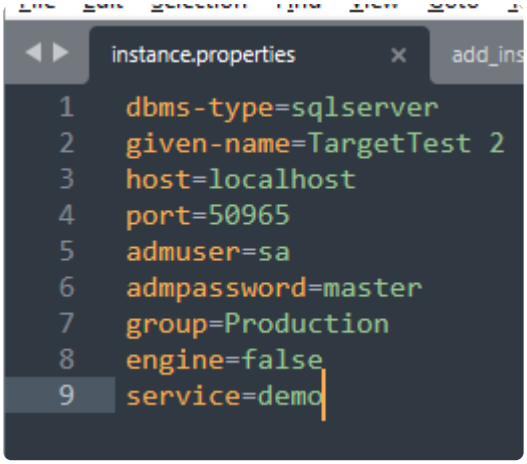
Add instance with CCC

Prerequisites

Before adding instances through CCC you need to go through the steps described in Setting up a CCC node

The properties for the instance to add are defined in a file (default is “instance.properties”). The syntax is key/value pairs seperated with “=”.
The available tags are the same as for csv import.

Example:



The following executes the add instance command:

```
Ole@Libra MINGW64 /e/tmp/script
$ "E:\dbWatch\ControlCenter\rel_523\ccc.exe" add_instance.script definition=instance.properties
```

The script has two optional arguments.
server – specifies another host:port to connect to.
domainName – specifies a particular domain (relevant if the server has several domains)

Result

```
Ole@Libra MINGW64 /e/tmp/script
$ "E:\dbWatch\ControlCenter\rel_523\ccc.exe" add_instance.script definition=instance.properties
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1667227479266 / no.dbwatch.modes.ScriptRunner / main
Node address {1013375994994969001,-4677521194566085993}
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
no.dbwatch.monitoringtabs:monitoringtabs:1  -  empty dependent_services.json
Identity: demo.3707.dbwatch.com\1013375994994969001/-4677521194566085993
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(join)
Adding section(2):ScriptSection(add)
## - add_instance.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - add_instance.script.1 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
getReadyFor(localhost:7100)
**** establish (localhost:7100) ****
**** done (localhost:7100) ****
getReadyFor(localhost:7100) encryption ok
getReadyFor(localhost:7100) got domains: demo.3707.dbwatch.com (1)
getReadyFor(localhost:7100) got layout for: demo.3707.dbwatch.com
done=true exitCode=0
## - add_instance.script.2 at no.dbwatch.scriptengines.TargetEngine$.package(TargetEngine$:0)
getReadyFor(localhost:7100)
getReadyFor(localhost:7100) encryption ok
getReadyFor(localhost:7100) got domains: demo.3707.dbwatch.com (1)
getReadyFor(localhost:7100) got layout for: demo.3707.dbwatch.com
Installing instance
done=true exitCode=0

Ole@Libra MINGW64 /e/tmp/script
```

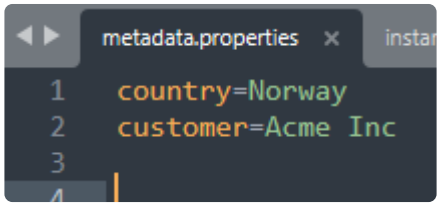
[<< FDL with CCC](#) / [Set metadata with CCC >>](#)

Set metadata with CCC

Prerequisites

Before settings metadata through CCC you need to go through the steps described in Setting up a CCC node

Metadata is defined in a file (default “metadata.properties”)




```
Ole@Libra MINGW64 /e/tmp/script
$ "E\:dbWatch\ControlCenter\rel_523/ccc.exe" metadata.script instanceName="Test Instance"
```

<< Add instance with CCC / Install jobs with CCC >>

Install jobs with CCC

Prerequisites

Before installing jobs through CCC you need to go through the steps described in Setting up a CCC node

```
Ole@Libra MINGW64 /e/tmp/script
$ "E\:dbWatch\ControllCenter\rel_523/ccc.exe" jobs.scrip instanceName="Test Instance" job=com.dbwatch.job:instance_noschema_startup_time
```

<< Set metadata with CCC / Get instance configuration with CCC >>

Get instance configuration with CCC

Prerequisites

Before you can get instance configuration through CCC you need to go through the steps described in Setting up a CCC node

Getting instance configuration

get_instance.script has the following input parameters:
server – specify server IP and port, by default localhost:7100
domainName – specify domain name, relevant only if the server has multiple domains, if it has only one, it will default to that one.
instanceName – instance name to get configuration for
result – output XML file for configuration, default instance.xml

Example:
ccc \script\get_instance.script instanceName="joiol" result=joiol_instance_configuration.xml

```
c:\Program Files\dbwatchControlCenter>ccc \script\get_instance.script instanceName="joiol" result=joiol_instance_configuration.xml
Picked up JAVA_TOOL_OPTIONS: -Djava.rmi.server.hostname=192.168.6.245
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1670837304464 / no.dbwatch.modes.ScriptRunner / main
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(setup)
Adding section(2):ScriptSection(join)
Adding section(3):ScriptSection(get)
## - \script\get_instance.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\get_instance.script.1 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\get_instance.script.2 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
Certificate OK
done=true exitCode=0
## - \script\get_instance.script.3 at no.dbwatch.scriptengines.TargetEngine$.package(TargetEngine$:0)
Output written to .\joiol_instance_configuration.xml
done=true exitCode=0

c:\Program Files\dbwatchControlCenter>
```

<< Install jobs with CCC / Register instance configuration with CCC >>

Register instance configuration with CCC

Prerequisites

Before you can get instance configuration through CCC you need to go through the steps described in [Setting up a CCC node]

Registering instance configuration

register_instance.script has the following input parameters:
server – specify server IP and port, by default localhost:7100
domainName – specify domain name, relevant only if the server has multiple domains, if it has only one, it will default to that one
definition – input XML file for configuration, default instance.xml

Example:
ccc \script\register_instance.script definition=joiol_instance_configuration.xml

```
c:\Program Files\dbwatchControlCenter>ccc \script\register_instance.script definition=joiol_instance_configuration.xml
Picked up JAVA_TOOL_OPTIONS: -Djava.rmi.server.hostname=192.168.6.245
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1670837304464 / no.dbwatch.modes.ScriptRunner / main
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(join)
Adding section(2):ScriptSection(register)
## - \script\register_instance.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\register_instance.script.1 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
Certificate OK
done=true exitCode=0
## - \script\register_instance.script.2 at no.dbwatch.scriptengines.TargetEngine$.package(TargetEngine$:0)
registerTarget localhost:7100, joiol_instance_configuration.xml
reply: DbwProgressMessage(registerTarget done,true,true,ConversationId(849165899,-2031323844))
registerTarget done
done=true exitCode=0

c:\Program Files\dbwatchControlCenter>
```

[<< Get instance configuration with CCC] / [Instance action script with CCC >>]

Instance action script with CCC

Instance action script

The instance action script (instance_action.script) can do the following commands:

Connect

You can connect an already registered and disconnected instance using the connect action. It is the same as what happens if you right-click on an instance in the GUI, and select “Connect”.
Example:
ccc.exe instance_action.script action=connect instanceName=‘iotra’

```
c:\Program Files\dbwatchControlCenter>ccc \script\instance_action.script action=connect instanceName="irith"
Picked up JAVA_TOOL_OPTIONS: -Djava.rmi.server.hostname=192.168.6.245
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1671021252795 / no.dbwatch.modes.ScriptRunner / main
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(setup)
Adding section(2):ScriptSection(join)
Adding section(3):ScriptSection(action)
## - \script\instance_action.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\instance_action.script.1 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\instance_action.script.2 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
Certificate OK
done=true exitCode=0
## - \script\instance_action.script.3 at no.dbwatch.scriptengines.TargetEngine$.package(TargetEngine$:0)
action [connect] localhost:7100 [instance[name='irith']]
targetaction done
done=true exitCode=0
```

Disconnect

You can disconnect an already registered and connected instance using the disconnect action. It is the same as what happens if you right-click on an instance in the GUI, and select “Disconnect”.
Example:
ccc.exe instance_action.script action=disconnect instanceName=‘iotra’

```
c:\Program Files\dbwatchControlCenter>ccc \script\instance_action.script action=disconnect instanceName="irith"
Picked up JAVA_TOOL_OPTIONS: -Djava.rmi.server.hostname=192.168.6.245
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1671021252795 / no.dbwatch.modes.ScriptRunner / main
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
Adding section(0):ScriptSection(setup)
Adding section(1):ScriptSection(setup)
Adding section(2):ScriptSection(join)
Adding section(3):ScriptSection(action)
## - \script\instance_action.script.0 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\instance_action.script.1 at no.dbwatch.scriptengines.PropertiesEngine$.package(PropertiesEngine$:0)
done=true exitCode=0
## - \script\instance_action.script.2 at no.dbwatch.scriptengines.JoinDomainEngine$.package(JoinDomainEngine$:0)
Certificate OK
done=true exitCode=0
## - \script\instance_action.script.3 at no.dbwatch.scriptengines.TargetEngine$.package(TargetEngine$:0)
action [disconnect] localhost:7100 [instance[name='irith']]
targetaction done
done=true exitCode=0
```

Start/stop webservice with CCC

> Not available in general release. You need the 22.03.2024 build or later.

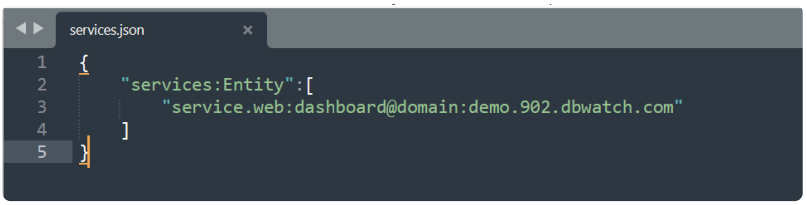
> Note: The term service in this context is an internal service in dbWatch, not a Windows service.

Prerequisites

First you need to go through the steps described in Setting up a CCC node

You also need to download the service scripts.

Find the id of the service you want to manage. This is located in the file "./config/services.json".



In our example the id is **service.web:dashboard@domain:demo.902.dbwatch.com**

Get status

To get the status of the service, use the service_status.script and supply the id as serviceName.

```
MINGW64 /c/Development/ccc_script
$ "C:\Program Files\dbWatchControlCenter\rel_765\ccc.exe" service_status.script server="localhost:7100" serviceName="service.web:dashboard@domain:demo.902.dbwatch.com"
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1711111292476 / no.dbwatch.modes.ScriptRunner / main
defineClass no.dbwatch.metadata.action.Scheduler$
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
done=true exitCode=0
Certificate OK
done=true exitCode=0
serviceStatus [service.web:dashboard@domain:demo.902.dbwatch.com] localhost:7100
dashboard Running
done=true exitCode=0
```

As you can see, ccc is a bit chatty, but the status is returned as the service name followed by "Running" or "Stopped".
In our example **dashboard Running**

Stop the service

To stop the service, use the service_stop.script and supply the id as serviceName.

```
MINGW64 /c/Development/ccc_script
$ "C:\Program Files\dbWatchControlCenter\rel_765\ccc.exe" service_stop.script server="localhost:7100" serviceName="service.web:dashboard@domain:demo.902.dbwatch.com"
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1711111292476 / no.dbwatch.modes.ScriptRunner / main
defineClass no.dbwatch.metadata.action.Scheduler$
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
done=true exitCode=0
Certificate OK
done=true exitCode=0
stopService [service.web:dashboard@domain:demo.902.dbwatch.com] localhost:7100
dashboard stop OK
done=true exitCode=0
```

Checking the status again, we see that the service is now stopped.

```
MINGW64 /c/Development/ccc_script
$ "C:\Program Files\dbWatchControlCenter\rel_765\ccc.exe" service_status.script server="localhost:7100" serviceName="service.web:dashboard@domain:demo.902.dbwatch.com"
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1711111292476 / no.dbwatch.modes.ScriptRunner / main
defineClass no.dbwatch.metadata.action.Scheduler$
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
done=true exitCode=0
Certificate OK
done=true exitCode=0
serviceStatus [service.web:dashboard@domain:demo.902.dbwatch.com] localhost:7100
dashboard Stopped
done=true exitCode=0
```

Start the service

To stop the service, use the service_start.script and supply the id as serviceName.

```
MINGW64 /c/Development/ccc_script
$ "C:\Program Files\dbWatchControlCenter\rel_765\ccc.exe" service_start.script server="localhost:7100" serviceName="service.web:dashboard@domain:demo.902.dbwatch.com"
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1711111292476 / no.dbwatch.modes.ScriptRunner / main
defineClass no.dbwatch.metadata.action.Scheduler$
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
done=true exitCode=0
Certificate OK
done=true exitCode=0
startService [service.web:dashboard@domain:demo.902.dbwatch.com] localhost:7100
dashboard start OK
done=true exitCode=0
```

Checking the status again, we see that the service is now started.

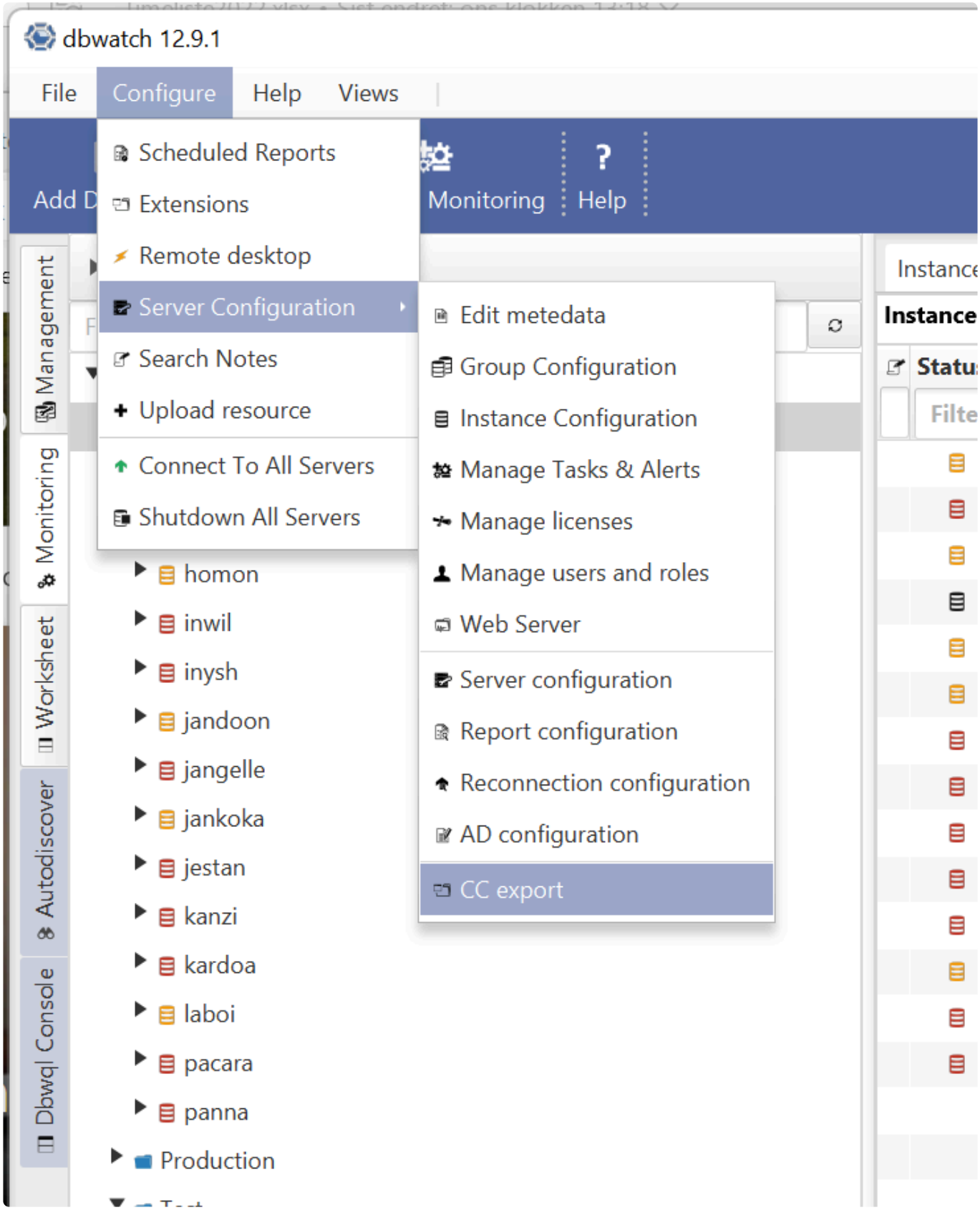
```
MINGW64 /c/Development/ccc_script
$ "C:\Program Files\dbWatchControlCenter\rel_765\ccc.exe" service_status.script server="localhost:7100" serviceName="service.web:dashboard@domain:demo.902.dbwatch.com"
Added archive store,
  Type:no.dependent.archive.repositorylayout.ArchiveRepositoryLayout
  Location: C:\ProgramData\dbWatchControlCenter\archives
run no.dbwatch:server:1711111292476 / no.dbwatch.modes.ScriptRunner / main
defineClass no.dbwatch.metadata.action.Scheduler$
no.dbwatch.graphics:graphics:1  -  empty dependent_services.json
done=true exitCode=0
Certificate OK
done=true exitCode=0
serviceStatus [service.web:dashboard@domain:demo.902.dbwatch.com] localhost:7100
dashboard Running
done=true exitCode=0
```

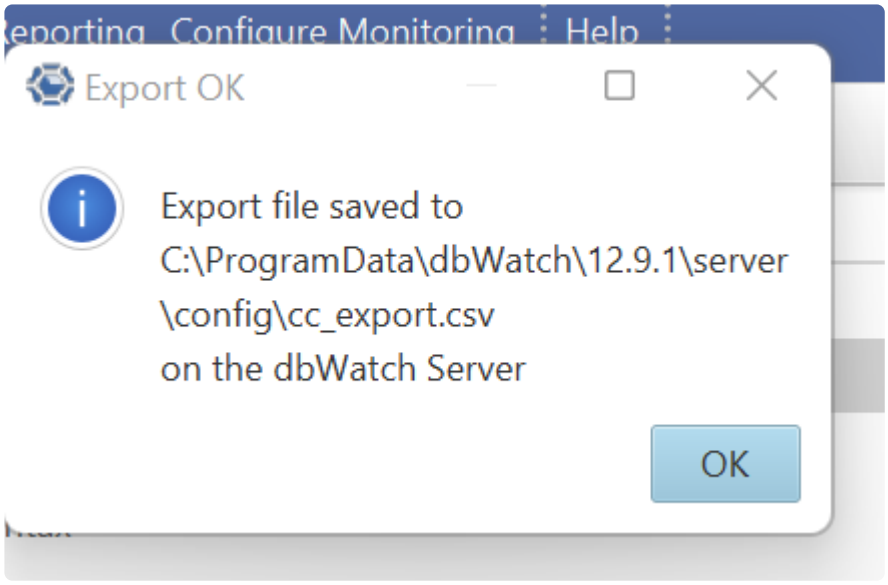
Exporting configuration from 12 to CC

Exporting database instance configuration from dbWatch 12 to CC

A version of 12.9.1 or above is needed for this.
The idea is that you can export the instance configuration from a dbWatch 12.9.x installation to install the same instances in dbWatch Control Center, as easy as possible.

Click on Configure->Server Configuration -> CC export to export the configuration. This will create a csv file that can be imported into dbWatch Control Center.

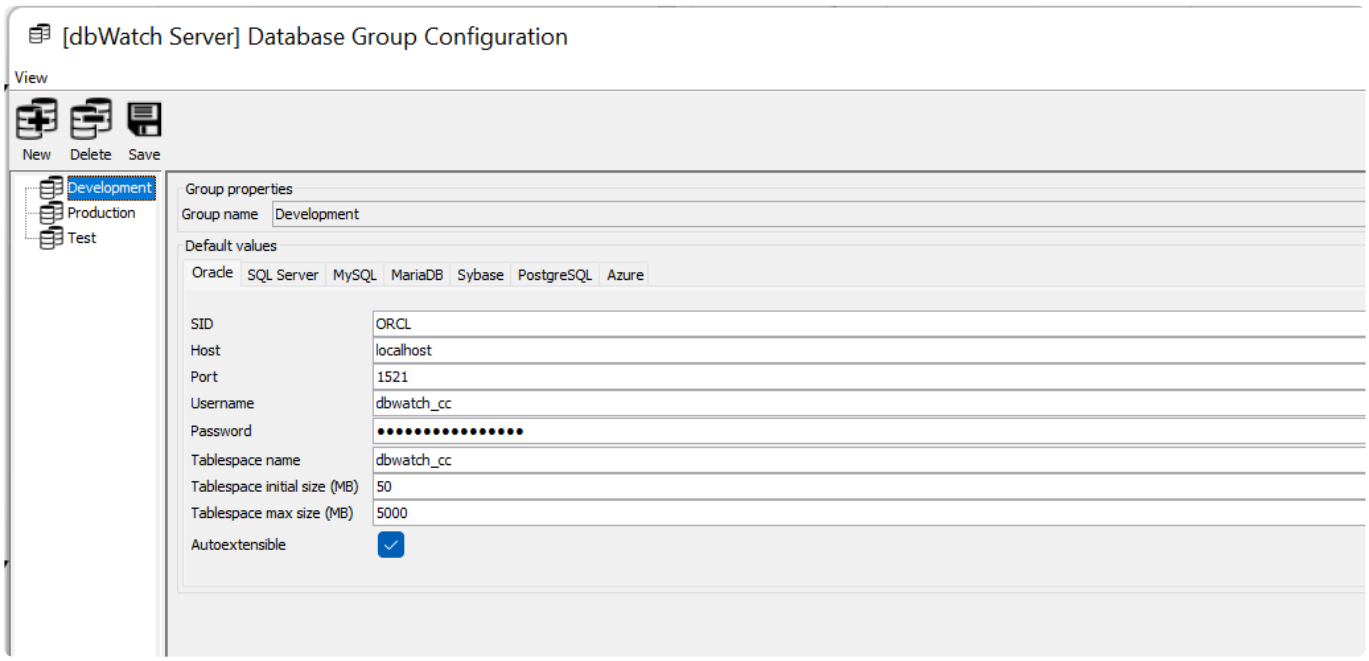

dbwatch 12.9.1
File
Configure
Help
Views
Scheduled Reports
Extensions
Remote desktop
Server Configuration
Search Notes
Upload resource
Connect To All Servers
Shutdown All Servers
Edit metedata
Group Configuration
Instance Configuration
Manage Tasks & Alerts
Manage licenses
Manage users and roles
Web Server
Server configuration
Report configuration
Reconnection configuration
AD configuration
CC export
Monitoring
Help
Management
Monitoring
Worksheet
Autodiscover
Dbwql Console
homon
inwil
inysh
jandoon
jangelle
jankoka
jestan
kanzi
kardoa
laboi
pacara
panna
Production
Instance
Instance



You will get a cc_export.csv file, that needs to be moved to the dbWatch CC client.

It should look like this:

Before importing this file to dbWatch Control Center, you might want to edit it for “service:” specification to match the licenses you have for dbWatch Control Center :
So instead of “service:demo”, you could change it to “service:cluster_oracle;oracle_performance_licenced;oracle_maintenance_licenced” for an other Oracle RAC installation including the performance module (each license separated with “;” .
Also before importing the file in Control Center, consider the “group” specification according to your “Database Group Configuration” in Control Center :



Import of Oracle cluster does not work well with import, and should be handled manually. Also, import of Postgres configuration will not be able to create a new database for the dbWatch schema, but works with already existing databases.

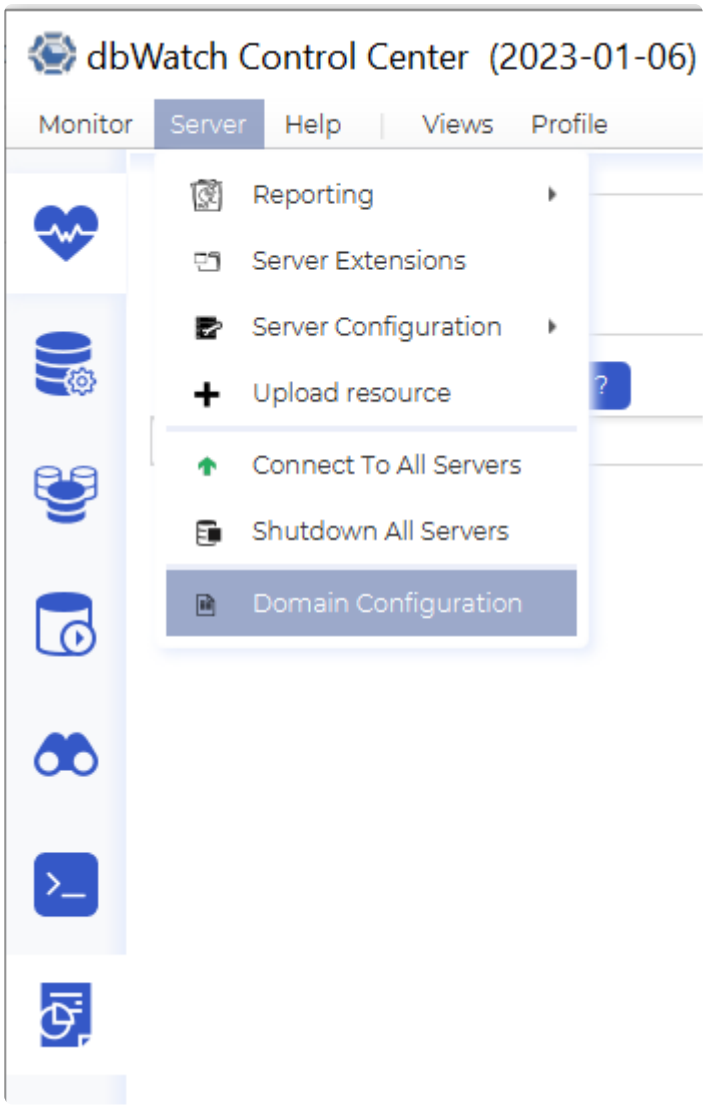
Internal Control Center firewall

The internal firewall

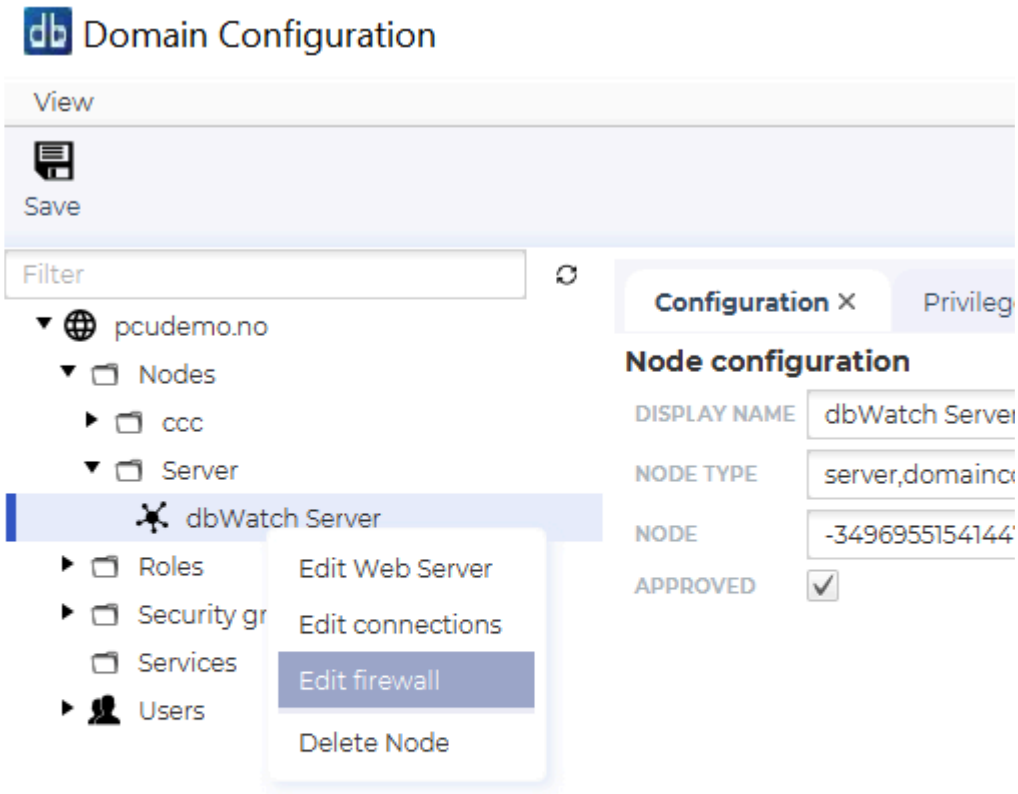
Mostly used for advanced configurations, where nodes are placed in locations facing internet traffic, there is a built-in firewall in Control Center.

You access the firewall configuration in the domain configuration dialog for each node.

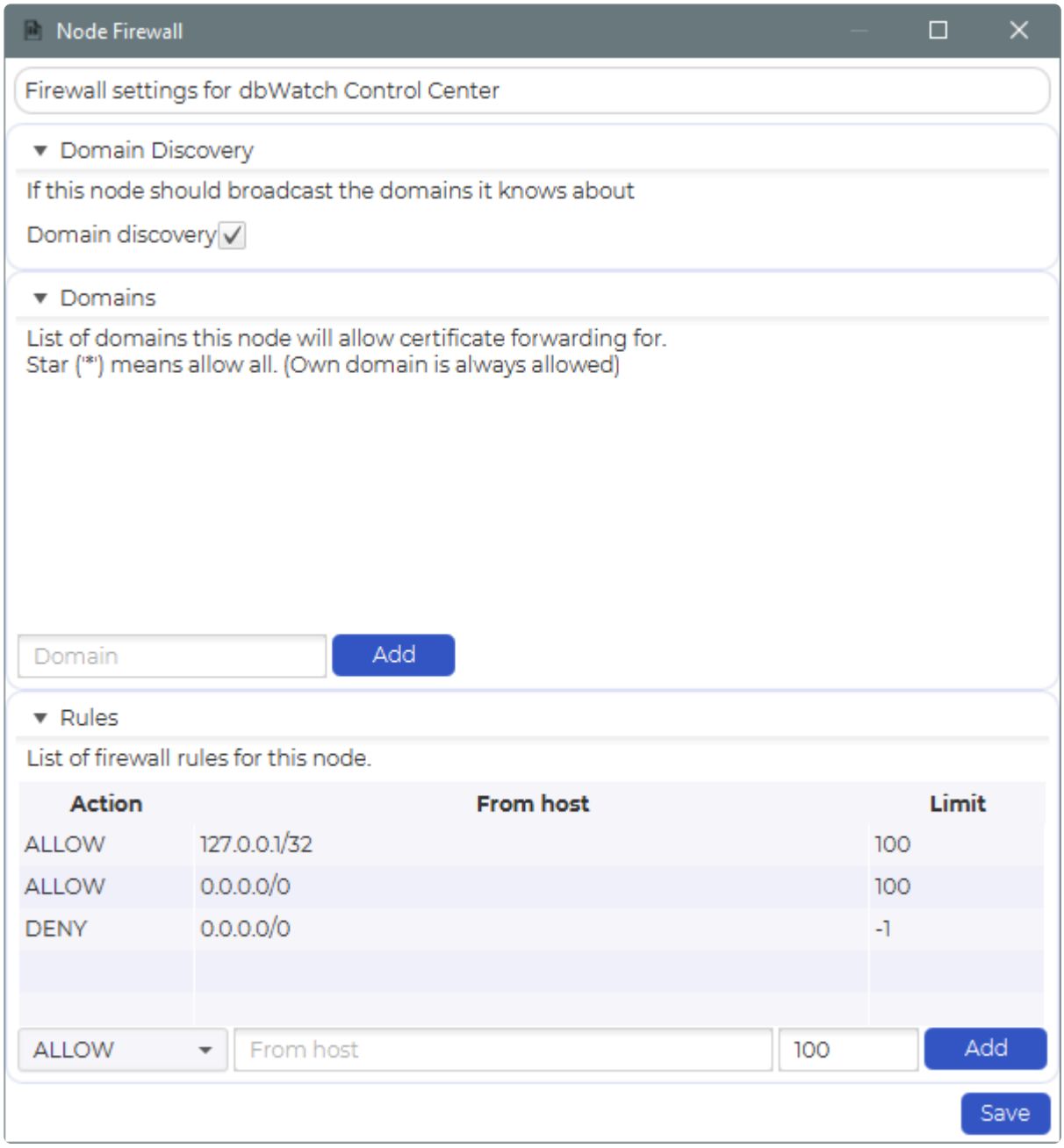
Opening domain configuration dialog:



Starting the firewall configuration GUI.



There are three sections to the firewall.



Domain Discovery

When connecting a dbWatch Monitor to a node, it will by default provide a list of its known domains and you can select wich one to (attempt to) join. If Domain Discovery is turned off, the list will not be provided and the user has to enter the domain manually.

Domains

Specifies what domains this node will forward certificates for. This is typically used for a node that works as a cloud router.

Rules

The rules section allows you to add IP addresses and ranges to allow or deny. The list starts at the top

and either allows or denies based on the rules.

The default setup is to have three rules:

Action	From host	Limit
ALLOW	127.0.0.1/32	100
ALLOW	0.0.0.0/0	100
DENY	0.0.0.0/0	-1

The first rule is explicit allowing connections from IP 127.0.0.1 (only this IP), limiting to 100 max connections.
The second rule is allowing all IP addresses, limiting to 100 max connections.
The third rule is denying all connections not hit by any of the above rules (which in this case would be none)

The idea is to replace the 0.0.0.0/0 rule with either a specific with an address in the CIDR address notation. (192.168.0.0/24 for 192.168.0.0-192.168.0.255, 192.168.0.3/32 of only the 192.168.0.3 IP etc)

Troubleshooting Guide

In this section, we'll discuss about:

- dbWatch Backup
- Restoring Configuration Backup
- Fault Finding Guide
- Issues and Troubleshooting
- Definition Mapping

<< Internal Control Center firewall / Backup of dbWatch >>

Backup of dbWatch

When we think about backup of dbWatch, this involves several different parts of the software and installation:

dbWatch Server and configuration files

dbWatch Server places its configuration files and workarea in “C:\ProgramData\dbWatchControlCenter\” by default on windows (can be changed in the installation wizard)

A zip of this directory is sufficient for backing up the dbWatch Server installation.
There are no registry keys for which the dbWatch Server depends on, but if you move the zip file to a new server, it is important that on Windows you add a new service to start dbWatch.

While we do recommend our customers do a complete backup of this directory, but the most important sub catalogs are
C:\ProgramData\dbWatchControlCenter\config and **C:\ProgramData\dbWatchControlCenter\crypto**

dbWatch Monitor and config files

dbWatch Monitor is installed in a directory which is normally /usr/local/dbWatch/[version] on Unix/Linux and C:\Program Files (x86)\dbWatch\[version] on Windows.
A zip of this directory is sufficient for backing up the dbWatch Monitor installation. There are no registry keys that the dbWatch Monitor depends on.

While we do recommend our customers do a complete backup of the installation directory, a few configuration files are essential and unique for your installation. The most important one is monitor.xml. This contains the information about the dbWatch Servers you connect to. This is located in the dbWatch Server root directory.

dbWatch Schema in the databases

dbWatch Schema is usually called “dbwatch” and can be dumped with a command such as exp/imp on Oracle. It is also normally backed up as part of the normal database backup. In the dbWatch Schema all information on which tasks are installed, their status and historic information is kept. If you lose this, you can remove the database from dbWatch, drop the dbWatch user if it exists from the database and then add the database in dbWatch again. You will be without some historic information, but you get the database monitored by dbWatch again, and that is what is most important.

[<< Troubleshooting Guide] / [Adding MySQL Instance with Super Privilege Error >>]

Adding MySQL Instance with Super Privilege Error

MySQL Super Privilege Error

When adding an instance for MySQL 5.0 or over, you may encounter the following error:

This error happens when dbWatch Control Center is creating a user. This error appears indicating that the user has no super privilege rights.

There are two methods in dealing with the problem. First, we'll do it using the commandline. Second, we'll do it using any management tool available.

Command Line

Open your windows command line and go to your MySQL directory. Open MySQL and input the user name and password. This should open up MySQL.
Afterwards, input the following query:

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

```
C:\Program Files\MySQL\MySQL Server 8.0>mysql -u root -p p@ssw0rd
'mysql' is not recognized as an internal or external command,
operable program or batch file.

C:\Program Files\MySQL\MySQL Server 8.0>cd bin

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p p@ssw0rd
Enter password: ********
ERROR 1049 (42000): Unknown database 'p@ssw0rd'

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1188
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SET GLOBAL log_bin_trust_function_creators = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

This sets the variable log_bin_trust_function_creators to ON.

Then exit the MySQL but input ‘exit’.

Using a management tool

Open your SQL workbench or any management tool you are using.

```
SHOW VARIABLES LIKE 'log_bin_trust_function_creators';
SET GLOBAL log_bin_trust_function_creators = 0; -- OFF


SET GLOBAL log_bin_trust_function_creators = 1; -- ON
```

Same as before, you want to input the same query:

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

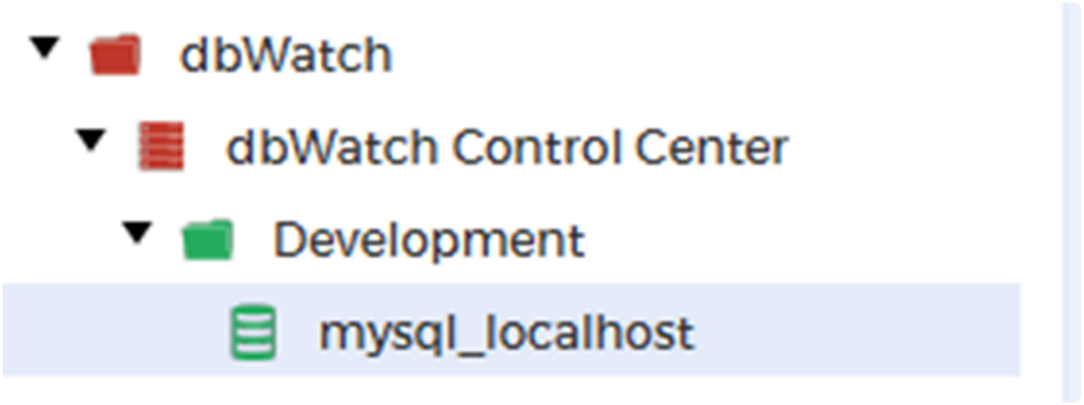
But this time, we'll add a way to see the variable:

```
SHOW VARIABLES LIKE 'log_bin_trust_function_creators';
```

To turn it off, simple exceute the following query: SET GLOBAL log_bin_trust_function_creators = 0;

Add Instance Successful

Once you completed one of the previous steps, you should be able to successfully add a MySQL Instance. Add that MySQL Instance and you should be able to see the following:



<< Backup of dbWatch / Restore configuration backup >>

Restore configuration backup

The configuration files in dbWatch are continously backed up using git.

Reverting to an old backup

Step one

If you don’t have git tools installed, you can download them here

Step two

After stopping the dbWatch Server, use a command prompt (that understands git commands) and, navigate to the config catalog.
By default this is “C:/ProgramData/dbWatch/[version]/server/config” on Windows.
Be sure to do this as an Administrator (or some other user that has write privileges on this catalog)

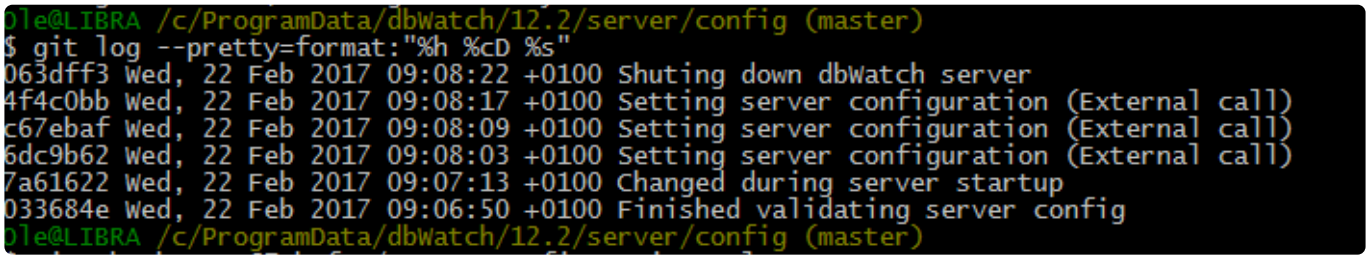
Step three

Take a backup of the entire config catalog (it’s never a mistake to have a backup)

Then type the following git command to see the git entries.

```
git log --pretty=format:"%h %cD %s"
```

You should see something like the following



Step four

Find the entry you want to revert to. This can involve a bit of guessing, but based on the time column and the descriptions, you should be able to find a line that corresponds to the version you want to revert to. If you are unable to figure out what entry you want, there are also graphical tools (like git gui) that can visualize the differences between the versions better.

Step five

The start of the line you want to revert to is a series of numbers and letters called a hash
Type the command “git checkout [hash] ./server_configuration.xml”, f.ex:

```
git checkout c67ebaf ./server_configuration.xml
```

Step six

Commit the new version to git by typing “git commit -m”[message]”, f.ex:

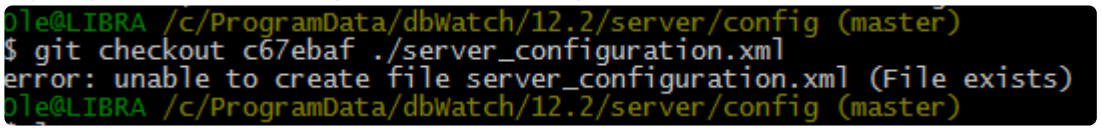
```
git commit -m"got backup"
```

Step seven

Restart the dbWatch Server

Troubleshooting

If you get an error message like the following:



Then the most likely cause is that your user does not have write privileges on the catalog.

<< Adding MySQL Instance with Super Privilege Error / Issues and troubleshooting >>

Issues and troubleshooting

High CPU usage on database

Occasionally when investigating resource-consuming SQL statements running on your system, you may find that some of them belong to the dbWatch application.

For example, a SQL statement that has the highest value of spent CPU time. In most cases, dbWatch executes the same SQL statements many times over a long time period so the accumulated CPU time becomes high, and at first glance this might be assumed to indicate a problem. However, if you calculate how much CPU time is consumed during a day, you will typically find that it is a matter of minutes.

<< Restore configuration backup / Definition Mapping >>

Definition Mapping

dbWatch 12	dbWatch Control Center	Definition	Changes
Tasks and Alerts	**Jobs**	Tasks provide statistics and growth rates for your database, which allows for better planning and performance analysis of how your system is behaving. Alerts provide alarms and warning that enables you to react to problems as quickly as possible.	Basically tasks and alerts are just renamed to Jobs. Which still serves the same purpose.
DBWQL (dbWatch Query Language)	**FDL (Farm Data Language)**	FDL is a new and improved query language based on XPath language – you can now query across your database farm even if its across different database platforms. You can perform aggregation and determine the resource usage across your database farm.	FDL is now part of the Farm Management Module.
	Farm Management	Farm management module a new feature wherein users can easily monitor the health of the whole database farm. Displays the summary of their performance, capacity and resource usage.	New feature
Memory Reduction	**AMR**	An upgraded feature that will be included within the farm management module. This helps DBAs automate the memory utilization allocation of their database farm.	Automatic Memory Reduction for SQLServer
AutoDiscover	**Database Discovery**	Database Discovery is now a part of the Farm management module which helps discover all database instances within a network range.	
Standard Installation	**Basic Installation**	No packages or procedures are installed within the monitored instances.	
Advanced Installation	**Standard Installation**	Packages and procedures installed within the monitored instances. Recommended way of monitoring the database instances.	
dbWatch Server	**dbWatch Server**	The nerve center of the dbWatch infrastructure. Connects and manages all dbWatch engines installed within the monitored instances.	
dbWatch Client	**dbWatch Client**	Provide the graphical user interface for the dbWatch user, with a highly intuitive and easy to use structure.	
dbWatch Engines	**dbWatch Engines**	dbWatch uses an intelligent database schema inside the database, called dbWatch Engines, for monitoring each database instance. The dbWatch Engine employs two types of monitoring procedures – dbWatch Alerts for real-	

		time alarms and warnings, and dbWatch Tasks for trend analysis and reporting.	
dbWatch Extensions	**dbWatch Extensions**	Extensions are made for easy and seamless integration with your third party systems management tools and processes.	

The graph:

Name	Definition
Node	A piece of data
Link	data connecting nodes together
The graph	The dbWatch model consists of nodes connected by links
Layer	A piece of the graph that can be turned visible or not
Entity	Addressable nodes in the graph.
Metadata	Data describing Entities in the graph. Metadata can be entities.
Named service	A node in dbWatch representing a service. Is usually connected to several connectors and other named services
Instance	A named service describing a database instance
Driver	The piece of software used by dbWatch to connect to something
Connector	A node representing a configured driver
Connected service	The service that a connector is communicating with

Events

Name	Definition	properties
Event	Something that has happened or is happening.	source, start, end
Sub event	An event can contain other events. These are called sub events.	
Event stream	The sequence of events.	
Issue	A problematic event.	severity, affected nodes
Severity	Used to describe how problematic a given issue is.	
Status	A value describing the health of a node in the graph. Some statuses are aggregates of statuses of sub-nodes	

Jobs:

Name	Definition
Job	A unit of work dbWatch can perform
Trigger	A filter of the event stream that starts the execution of a job
Schedule	A timed trigger
Compatibility	An FDL expression describing where the job can be run
Installable	A job that can be installed and configured in the graph
Status	—Discuss—
Anchor	The root node of the job execution.
Framework	A piece of code installed on the connected service to enable the execution of jobs
Framework job	A job relying on installed framework

Domain

Name	Definition
Node / Processor / Engine / member???	A process managed by the domain.
Domain	A shared configuration used by multiple nodes. A Domain is responsible for identity management and configuration management across its members.
Domain name	The name of a domain. example ‘dbwatch.com’

Resources

Name	Definition
Resource	A single file containing content for dbWatch
Feature	A set of resources used for publication. A feature is published under a domain and signed by a member of said domain.
Archive	An archive containing a set of features. An archive has name and version

User / roles / access

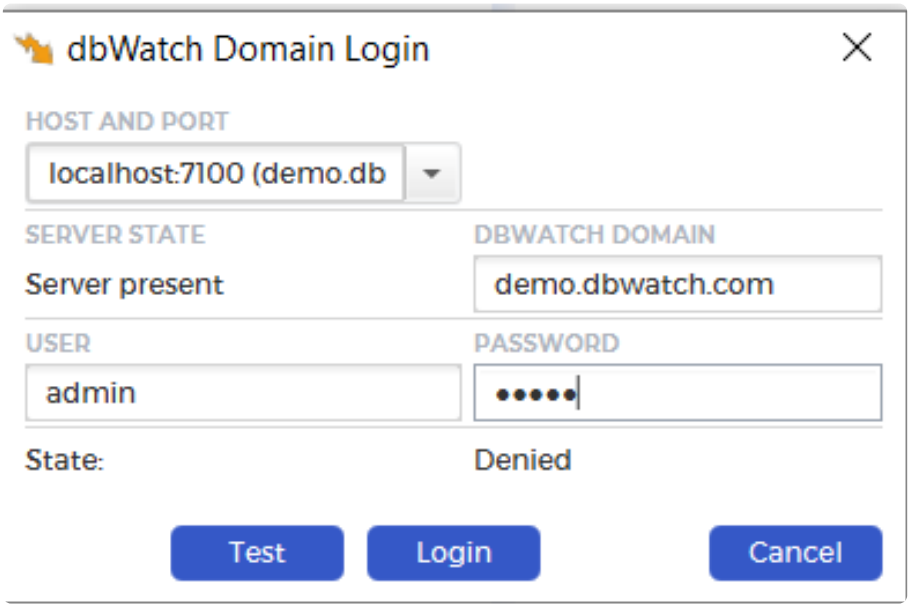
Name	Definition
User	A human using dbWatch
Role	Granting a role to a user results in privileges.

Secure connect

<< Issues and troubleshooting / Resetting dbWatch Control Center Admin Account >>

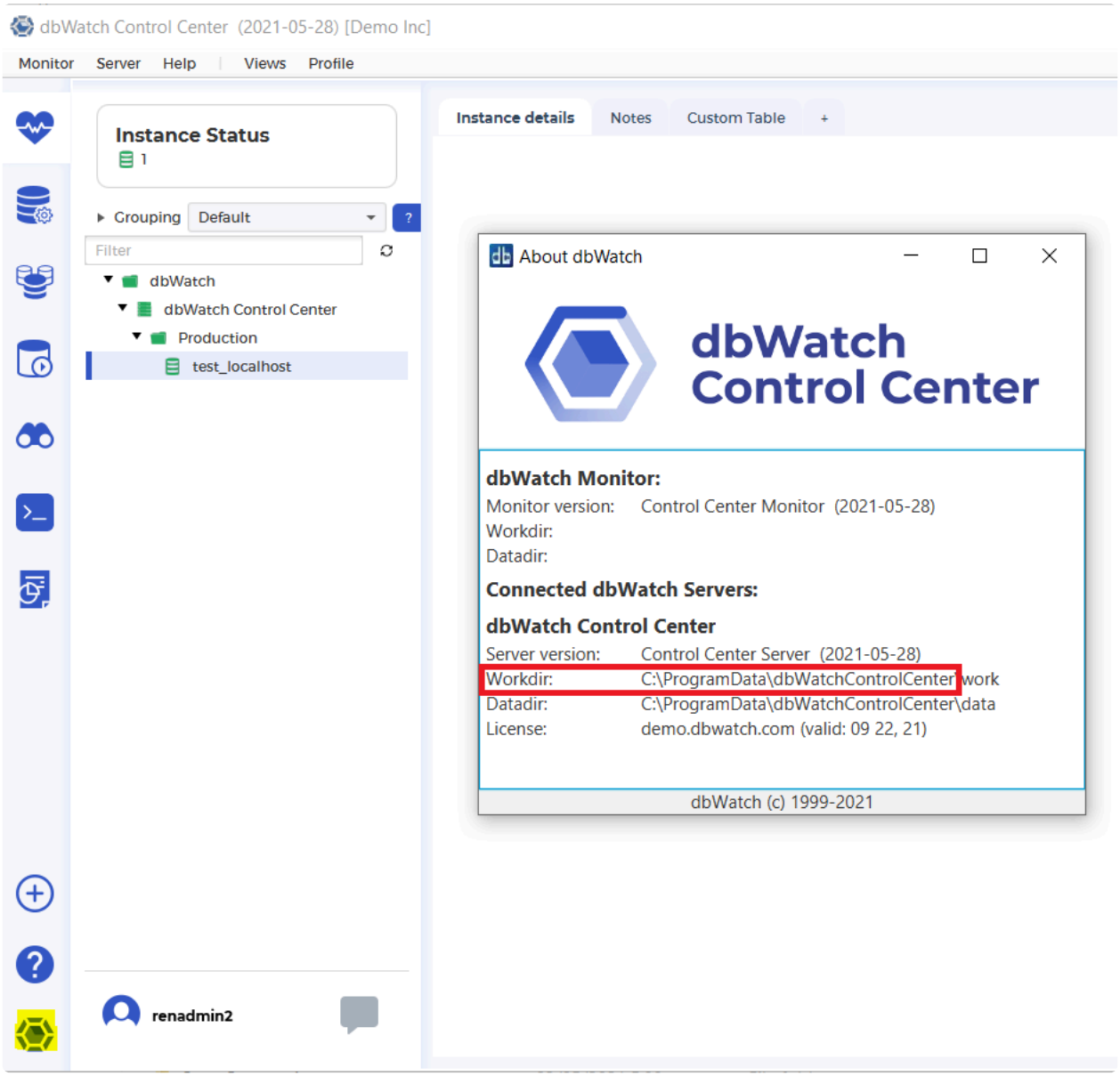
Resetting dbWatch Control Center Admin Account

Before resetting your admin account, you must be familiar with your working directory. By default, dbWatch Control Center will create a directory in ***C:\ProgramData\dbWatchControlCenter*** .If you are not sure, you can open the dbWatch Logo or go to Help > About dbWatch. Another note is you need to also be familiar with your dbWatch Service Name. Make sure you accomplish those things before attempting to do soft reset your dbWatch Control Center.



Resetting admin account

The first thing to do is to close your running dbWatch application. Either exit your application or click on “X”. Afterwards, open “Services” by searching it in your search bar. Stop the dbWatch Control Center Service for the time being. We need this step in order to successfully make alterations to dbWatch’s files.



Once the dbWatch Service has stopped, go to ***C:\ProgramData\dbWatchControlCenter\config\domain*** . Then locate the domain directory for your domain.

In this directory you can find the file authentication.xml. The file is write protected so editing it will require an editor to be opened as administrator.

For the user you want to reset the password entry from:

```
<password algoritm="SHA-512" salt="vGWwBQ==" value="7iaw3Ur350mqGo7jwQrpkj9hiY
B3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1dsUNzLDBMxfqa2Ob1f1ACio/w==" />
```

to

```
<password algoritm="" salt="" value="newpassword" />
```

Example file with user testold and user testnew.

```
<?xml version="1.0" encoding="UTF-8"?>
<authentication-configuration>
        <user name="testnew@domain:test.dbwatch.com">
                <identified_by authority="password">
                        <password algoritm="" salt="" value="newpassword"/>
                        <kerberos value=""/>
                </identified_by>
        </user>
        <user name="testold@domain:test.dbwatch.com">
                <identified_by authority="password">
                        <password algoritm="SHA-512" salt="vGWwBQ==" value="7i
aw3Ur350mqGo7jwQrpkj9hiYB3Lkc/iBml1JQODbJ6wYX4oOHV+E+IvIh/1dsUNzLDBMxfqa2Ob1f1
ACio/w==">
                        <kerberos value=""/>
                </identified_by>
        </user>
</authentication-configuration>
```

Save the file changes and after that, start the dbWatch service and open the dbWatch Control Center monitor. The plaintext password will be encrypted and written back to the file.

<< Definition Mapping / Fresh Installation of dbWatch Control Center >>

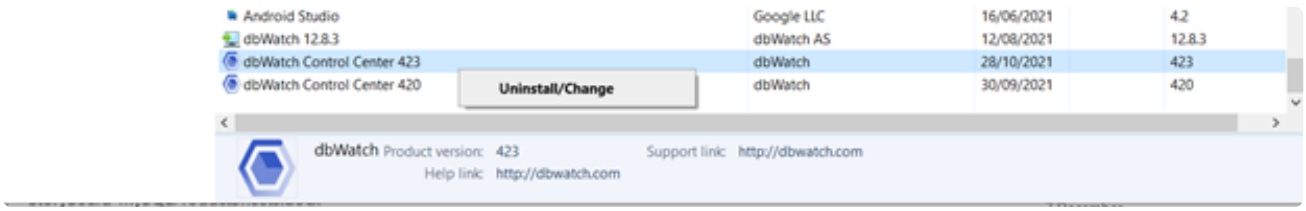
Fresh Installation of dbWatch Control Center

After installing a newer version of dbWatch Control Center, you may encounter problems such as an expired license or a locked account due to forgetting the admin credentials. Here's a simple guide on doing a fresh installation of dbWatch Control Center.

Removing the old version of dbWatch Control Center

> ! To start, do not install the latest version of dbWatch. We'll do that later. If you did, then just follow the guide below. Remove all directories of both the old version and the latest version.

First, you need to remove the old version of dbWatch Control Center. Go to the control panel and select dbWatch Control Center. Right-click on it and choose "uninstall/change". Wait for the uninstallation to finish.

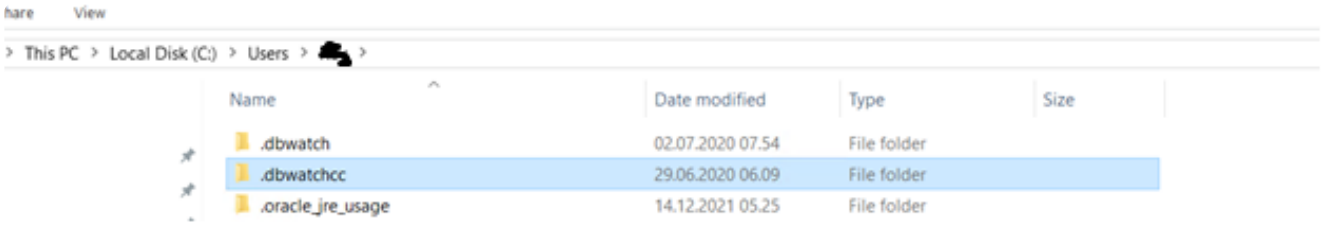


Next, open these four directories:

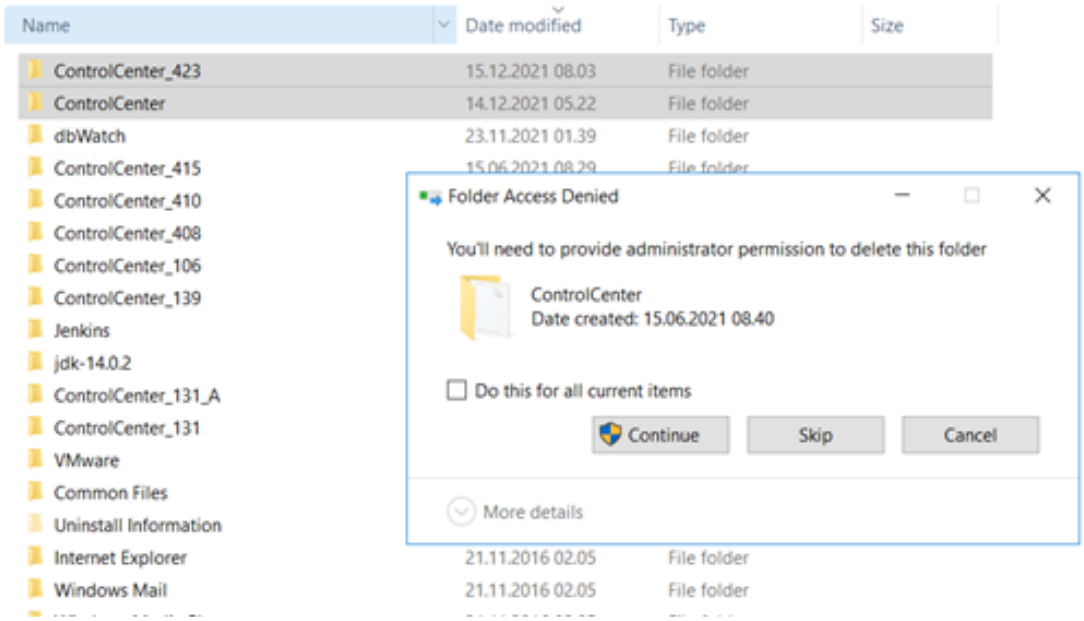
Description	Directory Path
Install Directory	C:\Program Files\ControlCenter\
Server work directory	C:\ProgramData\dbWatchControlCenter\
Library directory	C:\ProgramData\dbWatchControlCenter\archives\
Monitor work directory	C:\Users\(Users)\.dbwatch\.monitor\

Then, remove all of the above-mentioned directories.

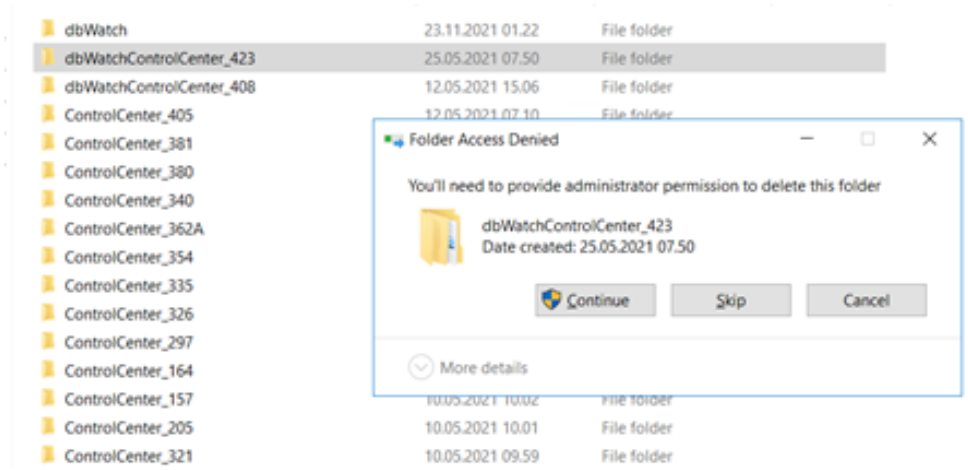
Removing Monitor Work Directory:



Removing Server Work Directory and Library Directory:



Removing Install Directory:

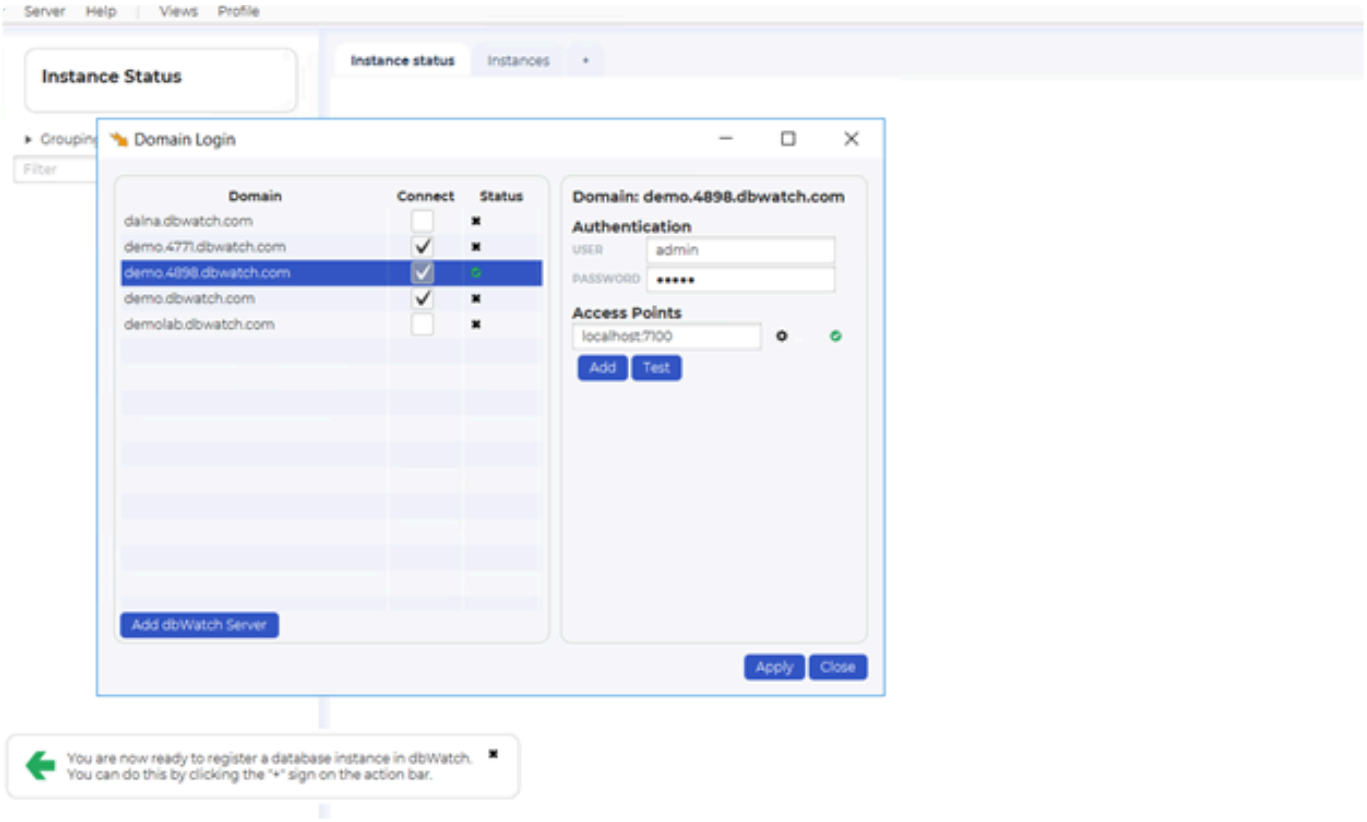


Why should we remove the specified directories?
During the uninstallation process, dbWatch Control Center does not remove these four directories. Most problems stem from having an old license file or using old credentials. To emulate a fresh batch of installation, you should remove the old files so when dbWatch is installed and boots up then you can use the software without a problem.

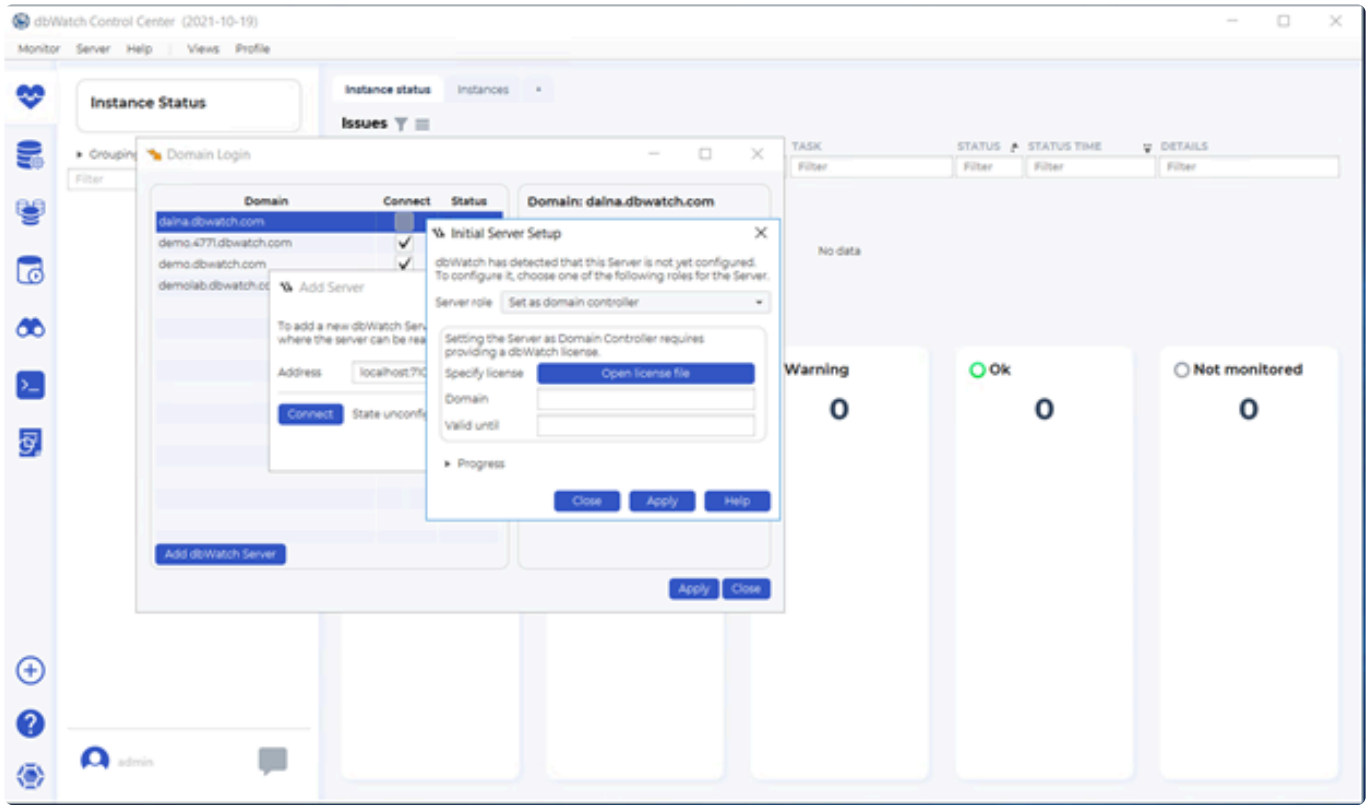
Installing the latest version of dbWatch Control Center

After doing all the steps above, you can now install the latest version of dbWatch Control Center. Just follow the steps for the windows installation.

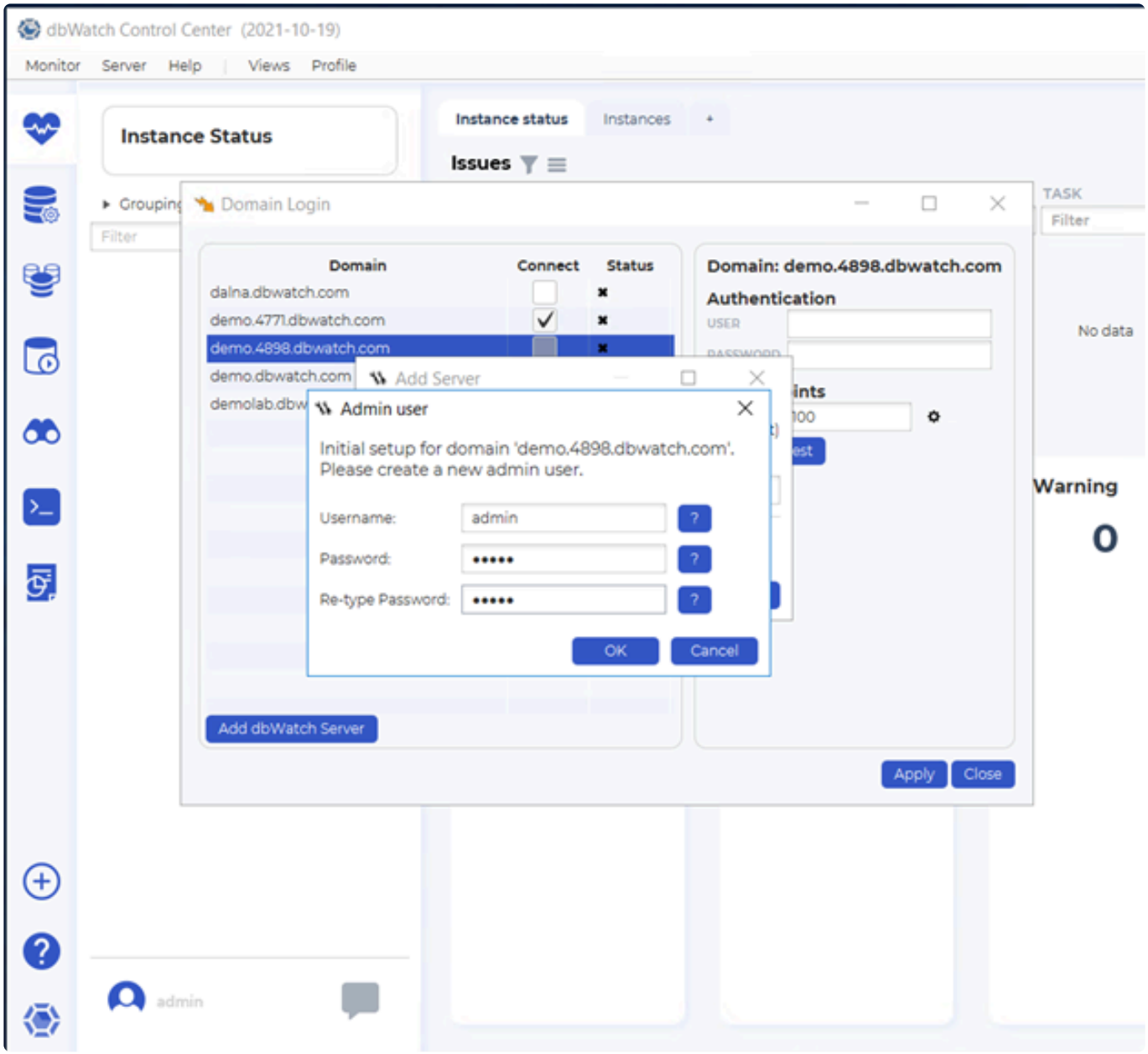
After installing dbWatch Control Center, open the application. Once the UI loads, double click the username or open monitor > domain login. You should be able to see the following:



Click on “Add dbWatch Server” and simply click “Connect”. This will prompt the initial server set-up window:



Select “set as demo server” and click “Apply”. A new window will appear and you will input your new admin credentials.



Once you're done with all these steps, you are now ready to use the latest version of dbWatch Control Center. To know more about Domain Configuration, you can see Initial Domain Set-up and Define Domain Controller

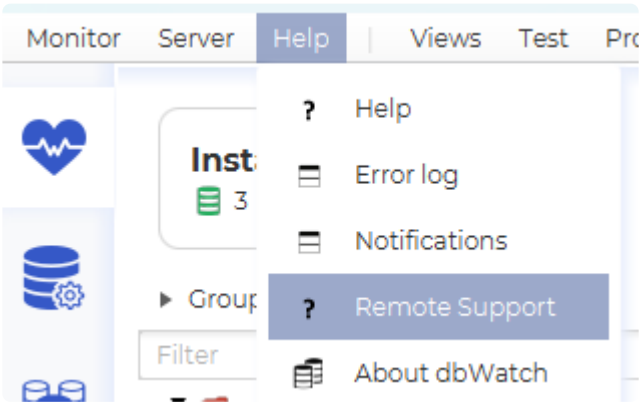
<< Resetting dbWatch Control Center Admin Account / Remote Support >>

Remote Support

It is sometimes useful to allow someone external to access your system in a controlled manner. For example, during a support session or when receiving assistance from a consultant.

In dbWatch this process is streamlined through the **Remote Support** functionality. This creates a temporary user, and gives external access by connecting to a dbWatch Cloud Router. An audit trail is created for all the actions performed by the user, and the audit log is saved. Afterwards the user is deleted and the Cloud Router connection is terminated.

This functionality can be found by clicking the “Remote Support” item on the “Help” menu.

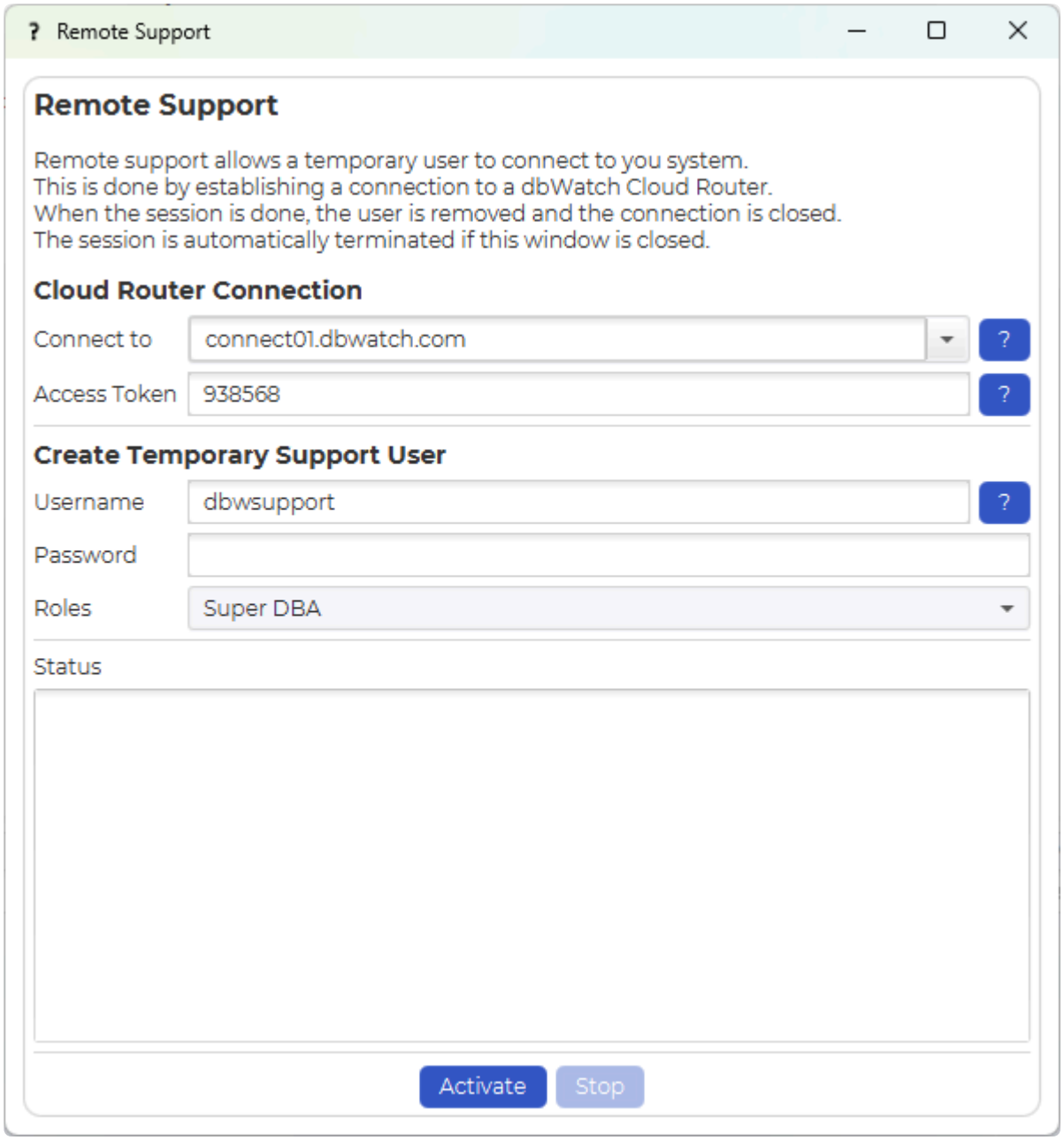


This brings up the **Remote Support** dialog.

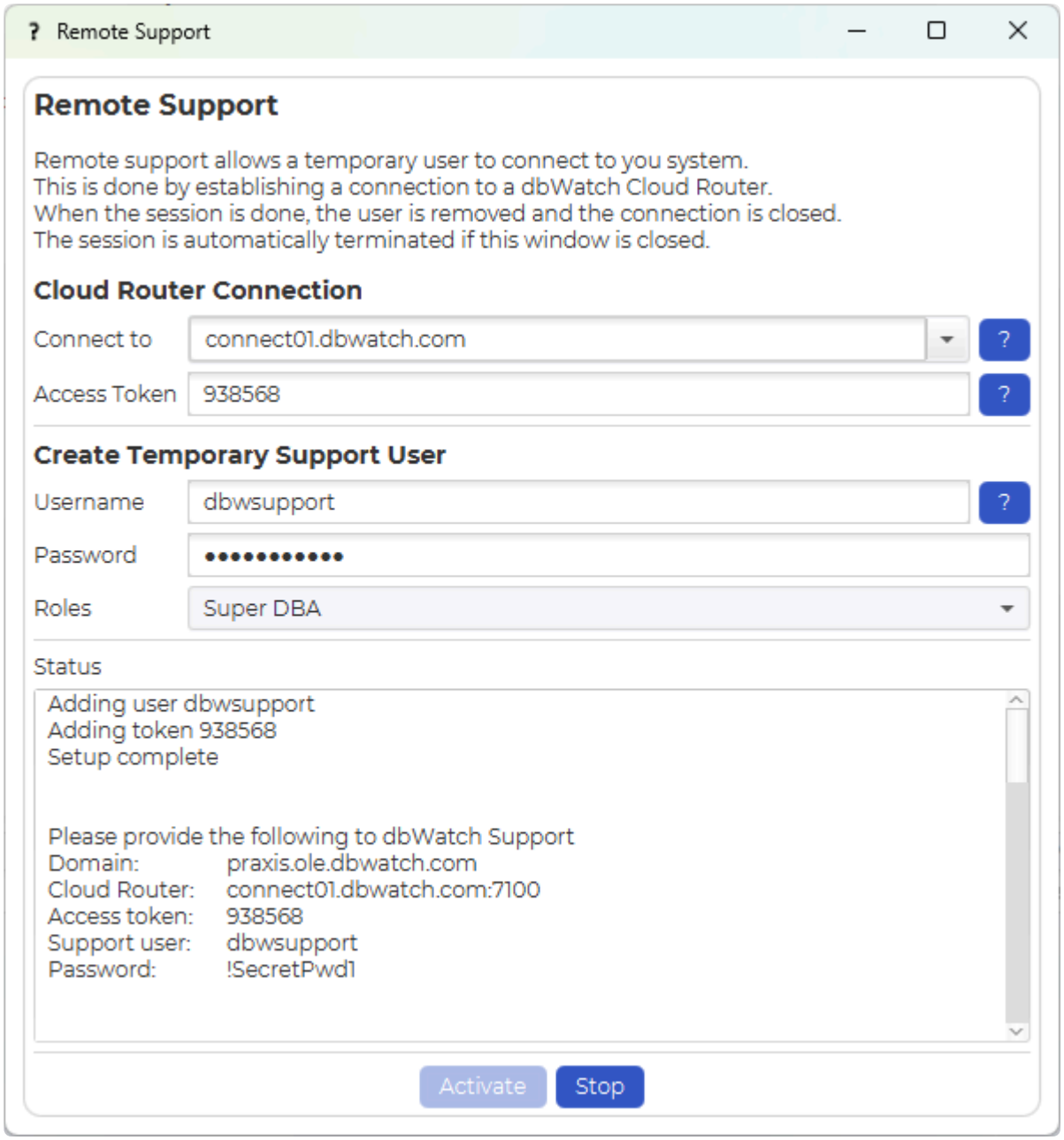
The **Cloud Router Connection** section defines the URL to the Cloud Router and an access token. dbWatch offers publicly available Cloud Routers at the connectXX.dbwatch.com URLs. (connect01 should always be available). A random access token is generated, but if you prefer, you can change this to anything you want.

The **Create Temporary Support User** section defines the user that will be created. A default username is suggested, but you can choose another name if you prefer. Create a password that meets your password policy requirements. Also, specify the Roles you want the user to have.

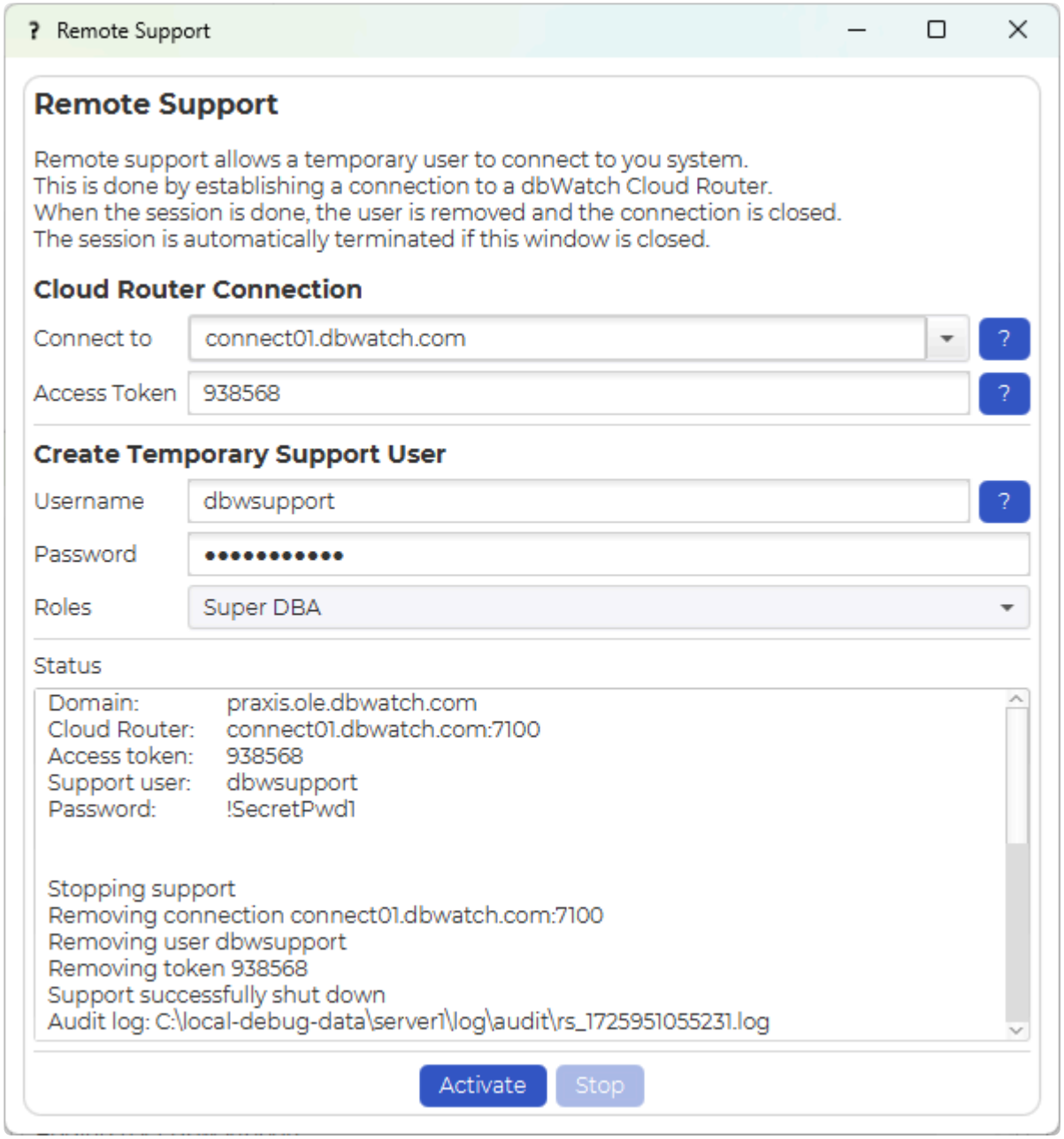
The **Status** field will display progress information and audit data.



Click the Activate button when satisfied.



If the connection is successful, the status field will give a summary of the data that you need to provide to the person that you want to connect.
When the temporary user connects, it will be visible in the status field.



The status field will also show the audit trail of the user actions. When the session is complete, the user and the connection are removed and the path to the audit log is shown.
The session will be terminated either by clicking on the Stop button, or by closing the Remote Support dialog.

Additional Resources

Here are an additional resources that might help you.

You can check all the FAQS on our website. Go to dbWatch/faq
To check our video libarary, you can access the link provided: dbWatch/videos

<< Remote Support

Database Farm Management Additional Resources

Blog

This blog delves deeper into this topic:
[From SQL Instance Management to Database Farm Management]

Video Resources

There are many videos on our [Youtube channel]

You can watch our presentation on Database Farm on our [Youtube] account:

1. **Webinar: Beyond Instance Tuning**

https://www.youtube.com/embed/cAFSjixFK64?rel=0

2. **Webinar: How does Database Farm Management can make DBA's life easier?**

https://www.youtube.com/embed/VHbPq6ZKs_A?rel=0

<< Additional Resources / Licensing terms >>

Licensing terms

dbWatch End User License Agreement

1. PARTIES

This License and Services Agreement (“Agreement”) is between dbWatch AS, organization number 991 500 456 Norway (“dbWatch”), and the Customer, which is the entity or person whose name and contact details as it appears on the order form or purchase agreement (“Customer”).

2. DEFINITIONS

“Agreement” meaning this License and Services Agreement, including any Exhibits.

“Customer” meaning the company identified above, including any partnership, joint venture, corporation or other form of enterprise that Customer directly or indirectly controls. Such entities will be known as “Affiliates”. As used in this definition, “controls” means (1) ownership, directly or indirectly, being fifty percent (50%) or more of voting securities of such party or (2) majority ownership of an entity as described herein.

“Documentation” means the user, system, and installation documentation for the Software, as provided and updated online by dbWatch from time to time.

“Software” means the features, functions, and configurations in any of dbWatch’ proprietary computer programs obtained under this Agreement (as chosen by the Customer online at the Internet site hosted by dbWatch), in machine-readable, object code form only, and any Updates thereto made available to Customers by dbWatch in machine-readable, object code form.

“Effective Date” means the date of Customer accepts the terms and conditions of this Agreement by clicking “Accept” or “Ok” during installation or upgrade of the Software.

“Error” means an abnormality in the Software which significantly degrades such Software as compared to the published performance specifications described in its Documentation.

“Update” means a new version of the Software, which is indicated by the Release date. Typically, an Update Release will include corrections, performance improvements, and new features.

“Version” means the Release date of the software.

3. LICENSE

3.1 Grant of License

In consideration of the License Subscription Fee paid by the Customer from time to time and subject to the terms and conditions of this Agreement, dbWatch grants to Customer a limited, non-sub-licensable, non-exclusive, non-transferable (except as expressly provided below) license for the term covered by the

License Subscription Fee (cf. clauses 6.1 and 12.1) to: (a) install or have installed, the Software on Customer's Enterprise Network, (b) use the Software, in accordance with the Documentation solely for its own internal use, © make one copy of the Software for archival or backup purposes only, provided that all titles, trademarks, copyright, proprietary and restricted rights notices shall be reproduced in such copy, and that such copy shall be subject to the terms of this Agreement, and (d) make copies of the Documentation as necessary or convenient for the internal use of the Customer, provided that all titles, trademarks, copyright, proprietary and restricted rights notices shall be reproduced in all such copies, and that all such copies shall be subject to the terms of this Agreement.

3.2 License Restrictions

Customer may not (except to the extent permitted by section 39i of the Norwegian Copyright Act 1961 (åndsverksloven)): (a) modify, disassemble, decompile or reverse engineer the Software nor permit or encourage any third party to do so, (b) use the Software in any manner to provide service bureau, time-sharing or other computer services to third parties, except as expressly provided herein, © use the Software in any manner to assist or take part in the development, marketing, or sale of a product potentially competitive with the Software, (d) reveal to any third party any benchmark results, comparing any part of the Software with any potentially competing product, (e) use the Software, or allow the transfer, transmission, export, or re-export of the Software or portion thereof in violation of any export control laws or regulations of any government agency, or (f) read, write, view or extract information without the appropriate license.

3.3 Use of Software by Third Parties

Customer may allow its contractors and agents to use the Software, solely on its behalf, in the same manner as Customer under this Agreement and subject to all the terms and conditions of this Agreement. Any such contractor or agent using the Software will be deemed to have agreed to comply with this Agreement. The customer will notify the contractor or agent of the same and will be responsible for any non-compliance by it.

3.4 Limited Rights

Customer's rights to the Software will be limited to those expressly granted in this License Terms Section 3 . dbWatch reserves all rights and licenses in and to the Software not expressly granted to Customer under this Agreement. Any Affiliate using the Software will be deemed to have agreed to comply with this Agreement. The customer will notify such Affiliate of the same and will be responsible for any non-compliance by it.

3.5 Changes to Software

From time to time, dbWatch may require that the Software is discontinued, changed, or improved. dbWatch shall have the right to discontinue the Software and make any changes and improvements to the Software at any time without dbWatch incurring any obligation to Customer. Customer shall accept the Software so changed or improved in the fulfillment of this Agreement.

4. OWNERSHIP

dbWatch shall have and retain all worldwide rights, title, and interest in and to the Software, including any Fixes, and Updates, any modifications made to the foregoing by whoever made feedback related to the above, and the Documentation. Customer hereby assigns to dbWatch any interest it may have, or may obtain, in the foregoing, free of confidentiality restrictions.

5. MAINTENANCE AND SUPPORT

5.1 Maintenance

Subject to payment by the Customer of the appropriate fees, dbWatch will provide the maintenance services for the Software as described below.

During the period where Customer is provided Maintenance, Customer shall be offered all Releases of the Software of all parts and modules the Customer has a fully paid up subscription for, free of additional charge (other than any taxes and duties that may be imposed). By utilizing the dbWatch Support Web Site, the Customer's designated support contacts will be notified about Releases as they become available. Work required to implement an Update is the sole responsibility of the Customer. dbWatch may provide, for a fee, services to assist in this process if requested by the Customer.

5.2 Support

Subject to payment by the Customer of the appropriate fee, dbWatch shall provide Remote Support services to the Customer's designated technical support contacts concerning the use and performance of the licensed Software. "Remote Support" means technical assistance concerning the use of the Software via email provided by dbWatch during its regular business hours between 0800 and 1700 Norwegian Time Monday thru Friday, excluding weekends and Norwegian holidays.

dbWatch shall exercise commercially reasonable efforts to correct any Error reported by Customer in the Software.

dbWatch will provide technical support only for the most recent version of the Software and will provide support for the highest Release of the last prior Version of the Software for a period (not to exceed twenty-four (24) months) following the General Availability of the new Version. Technical support for the prior Version may not include Updates or code-level fixes.

dbWatch shall have no obligation of any kind to provide maintenance and support services for problems in the operation or performance of the Software caused by any of the following (each a "Customer-Generated Error"): (a) software or hardware products not recommended or specified by dbWatch; (b) Customer's failure to properly maintain Customer's site and equipment on which the Software is installed; © alterations to Customer's site or equipment made by Customer or a third party after installation of the Software.

If dbWatch determines that it is necessary to perform maintenance and support services for a problem caused by a Customer-Generated Error, dbWatch will notify Customer thereof as soon as dbWatch is aware of such Customer-Generated Error, and dbWatch will have the right to invoice Customer at

dbWatch's then-current rates for time and materials for all such maintenance and support services performed by dbWatch.

5.3 Other Services

dbWatch may, in its sole discretion, perform installation and other additional services for Customer as requested by Customer in accordance with the terms and conditions of this Agreement. dbWatch will invoice Customer at dbWatch's then-current rates for time and materials for all such additional services performed by dbWatch.

6. PAYMENTS AND AUDITS

6.1 License Subscription Fee

The Customer shall pay to dbWatch the License Subscription Fee in the amount of as specified in their contract with dbWatch. Payment of the License Subscription Fee will provide the Customer with a license to use the Software as set out in this Agreement for a term of 12 months. The Customer is entitled to prolong the license period for an additional 12-month period as set out in clause 12.1, provided that the applicable License Subscription Fees for such additional periods are paid to dbWatch.

6.2 Payment Terms and Taxes

Any overdue payments shall bear a late payment fee of one and a half percent (1.5%) per month, or if lower, the then-current late payment interest rate, pursuant to the Norwegian Act on late payment Interest. Customer will be responsible for and will promptly pay and agrees that dbWatch Software or its payment solution provider may charge the Customer's credit or debit card all taxes of any kind (including but not limited to sales and use taxes) associated with this Agreement or Customer's receipt or use of the Software and services, except for taxes based on dbWatch's net income.

6.3 Certification

At dbWatch's written request, not more frequently than annually, Customer shall furnish dbWatch with a signed certification: (i) verifying that the Software is being used pursuant to the provisions of this Agreement; and (ii) listing the locations, types, and serial numbers and names of users of all equipment upon which the Software is run.

6.4 Audit

dbWatch may, at its own expense, audit the Customer's use of the Software. Any such audit shall be conducted during regular business hours at Customer's facilities and shall not unreasonably interfere with Customer's business activities. If an audit reveals that the Customer has underpaid license subscription fees to dbWatch, the Customer shall be invoiced for any such underpaid fees according to dbWatch's latest prices for such software. If the underpaid fees exceed 2% of the license subscription fees paid, then the Customer shall also pay dbWatch's reasonable costs of conducting the audit. Audits shall be conducted no more than once annually.

7. WARRANTY

7.1 Limited Software Warranty

The following shall be Customer's sole and exclusive remedy and dbWatch's entire liability for any breach of the foregoing. dbWatch warrants that for a period of thirty (30) days after the Effective Date that the Software: (a) will be free from defects in materials and workmanship under regular use; and (b) the Software will function substantially in accordance with the Documentation. This warranty covers only problems reported to dbWatch during the warranty period. The customer may perform acceptance testing of the Software so long as the acceptance period is within the warranty period and the intent of the acceptance testing is to determine that the Software will function substantially in accordance with the Documentation. dbWatch will, at its sole option and expense, promptly repair or replace any Software which fails to meet this limited warranty or, if dbWatch is unable to repair or replace the Software, refund to Customer the applicable fees, if any, already paid by Customer upon return of the non-conforming Software to dbWatch.

7.2 Disclaimer of Warranties

Except for the express warranty set forth in this section (7), dbWatch makes no warranties, expressed or implied, with respect to the Software, the Maintenance and Support Services, or any other materials (tangible or intangible) or services supplied by dbWatch, and dbWatch hereby expressly disclaims any implied warranties, including without limitation, any implied warranties of merchantability, fitness for a particular purpose, non-interference, accuracy of data and non-infringement.

7.3 Third-Party Software

dbWatch acknowledges that a portion of the Software provided to Customer under this Agreement contains a third-party code. dbWatch represents and warrants that dbWatch has the legal rights to license such third-party code and that the Software will be fully supported in accordance with the Maintenance and Support Services.

8. INDEMNIFICATION

8.1 Infringement Indemnity

dbWatch will indemnify Customer and, at its option, defend any action brought against Customer to the extent that it is based upon a claim by a third party that the Software, as provided by dbWatch to Customer under this Agreement and used within the scope of this Agreement, infringes any Norwegian, copyright, trademark, trade secret, or any other proprietary right of any kind, and will pay any costs, damages and reasonable attorneys' fees and court costs attributable to such claim that are awarded against Customer, provided that Customer: (a) notifies dbWatch in writing of the claim within ten (10) days after becoming aware of such claim; (b) grants dbWatch sole control of the defense and settlement of the claim if dbWatch assumes such defense; and © provides dbWatch with all assistance, information, and authority required for the defense and settlement of the claim.

8.2 Injunctions

If Customer's use of the Software hereunder is, or in dbWatch's opinion is likely to be, enjoined due to the type of infringement specified in Section 8.1 above, dbWatch may, at its sole option and expense: (a) procure for Customer the right to continue using such Software under the terms of this Agreement; (b) replace or modify such Software or have it replaced or modified so that it is non-infringing and substantially equivalent in function to the enjoined Software; or © if options (a) and (b) above cannot be accomplished despite dbWatch's best efforts, then dbWatch may terminate Customer's rights and dbWatch's obligations hereunder with respect to such Software.

8.3 Exclusions

Notwithstanding the terms of Section 8.1 and 8.2, dbWatch will have no liability for any infringement claim of any kind to the extent it results from: (a) modification of the Software made other than by dbWatch; (b) the combination, operation, or use of any Software supplied hereunder with equipment, devices or software not supplied by dbWatch to the extent such a claim would have been avoided if the Software was not used in such combination; © failure of Customer to use updated or modified Software provided by dbWatch to avoid the infringement; (d) unauthorized or unintended use of the Software; (e) failure by Customer to take all reasonable action to prevent or mitigate losses, damages, costs, and expenses; or (f) compliance by dbWatch with designs, plans or specifications furnished by or on behalf of Customer. Customer shall indemnify, defend and hold dbWatch harmless against any expense, judgment or loss for alleged infringement of any patents or copyrights or misappropriation of trade secrets that result from dbWatch's compliance with Customer's designs, specifications or instructions.

8.4 Sole Remedy

The provisions of this Section 8 set forth dbWatch's sole and exclusive obligations, and Customer's sole and exclusive remedies, with respect to infringement of intellectual property rights of any kind.

8.5 Indemnity by Customer

Customer will indemnify dbWatch and, at its option, defend any action brought against dbWatch arising out of or related to (a) any claim of infringement or misappropriation of any intellectual property to the extent that such infringement or misappropriation is caused by actions taken by Customer, (b) Customer's violation or alleged violation of any applicable laws or © the conduct of Customer's business.

9. CONFIDENTIALITY

9.1 Definition

"Confidential Information" means: (a) the Software; (b) the terms of this Agreement, and © any business or technical information of dbWatch or Customer, including but not limited to any information relating to dbWatch's or Customer's product plans, designs, costs, product prices and names, finances, Customer's evaluation or opinion of dbWatch's performance under the Agreement, marketing plans, business opportunities, personnel, research, development or know-how, in either written or oral form, that is designated by the disclosing party as "confidential" or "proprietary."

9.2 Exclusions

Confidential Information does not include information that: (a) is or becomes generally known to the public through no fault or breach of this Agreement by the receiving party; (b) is known to the receiving party at the time of disclosure without an obligation of confidentiality; © is independently developed by the receiving party without the use of the disclosing party's Confidential Information; (d) the receiving party rightfully obtains from a third party without restriction on use or disclosure, or (e) is disclosed with the prior written approval of the disclosing party.

9.3 Use and Disclosure Restrictions

During the term of this Agreement, and for (i) perpetuity with respect to the Software, and (ii) a period of five (5) years after any termination of this Agreement with respect to any Confidential Information other than the Software, each party will not use the other party's Confidential Information except as permitted herein, and will not disclose such Confidential Information to any third party except to employees and consultants as is reasonably required in connection with the exercise of its rights and obligations under this Agreement (and only subject to binding use and disclosure restrictions at least as protective as those set forth herein executed in writing by such employees and consultants). However, each party may disclose Confidential Information of the other party: (a) pursuant to the order or requirement of a court, administrative agency, or other governmental body, provided that the disclosing party gives reasonable notice to the other party to contest such order or requirement; and (b) on a confidential basis to legal or financial advisors.

9.4 Disclosure of Agreement Terms

Notwithstanding Section 9.3, either party may disclose the terms of this Agreement pursuant to an acquisition, merger, or sale of substantially all of such party's assets, financing, or as required by securities laws or regulations; provided that the receiving party shall be bound to a confidentiality agreement to the extent possible.

9.5 Security of Customer Systems

In performing pursuant to this Agreement, dbWatch, its employees, agents, subcontractors, and any other individual permitted by dbWatch to access any computer system, network, file, data, or software owned by or licensed to Customer will: (i) use and take all reasonable security measures necessary to protect the security of all such computer systems, networks, files, data and software; and (ii) abide by the system security requirements and guidelines of Customer.

10. Not Used

11. LIMITATION OF LIABILITY

11.1 Total Liability

dbWatch's cumulative liability to Customer, from all causes of action and all theories of liability, will be limited to and will not exceed the annual amount paid to dbWatch by Customer pursuant to this

Agreement for the Software, which is the subject of the cause of action or claim. The limitation of the liability shall apply notwithstanding the failure of any limited remedy to fulfill its essential purpose.

11.2 Exclusion of Damages

In no event will dbWatch be liable to Customer for any special, indirect, incidental, or consequential damages (including loss of use, data, business or profits) arising out of or in connection with this Agreement or the use or performance of the Software or services, whether such liability arises from any claim based upon contract, warranty, tort (including negligence), product liability or otherwise, and whether or not dbWatch has been advised of the possibility of such loss or damage.

11.3 Basis of Bargain

The parties expressly acknowledge and agree that dbWatch has set its prices and entered into this Agreement in reliance upon the limitations of liability specified herein, which allocate the risk between dbWatch and Customer.

12. TERM AND TERMINATION

12.1 Term and Termination for Convenience

Unless terminated earlier in accordance with the terms of this Agreement, this Agreement will begin on the Effective Date and will remain in effect for an initial period of 12 months (the “Initial Term”) after which it will, provided that the Customer pays the applicable Subscription Fees, automatically renew for an additional 12 month period (each such period to be called a “Renewal Period”) unless terminated in writing by either Party at least 30 days prior to the expiry of the Initial Term or any subsequent Renewal Period.

12.2 Termination for Breach

Each party will have the right to terminate this Agreement or any Software license granted hereunder if the other party materially breaches any material term of this Agreement and fails to cure such breach within thirty (30) days after written notice thereof.

12.3 Effect of Termination

Upon any termination of this Agreement, Customer will, at dbWatch’s request, promptly return the Software to dbWatch or destroy the Software and all copies and portions thereof, in all forms and types of media, and provide dbWatch with an official written certification, certifying to Customer’s compliance with the foregoing.

12.4 Non-exclusive Remedy

Termination of this Agreement by either party will be a non-exclusive remedy for breach and will be without prejudice to any other right or remedy of such party.

13. GENERAL

13.1 Assignment

The customer will have no right to assign this Agreement, in whole or in part, without dbWatch's prior written consent. Any attempt to assign this Agreement without such consent will be null and void. dbWatch may assign this Agreement to any entity.

13.2 Governing Law and Jurisdiction

This Agreement will be governed by and construed in accordance with the laws of Norway. Any legal action or proceeding arising under this Agreement will be brought exclusively in the courts in Oslo, Norway and the parties hereby consent to personal jurisdiction and venue therein.

13.3 Severability

If, for any reason, a court of competent jurisdiction finds any provision of this Agreement invalid or unenforceable, that provision of the Agreement will be enforced to the maximum extent permissible, and the other provisions of this Agreement will remain in full force and effect.

13.4 Waiver

The failure by either party to enforce any provision of this Agreement will not constitute a waiver of future enforcement of that or any other provision.

13.5 Force Majeure

Neither party will be responsible for any failure or delay in its performance under this Agreement due to causes beyond its reasonable control, including but not limited to labor disputes, strikes, lockouts, shortages of or inability to obtain labor, energy, raw materials or supplies, war, riot, act of God or governmental action.

13.6 Relationship of Parties

The parties to this Agreement are independent contractors, and this Agreement will not establish any relationship of partnership, joint venture, employment, franchise, or agency between the parties. Neither party will have the power to bind the other or incur obligations on the other's behalf without the other's prior written consent.

13.7 Publicity

At the time of Agreement execution, Customer concurs with dbWatch using the Customer and Customer's logo as a reference in its sales and marketing activities. The customer also agrees that its name may be included in dbWatch's customer lists and provide quotes from Customer's executive team for dbWatch's website and for various marketing documents.

13.8 Entire Agreement

This Agreement contains the complete understanding and agreement of the parties and supersedes all prior or contemporaneous agreements or understandings, oral or written, relating to the subject matter herein. Any additional or contrary terms and conditions that may appear on a Customer's purchase order submitted to dbWatch for Software licenses and/or services provided by dbWatch will be considered to be null and void. Any waiver, modification, or amendment of any provision of this Agreement will be effective only if in writing and signed by duly authorized representatives of the parties.

13.9 Export Control

The customer agrees to comply with applicable laws, regulations, rulings, and executive orders on exportation (not only Norwegian laws and regulations) and with all applicable laws on the import of software and technology.

<< Database Farm Management Additional Resources