Contents

[QueueCheck Probe 2.9.4 3](#_Toc111736247)

[1 – Setup 4](#_Toc111736248)

[1.1 Import probe into your local archive 4](#_Toc111736249)

[1.2 Deploy the probe on your master hub 4](#_Toc111736250)

[1.3 Nimsoft\_generic.dat 4](#_Toc111736251)

[1.4 queuecheck.cfg file 5](#_Toc111736252)

[1.4.1 Generic Parameters 5](#_Toc111736253)

[1.4.2 Queue Parameters 7](#_Toc111736254)

[1.5 Blat setup 12](#_Toc111736255)

[1.6 CI\_Type setup 14](#_Toc111736256)

[1.7 Verify Probe definitions 15](#_Toc111736257)

[1.8 Start the probe 16](#_Toc111736258)

[1.9 Check created QoS 16](#_Toc111736259)

[1.10 execute Qos\_View 17](#_Toc111736260)

[1.11 customize: mailing script 17](#_Toc111736261)

[1.12 QoS default retention 18](#_Toc111736262)

[2 Probe Output (QoS and Alarm) 20](#_Toc111736263)

[2.1 QoS created 20](#_Toc111736264)

[2.2 Generated Alarms 20](#_Toc111736265)

[2.3 Logfile 22](#_Toc111736266)

[3 – QOS Usage in your Dashboard 23](#_Toc111736267)

[3.1 Dashboard import 23](#_Toc111736268)

[3.2 Color Mapping 24](#_Toc111736269)

[3.3 Empty Dashboard 25](#_Toc111736270)

[4 – QOS Usage in Metric View 26](#_Toc111736271)

[5 - QOS Usage in CABI/Jaspersoft report 30](#_Toc111736272)

[6 - Probe Logic 33](#_Toc111736273)

[7 - Example scenarios 34](#_Toc111736274)

[7.1 Only create Qos for queued with no alarms 34](#_Toc111736275)

[7.2 Create Qos for queued with alarm if threshold is breached 37](#_Toc111736276)

[7.3 Create Qos for queued with alarm and with exceptions 38](#_Toc111736277)

[*8* - Nimsoft\_generic.dat (settings) 39](#_Toc111736278)

[8.1 SQL driver 42](#_Toc111736279)

[8.2 ODBC System DSN 43](#_Toc111736280)

[9 - Create Windows Perl environment 50](#_Toc111736281)

[10 – Possible Problems 53](#_Toc111736282)

[10.1 QoS data retention 53](#_Toc111736283)

[10.2 S\_QOS\_Snapshot dtata retention 53](#_Toc111736284)

[10.3 Trellis not processing alarms 54](#_Toc111736285)

[10.4 log messages 71 &72 54](#_Toc111736286)

# QueueCheck Probe 2.9.4

To monitor the health of our UIM environment, monitoring the hub and the nas (internal) queues are one of the most important tasks to guarantee a healthy UIM environment.

Queuing higher than user specified thresholds will warn you, via alarm(s), dashboard and/or mail that you risk that your UIM environment is not performing like expected.

Probe QueueCheck can: (via parameters in queuecheck.cfg)

* Monitor hub and nas active
* Monitor Hub queues
* Monitor internal Nas queues
* Count number of queues (important for TEMP queues)
* Send Alarm when hub/nas threshold is breached or hub/nas is inactive
* Send email when hub/nas treshold is breached or hub/nas is inactive
* Create QoS metrics for Nas and Hub queued
* Create QoS metrics for Hub count (=total #processed qos/alarms since start)
* Create QoS metric for #processed alarm/Qos for each probe interval
* Sample dashboard to show generated QoS
* Send Alarm when GET queue is not connected
* Send alarm when ATTACH queue has no consumer
* Clear alarm when problem is solved

Each of the tests/tresholds can be activated/inactivated via the config settings.

One of the advantages of the QueueCheck probe is that it monitors automatically all your hubs and all your nas queues. If you add a new hub, the next run of the probe will detect the addition and add the new hub/nas.

The probe uses a cfg file where you can define the queues, thresholds and if you want also a mail in case of certain alarms.

**Note**: one of the reasons to create this QueueCheck probe is that the older LUA script to monitor the queues (get\_hub\_queue\_status.lua) was not able to create 100% correct QoS metrics that could be viewed/added via the Operator Console Metric Viewer. (nsa.exe is/was using the source as ci\_name where it should take the target to create the ci\_name)

# – Setup

This probe was written in Perl and is packaged with the Perl source and with the compiled Perl. If you don’t have Perl installed on your UIM Master hub, you can use the compiled version.

## Import probe into your local archive

This tool is packaged in 2 zip file:

* The probe is available as a zip file, format: **queuecheck\_2.9.4.zip**. Copy this zip file into your local archive.
* Additional files to help you to setup, document the probe are located in file: **queuecheck\_utilities\_2.9.4.zip.** Unzip this file to access additional setup files.

**Note:** the zip file version number can be different that noted here

## Deploy the probe on your master hub

Deploy the QueueCheck probe, latest version, to your master hub. This will create a subdirectory under your UIM probes folder: /probes/custom/queuecheck

It is normal that the probe is not started after deployment; you have to perform some “one time” customization steps before starting the probe.

## Nimsoft\_generic.dat

The probe package contains nimsoft\_generic.dat to avoid overwriting your possible previous custom settings.

To avoid that you have to repeat all security and server setup parameters for each tool, we created a unique parameter file that contains all important information:

* Uim server
* Sql server
* Userid’s
* Password in encrypted format

If you use already a tool that use this nimsoft\_generic.dat file, you can copy that already customized file in the /probes/custom/queuecheck directory (and go directly to next section)

If this is your first tool that uses nimsoft\_generic.dat, perform the steps explained in chapter 7.

**Note1**: the file nimsoft\_generic.pm is used to decode the nimsoft\_generic.dat file

**Note2**: if you use multiple tools that use nimsoft\_generic.dat, and you want that all tools use the same version of nimsoft\_generic.dat, you can copy the file to your perl/lib directory

## queuecheck.cfg file

The probe package contains the file queuecheck.cfx to avoid that the deploy of the new probe will overwrite previous custom settings.

Most probe configuration parameters other that server name and user/password related are located in: QueueCheck.cfg

The default queuecheck.cfg can be used as a good starting point; it will not try to send emails, only generate alarms for found problems. The only thing to adapt is the threshold levels suitable to your environment.

### 1.4.1 Generic Parameters

* **loglevel** = 0: the probe execution will create a log file in the probes/custom/queuecheck directory: queuecheck.log. With the loglevel you can indicate how much logging you need during execution. When you report a probe problem, please have a loglevel=2 execution available. Loglevel=1 will add for each queue 2 addition log lines that will show more details about the queue difference calculation.
* **logfile** = queuecheck.log: default name of the log file
* **interval** = 180: probe execution interval in seconds.
* **logsize** = 1000000
* **mail\_script** = c:/tools/nas\_mail\_queued.bat: script name that will be launched when we want to send a mail. The example script is using the open source tool “blat” to send emails, see next section for the first setup and test. Use a / as folder delimiter and enclosed by “
* **mail\_bypass\_times** = 10: in case we send an alarm mail due to a threshold breach, we will not send again an email the next xx executions. This to avoid that you receive the same email on each execution.
* **Mail\_bypass\_alarm\_level =** 1: alarm level (1-5) used for mail\_bypass\_alarm. If you don’t need/want an alarm in case a mail is bypassed, set this value to 0 or no.
* **device\_name** = robot: each generated QoS will try to create the qos metric in a format that is suitable for dashboards and OC PRD and OC Metric Viewer. This parameter device\_name decides what value we will enter in the QOS device\_name:
  + **robot**: This is the default and will use the hub robot name in the device\_name of the qos. This means that if you goto OC metric Viewer you will see Private – Custom in the select metrics view (see later)
  + **hub**: In case you don’t want to mix the queuecheck metrics with your normal hub robot metrics, you can store them under the hub name. Your hub names will then be available as a device.
* **Alarm\_level\_hub\_inactive** = 5 : alarm level (1-5) used for the alarm when a hub is found inactive. Default: 1.
* **Mail\_on\_hub\_inactive** = no: do you want a mail if a hub is found inactive (yes/no)
* **Alarm\_level\_nas\_inactive** = 5 : alarm level (1-5) used for the alarm if an installed nas is found to be inactive. Set the value to **no** is you don’t need/want this alarm. Default: 1.
* **Mail\_on\_nas\_inactive** = no : do you want a mail if a nas is found inactive (yes/no)
* **Hub\_exclude:** add, in regex format, hubs that you want to exclude from processing.
* **Nas\_exclude:** add, in regex format, nas names to exclude. This name will be matched on the NAS address /domain/hub/robot.
* **Get\_queue\_exclude:** add, in regex format, get queue names that you want to exclude from processing. This will be compared with the queue address format: /domain/hub/robot/queue\_name

Example: \/your\_hub\/.\*\/\*part\_of\_get\_queue\_name\*

* **Attach\_queue\_exclude:** add, in regex format, attach queue names that you want to exclude from processing. This will be compared with the queue address format: /domain/hub/robot/queue\_name.

Note: if you used hub\_exclude, all attach queues on that hub are also excluded.

* **Hub\_wait\_seconds =** 130: if the uptime of a hub is less than the defined seconds we will not check this hub in this interval. This to avoid that the probes on this hub are still starting and that the probe would flag these as being “without consumer”.
* **Request\_timeout =** 15. UIM callback timeout in seconds, default: 15
* **Request\_retries =** 4. UIM callback retry number if probe is not responding, default: 4.
* **Log\_subscriber\_info =** no. List in queuecheck.log an extra info for each hub with the number of subscribers used. This is important on a Windows hub because when arrived at 64, the hub can start having connection problems.
* **Check\_regex =** yes. Do you want to check the regex format of: get\_queue\_exclude, attach\_queue\_exclude and in the queue section the name field. (yes/no) Default: yes.

**Note**: when defining an exclude rule and you want to use an UIM address, you will need to escape the “/” character, like

Get\_queue\_exclude = \/domain\/hub\/robot\/get\_queue\_hubname

### 1.4.2 Queue Parameters

The queue parameters are grouped in sections so that you can be as generic or specific as you want.

Meaning; you can define 1 generic section for all ATTACH queues with all the same threshold, alarm and mail settings. Or you can define a generic ATTACH section and a more specific section for the data\_engine and nas queues. (or any other queue(s))

Define first the generic “name” definitions and than the more specific “name” definitions. The probe will read the definitions sequentially and use the last matching definition.

**Example:**

First we define a generic section for all ATTACH queues. In a second section we define a specific queue, data\_engine, with some different settings.

<queues>

<0>

name = .\*

type = ATTACH

qos\_on\_count = no

qos\_on\_queued = yes

qos\_on\_diff = no

alarm\_on\_queued = 1000

alarm\_on\_queued\_bypass = 1

alarm\_on\_diff = no

alarm\_on\_number\_of\_queues = no

alarm\_on\_attach\_no\_consumer = no

alarm\_on\_get\_not\_connected = no

alarm\_level\_attach\_no\_consumer = 2

mail\_on\_alarm\_queued = no

alarm\_level\_hub\_queue = 3

alarm\_level\_number\_of\_queues = 2

alarm\_level\_mail = 2

</0>

<1>

name = data\_engine

type = ATTACH

qos\_on\_count = yes

qos\_on\_queued = yes

qos\_on\_diff = yes

alarm\_on\_queued = 3000

alarm\_on\_diff = 10000

alarm\_on\_number\_of\_queues = no

alarm\_on\_attach\_no\_consumer = no

alarm\_on\_get\_not\_connected = no

mail\_on\_alarm\_queued = yes

alarm\_level\_hub\_queue = 5

alarm\_level\_diff\_queue = 1

alarm\_level\_number\_of\_queues= 2

alarm\_level\_mail = 2

ignore\_alarm\_time\_start = 2200

ignore\_alarm\_time\_end = 2330

</1>

…..

</queues>

**Note1**: in this example we miss several queue sections, one for the GET queues, one for the TEMP queues and another for the NAS queues. These missing sections are not required; if they are not present, these types of queues are not monitored.

**Note2**: the example delivered with the probe contains an example setup you can use to start with; only check the thresholds because they are set for a small test environment.

* **name** = .\* : generic or specific queue name. Use .\* to define all names for the specified TYPE. (regex format). Required field with no default.

This field will be matched against the hub or queue address in the format:

/domain/hub/robot/queue

* + .\* : will match every name of the defined TYPE
  + Data\_engine : will match the data\_engine name on all robots
  + \/hub\_name\/robot\_name\/queue\_name : will match a very specific queue name on a specific robot
  + Robot\_name : will exclude all queues where the robot name appears in the name
  + \/hub\_name\/robot\_name : will match all queues on that specific robot
* **type** = ATTACH : can be: ATTACH, GET, TEMP or NAS. Required field with no default.
* **qos\_on\_count** = no : valid for type=GET/ATTACH: create a qos/alarm for the number of qos metrics processed since the start of the probe. (yes/no) This QoS can give you an indication how many alarms or qos metrics are processed since start. Typically you will use these qos metrics in a PRD graph. Recommended is to use this parameter on some specific and important queues, like NAS and DATA\_ENGINE. (default: no). This value is extracted from the callback: list\_subscribers, field: count.
* **qos\_on\_queued** = yes: valid for type= GET/ATTACH/NAS create a QOS for the monitored queues (yes/no). This QOS will contain the number of queued qos metrics. These qos metrics will be used in the included dashboard to show you where you have possible queue bottlenecks/problems. (default: no). This value is extracted from the callback: list\_subscribers, field: npds.
* **qos\_on\_diff =** yes: valid for type=GET/ATTACH create a QOS for the # of processed alarm/qos since the previous probe execution interval. (yes/no). Default: no. This field is the difference between the field: count between 2 probe intervals.
* **alarm\_on\_queued** = 1000: alarm threshold to create an alarm or/and email. This threshold is valid for type ATTACH and NAS. Use value: 0 for no alarm. Default: 0

When you define an ”alarm\_on\_queued” on type: TEMP the generated alarm will also contain id, and subject. This will help to identify the queue.

* **Alarm\_on\_queued\_bypass** = 0, number of times the alarm for a breach of a queue threshold must be bypassed. Default: 0.
* **alarm\_on\_diff = 0:** alarm threshold to create an alarm if the #alarm/qos since previous interval was higher than threshold. This can be used to find/detect any abnormal behavior. If set to “0” or “no”, no threshold is set. Default: 0.
* **alarm\_on\_number\_of\_queues** = no: can be no or a # that will be the threshold to create an alarm if the number of queues for the type is passed. This was mainly created to be able to send an alarm when you had more than xx TEMP queues. A TEMP queue is opened hen you use IM, IM console or Dr Nimbus (yes/no). Default: no.
* **alarm\_on\_attach\_no\_consumer** = no: can be no or yes and will send an alarm when an attach queue has no active consumer, meaning that no probe is using/consuming the attach queue. Default: no.
* **alarm\_on\_attach\_no\_consumer\_bypass =** 0, number of times the alarm for an attach queue with no consumer will be bypassed. This will bypass an alarm if an attach queue had a 1 time problem if valueis set to 1. Default: 0.
* **Alarm\_on\_get\_no\_connection:** no: can be yes or no and will send an alarm if a get queue is not connected to the target attach queue. Default: no
* **Alarm\_on\_get\_no\_connection\_bypass:** 0, number of times the alarm for a get queue without a valid connection to an attach queue will be bypassed. Default: 0.
* **Alarm\_level\_hub\_queue** = 4 : alarm level (1-5) used when a hub queue breaches the defined threshold (when TYPE: ATTACH, GET or TEMP). Default: 4.
* **Alarm\_level\_nas\_queue** = 3: alarm level (1-5) used when a nas queue breaches the defined threshold (when TYPE: NAS). Default: 3.
* **Alarm\_level\_diff\_queue =** 1: alarm level (1-5) used when the diff value breaches the define dthreshold. Default: 1.
* **Alarm\_level\_number\_of\_queues** = 1 : alarm level (1-5) used when a queue count threshold is passed. Default: 1: (activated by: alarm\_on\_number\_of\_queues)
* **Alarm\_level\_get\_not\_connected:** 1:alarm level (1-5) used for the alarm when a GET queue is detected that is not connected. Set the value to **no** or ‘0’ if you don’t need/want this alarm. No mail possible (yet) for this alarm. Default: 1.
* **Alarm\_level\_attach\_no\_consumer**: 1: alarm level (1-5) used for an alarm when an ATTACH queue is detected that has no probe active that consumes the qos/alarms in the queue. Set the value to **no** or ‘0’ if you don’t want this alarm. No mail possible (yet) for this alarm. Default: 1.
* **mail\_on\_alarm\_queued** = no : do you want an email if a threshold is passed for an ATTACH or NAS queue. Here it’s important to think about the setup of the different section so that you can use this mail parameter only on some specific “name” values. (yes/no). Default: no
* **Alarm\_level\_mail** = 2: alarm level (1-5) used for the alarm that indicates that an email for an alarm is send. Default: 0.
* **ignore\_alarm\_time\_start =** ; format: hhmm. Default: none
* **ignore\_alarm\_time\_end=** ; format: hhmm. Default: none

To make it possible to ignore alarms (qos will continue to be created) for a specific queue during a time period defined via the parameters: ignore\_alarm\_time\_start and ignore\_alarm\_time\_end; both in format: hhmm (24h format).

This option was created especially for the data\_engine queue that could be blocked and queuing during the backup of the CA\_UIM database.

## Blat setup

The BLAT utility is only needed if you want to send an email for certain queuecheck alarms.

With the cfg setting “mail\_on\_alarm\_queued” you ask the probe to launch a script defined with the parameter: mail\_script.

Example bat file that uses the BLAT tool to send emails:

rem --- 1 - inactive / queue\_hub or queue\_nas

set what=%1

rem --- 2 - hub\_name, name, nas\_addr or nas\_name

set name=%2

rem --- 3 - in case of "inactive", the word: nas or hub, else the #queued

set numb=%3

rem --- path to blat and nimalarm added to normal path

path=C:/Nimsoft/bin;C:/Windows/System32;%path%

rem --- SET target email address (comma separated if multiple)

set email="user1@google.com,user2@outlook.com"

rem --- message to console & send email

if "%what%" == "inactive" (

nimalarm -l 1 "QueueCheck executing: %0 %name%: %numb% %what%"

blat - -to %email% -server uit.telenet.be -sender nimsoft@donotreply.com -body "Check your %numb%: %name% %what%" -subject "UIM Alert for %numb%: %name% %what%"

) else (

nimalarm -l 1 "QueueCheck executing: %0 queue: %name% %what% queued: %numb%"

blat - -to %email% -server uit.telenet.be -sender nimsoft@donotreply.com -body "Check your %name% %what% because number of messages queued: %numb%" -subject "UIM Alert for %what%: %name% is queueing: %numb% alarms"

)

This script is called from the probe with 3 parameters:

* where/what is the problem (inactive, queue\_hub or queue\_nas
* queue name, hub\_name or nas\_name
* number of messages/qos queued or the word nas/hub in case of inactive

Customize:

* path: where script can find BLAT and NimAlarm
* email: comma separated list of target email address

First time customization to setup BLAT:

* download blat from: <https://sourceforge.net/projects/blat/>
* copy the 4 blat modules into a directory in your path OR adapt the script so that he can find Blat.
* You need a first execution that defines the smtp server and if needed the used user and password
  + blat -install smtp\_server email@used\_to\_send\_the\_email
  + blat -install smtp\_server email@used\_to\_send\_the\_email –u smtp\_user –pw smtp\_password
* now we can send a test mail:
  + blat - -to your\_email@providor -body "test body message" -subject "test uim message"

**Note**: by using a script to send the email(s) you can use any tool and any logic in this script.

## CI\_Type setup

To create valid QoS metrics that are mapped correctly to a ci\_type, you have to execute some sql statements via SQL Server Management Studio:

You need to execute the sql statements from the file:

**doc\_ci\_type\_add.txt**

These statements will define entries in the tables: **CM\_CONFIGURATION\_ITEM\_DEFINITION** and **CM\_CONFIGURATION\_ITEM\_METRIC\_DEFINITION** so that qos definitions/metrics will be displayed correctly in the Operator Console, specially in Metric View.

insert into CM\_CONFIGURATION\_ITEM\_DEFINITION (ci\_type, ci\_parent, ci\_description) values('9.4','9','Private.Custom')

delete from CM\_CONFIGURATION\_ITEM\_METRIC\_DEFINITION where met\_type in ('9.4:40','9.4:41','9.4:42')

insert into CM\_CONFIGURATION\_ITEM\_METRIC\_DEFINITION (met\_type, met\_description, unit\_type, ci\_type) values('9.4:40','queuecheck.hub.count','count','9.4')

insert into CM\_CONFIGURATION\_ITEM\_METRIC\_DEFINITION (met\_type, met\_description, unit\_type, ci\_type) values('9.4:41','queuecheck.hub.queued','count','9.4')

insert into CM\_CONFIGURATION\_ITEM\_METRIC\_DEFINITION (met\_type, met\_description, unit\_type, ci\_type) values('9.4:42','queuecheck.nas.queued','count','9.4')

insert into CM\_CONFIGURATION\_ITEM\_METRIC\_DEFINITION (met\_type, met\_description, unit\_type, ci\_type) values('9.4:43','queuecheck.hub\_diff','count','9.4')

The probe will generate 4 different QoS metrics:

* QOS\_QUEUECHECK\_COUNT with ci\_type=9.4:40
* QOS\_QUEUECHECK\_QUEUED with ci\_type=9.4:41 (hub queues)
* QOS\_QUEUECHECK\_QUEUED with ci\_type=9.4:42 (nas queues)
* QOS\_QUEUECHECK\_DIFF with ci\_type=9.4:43 (count difference # since previous probe interval)

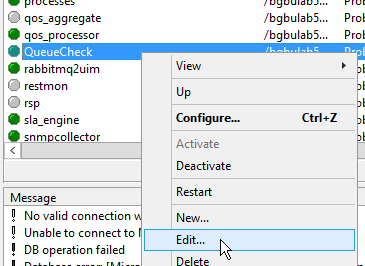
**Note**: it is recommended NOT to skip this step and execute this step BEFORE you start the probe for the first time (or better: before the first QoS metrics are created by this probe).

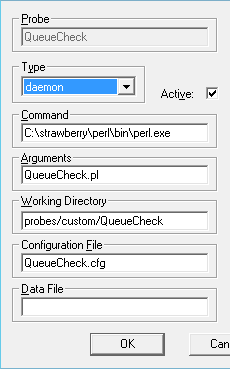
Depending where and what screen you need to use in the Operator Console 20.40 the logic (=sql query) sometimes requires the “met\_type”, and without these configuration\_item definitions you would not obtain a correct display of your qos metrics.

## Verify Probe definitions

The just deployed probe has some default definitions that are perhaps not correct in your environment.

Open in Infrastructure Management the QueueCheck probe in Edit:





* If you have a valid Perl environment on your master hub, enter here the correct path to perl.exe
* If you want to run the compiled probe, enter in the command line: queuecheck.exe and blank out the Argument line

## Start the probe

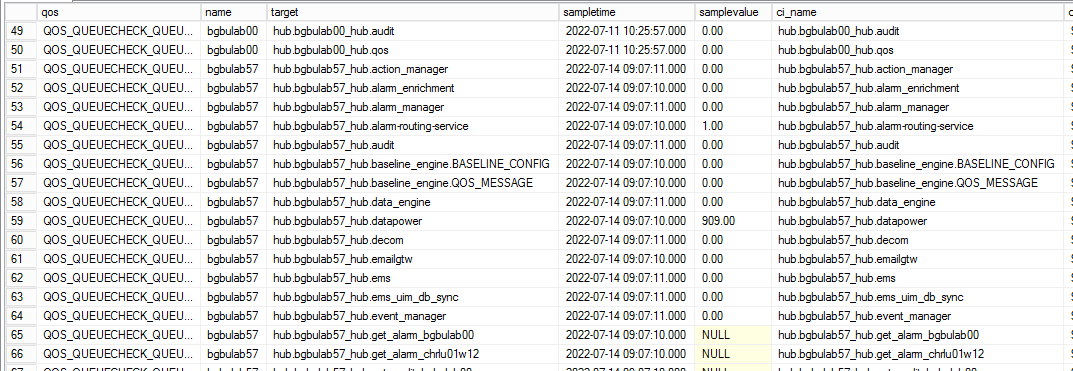
For a normal execution, once the probe starts successfully it is recommended to run with: loglevel=0.

Loglevel=1 will give you more calculation details of the difference qos that is calculated between intervals.

Loglevel=2 is useful when you report a problem.

## Check created QoS

If the probe is running you can check what QoS metrics are created by executing in “sql management studio” the query found in file: **doc\_list\_sql\_qos.txt**



## execute Qos\_View

To make reporting on QoS metrics easier the example dashboard uses a sql “view” on the generated qos.

Execute in “sql management studio” the query found in file: @qos\_views\_3.1.sql. This query will create a sql view for each qos in your environment. This query can be executed later again; it will only create a new view for new qos definitions.

Example qos view usage:

select \* from v\_qos\_queuecheck\_diff

where target like '%data\_engin%'

order by target,sampletime

## customize: mailing script

In your queuecheck.cfg file you have a parameter: mail\_script that can activate a mail in case of queue threshold or hub/nas inactive alarm.

In the utility zip file there is a sample script that can be used to send emails in case of thresholds passed: nas\_mail\_queued.bat

* The first custom variable: “email” is a comma separated list of target email addresses, defined between “”.

Example:

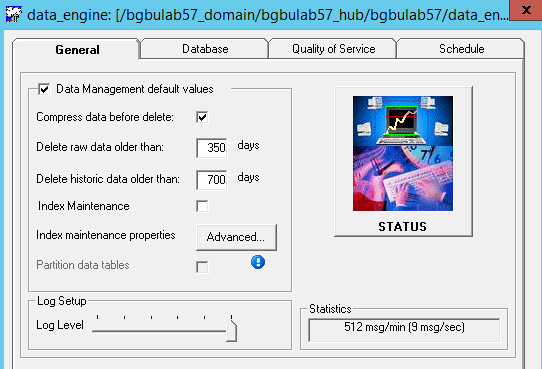
set email=”target1@gmail.com,target2@outlook.com”

* The second important custom variable is: path. Here you specify the folder where the script can find nimalarm.exe and blat.exe. In the example script, we copied blat in /nimsoft/bn.

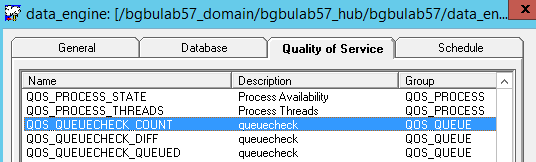
## QoS default retention

The default retention of QOS data is probably set to high for QueueCheck metrics.

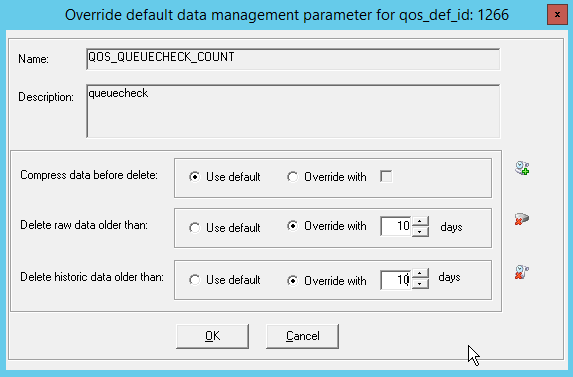
In IM, you can select the data\_engine probe:



On the “General” page you can see the retention setting valid in your environment.



On the “Quality of Service” page you can change these generic retention days for some specific QoS definitions.



In most cases you only need some days of QueueCheck data, or even only 1 day if you don’t need to create historical overview reports or graphs.

# Probe Output (QoS and Alarm)

## QoS created

* QOS: QOS\_QUEUECHECK\_COUNT Target (=ci\_name): hub.(hubname).(queuename): number of queue entries processed since start (activated by: qos\_on\_count = yes TYPE=GET/ATTACH))
* QOS: QOS\_QUEUECHECK\_QUEUED Target (=ci\_name): hub.(hubname).(queuename): number of entries queued (activated by: qos\_on\_queued = yes TYPE=GET/ATTACH)
* QOS: QOS\_QUEUECHECK\_QUEUED Target (=ci\_name): nas.(hubname).(queuename): number of entries queued (activated by: qos\_on\_queued= yes TYPE=NAS)
* QOS: QOS\_QUEUECHECK\_DIFF Target (=ci\_name): hub.(hubname).(queuename): number of qos/alarm since previous interval/cycle/loop of the probe

## Generated Alarms

The here displayed possible alarms are with the internal variable names:

* QueueCheck - hub: $hub\_addr is not active sendmail: $mail\_on\_hub\_inactive
* QueueCheck - mail script executed for: hub: $hub\_name inactive
* QueueCheck - mail script bypassed ${mail\_count.$hub\_name}/$mail\_bypass\_times for: hub: $hub\_name inactive
* QueueCheck - failed to query hub: $hub\_addr rc=$rc
* QueueCheck - hub: $hub\_name type: $type queue: $name has no consumer (GET queue or Probe is not active)
* QueueCheck - hub: $hub\_name type: $type queue: $name inactive
* QueueCheck - hub: $hub\_name type: $type queue: $name diff: $diff\_num (alarm level: $alarm\_on\_diff)
* QueueCheck - hub: $hub\_name type: $type queue: $name queued: $npds id: $id subject: $subject (alarm level: $alarm\_on\_queued mail: $mail\_on\_alarm\_queued)
* QueueCheck - hub: $hub\_name type: $type queue: $name queued: $npds (alarm level: $alarm\_on\_queued mail: $mail\_on\_alarm\_queued)
* QueueCheck - mail script executed for: hub: $hub\_name type: $type queue: $name queued: $npds
* QueueCheck - mail script bypassed ${mail\_count.$hub\_name.$name}/$mail\_bypass\_times for: hub: $hub\_name queue: $name queued: $npds
* QueueCheck - number of queues with name $queue\_ref->{name} and of type $queue\_ref->{type} at hub $hub\_name $noq > $noq\_ref
* QueueCheck - nas: $nas\_addr1 is not active sendmail: $mail\_on\_nas\_inactive
* QueueCheck - mail script executed for: nas: $nas\_addr inactive
* QueueCheck - mail script bypassed ${mail\_count.$nas\_addr}/$mail\_bypass\_times for: nas: $nas\_addr inactive
* QueueCheck - hub: $hub\_name nas\_queue: $nas\_name queue: $nas\_queued (alarm level: $alarm\_on\_queued)
* QueueCheck - mail script executed for: hub: $hub\_name type: $type queue: $nas\_name queued: $nas\_queued
* QueueCheck - mail script bypassed ${mailnas\_count.$nas\_hub.$nas\_name}/$mail\_bypass\_times for: hub: $hub\_name queue: $nas\_name queued: $nas\_queued
* QueueCheck - script file: $mail\_script does not exist
* QueueCheck - regex $reg\_var1: $reg\_var2 is not a valid regex, stop probe

## Logfile

The logfile in the /custom/queuecheck directory running with loglevel=0 can contain:

Jul 25 09:41:11:896 QueueCheck: --- start new loop execution, run: 9 ---

Jul 25 09:41:11:897 QueueCheck: alarm: level: 5 QueueCheck - hub: /bgbulab57\_domain/chrlu01w12\_hub/chrlu01w12/hub is not active sendmail: no

Jul 25 09:41:11:900 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_probe\_discovery\_chrlu01w12 inactive

Jul 25 09:41:11:901 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_probe\_discovery\_bgbulab12 inactive

Jul 25 09:41:11:903 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_audit\_chrlu01w12 inactive

Jul 25 09:41:11:904 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_qos\_chrlu01w12 inactive

Jul 25 09:41:11:907 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_alarm\_chrlu01w12 inactive

Jul 25 09:41:11:907 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_audit\_bgbulab00 inactive

Jul 25 09:41:11:909 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_qos\_bgbulab00 inactive

Jul 25 09:41:11:909 QueueCheck: alarm: level: 1 QueueCheck - hub: bgbulab57\_hub get queue: get\_alarm\_bgbulab00 inactive

Jul 25 09:41:11:920 QueueCheck: alarm: level: 2 QueueCheck - hub: bgbulab57\_hub attach queue: rabbitMQ has no consumer

Jul 25 09:41:11:924 QueueCheck: alarm: level: 2 QueueCheck - hub: bgbulab57\_hub attach queue: datapower has no consumer

Jul 25 09:41:11:926 QueueCheck: alarm: level: 2 QueueCheck - hub: bgbulab57\_hub attach queue: messagegtw has no consumer

Jul 25 09:41:11:931 QueueCheck: alarm: level: 2 QueueCheck - hub: bgbulab57\_hub attach queue: smsgtw has no consumer

Jul 25 09:41:11:956 QueueCheck: alarm: level: 5 QueueCheck - hub: /bgbulab57\_domain/bgbulab00\_hub/bgbulab00/hub is not active sendmail: no

Jul 25 09:41:11:977 QueueCheck: alarm: level: 5 QueueCheck - nas: /bgbulab57\_domain/chrlu01w12\_hub/chrlu01w12/nas is not active sendmail: no

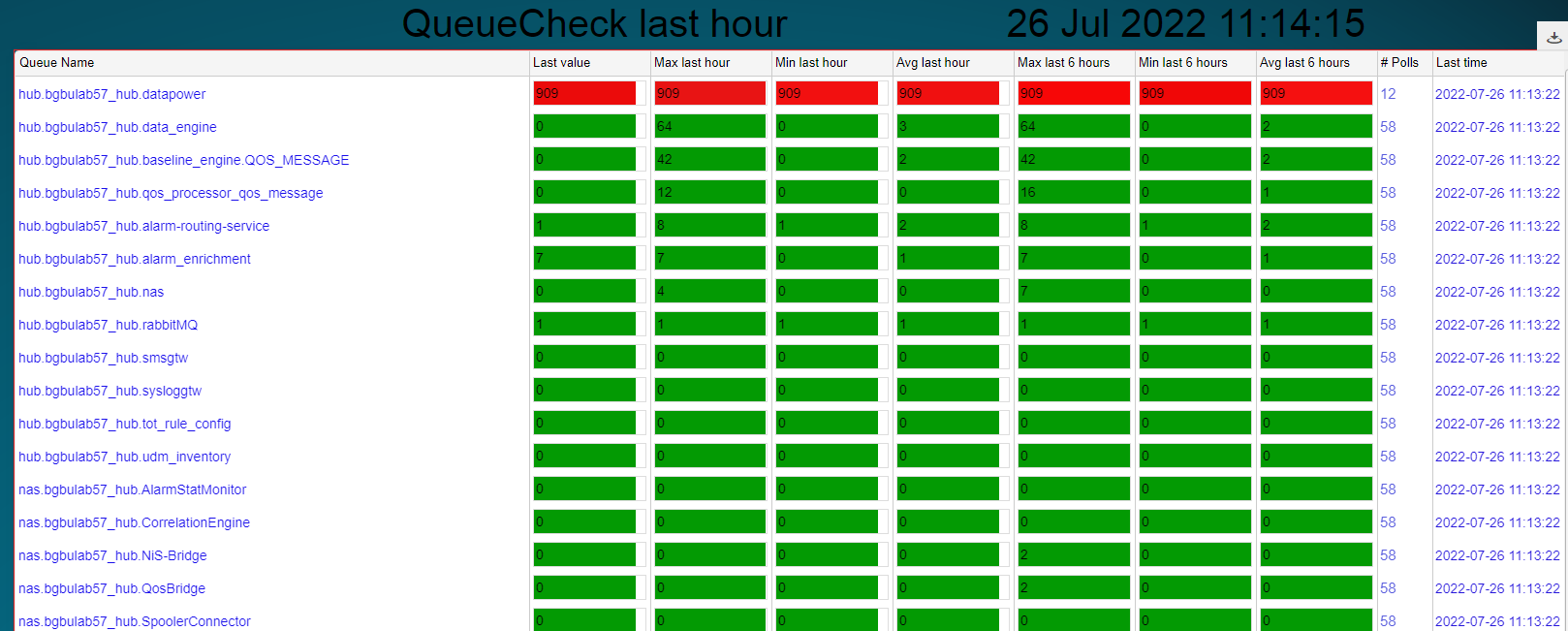
Jul 25 09:41:11:977 QueueCheck: --- end loop execution, run: 9 #alarm: 5 #clear: 0 #mail: 0 #hub: 3 #hub\_ok: 3 #hub\_nok: 0 #nas: 2 #nas\_ok: 2 #nas\_nok: 0 #ATTACH: 38 #ATTACH\_ok: 34 #ATTACH\_nok: 4 #GET: 7 #GET\_ok: 7 #GET\_nok: 0 #TEMP: 6 #NAS\_queue: 10 ------

# – QOS Usage in your Dashboard

## Dashboard import

The utilities package contains 2 dashboard examples, one for the “queued” metrics and a second one for the “diff” metrics.

In your Operator Console – Settings – Dashboard Designer you can now import the sample dashboard file: **oc\_queuecheck.zip**.



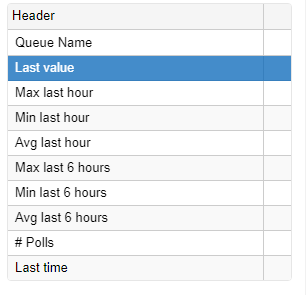
* GET queues that are not connected do have an empty value in the “last value” column. (ticket is open to have empty values a custom color).
* You will need to adapt, via “Setup – dashboard designer” the color mapping for the warning/critical levels that are suitable for your environment.



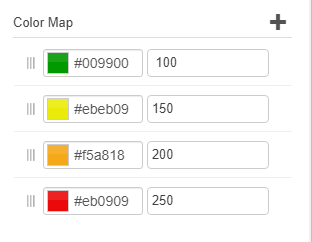
In production these values will show you the number of qos/alarms processed in the probe interval.

## Color Mapping

For each column in the example dashboard you can define a related color mapping:



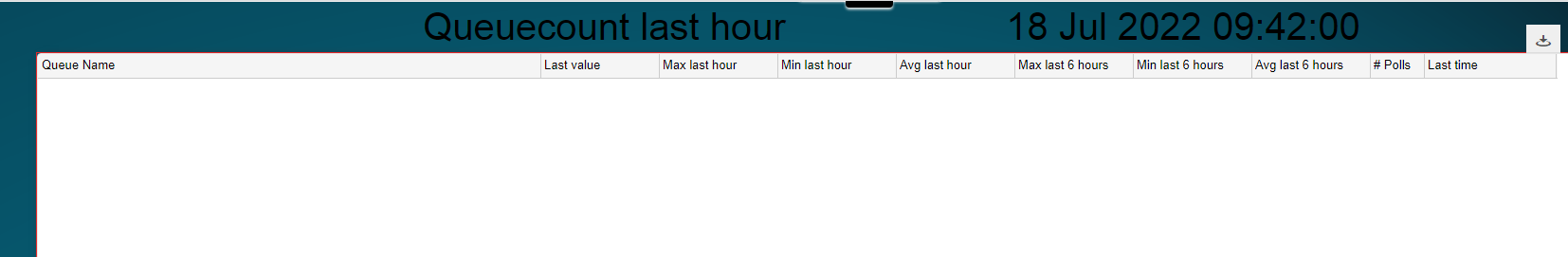
You have a related “color map” with the color thresholds:



You will have to set the color thresholds to a production value; the example contains values for a small test environment.

## Empty Dashboard

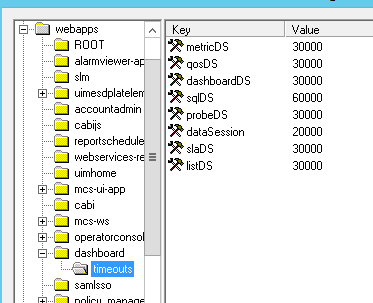
If your dashboard stays empty, like:



This can have 2 reasons:

* You didn’t execute the sql query from file: @qos\_views\_3.1.sql (see previous step)
* Your SQL response time is too slow and the query behind the dashboard didn’t end in the default timeout

To modify the SQL Dashboard timeout, open your OC Wasp probe via Raw Configure:



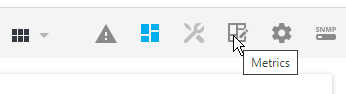
* listDS -- for all data obtained within a list
* metricDS -- for all metric data sources
* probeDS -- for all probe data sources
* qosDS -- for all QoS data sources
* slaDS -- for all SLA data sources
* sqlDS -- for all SQL data sources
* dashboardDS -- for all dashboard data sources

The value is defined in milliseconds.

# – QOS Usage in Metric View

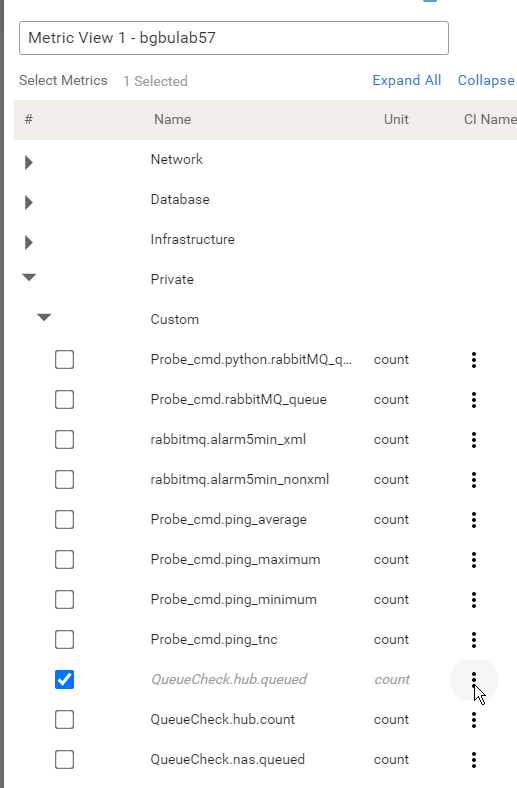
If all steps from the setup chapter are followed you will have correct QoS metrics.

If you go, via Inventory, to your main hub and select: Metrics:



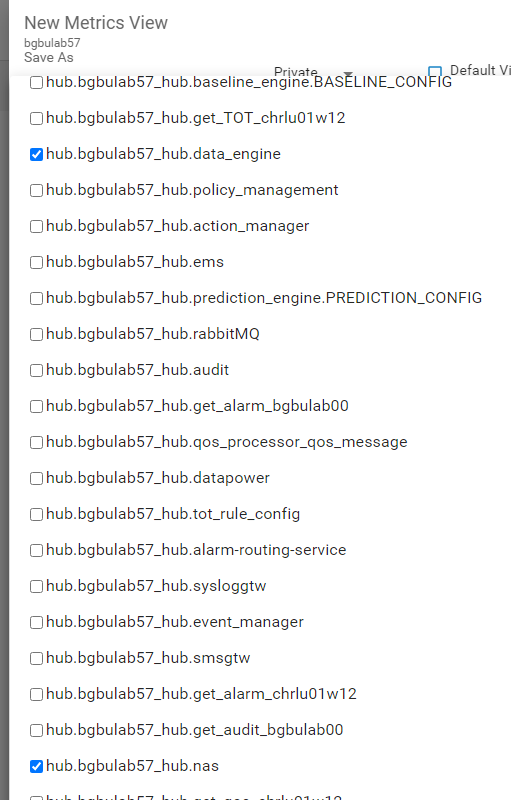
And select: New View

On this next view there is a default name generated, now it’s the time to give this a logical name.



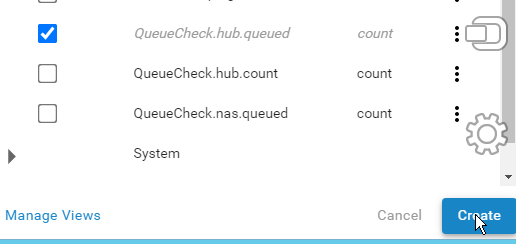
Under Private – Custom you can find all your custom QoS metrics.

If you select the 3 dots for “QueueCheck.hub.queued”, but only after selecting that qos metric, you will receive a popup with all available qos metrics for the selected qos:



Note: on this screen you can only select monitored queues if you selected a qos on previous screen.

If you come back to previous screen you can select: Create



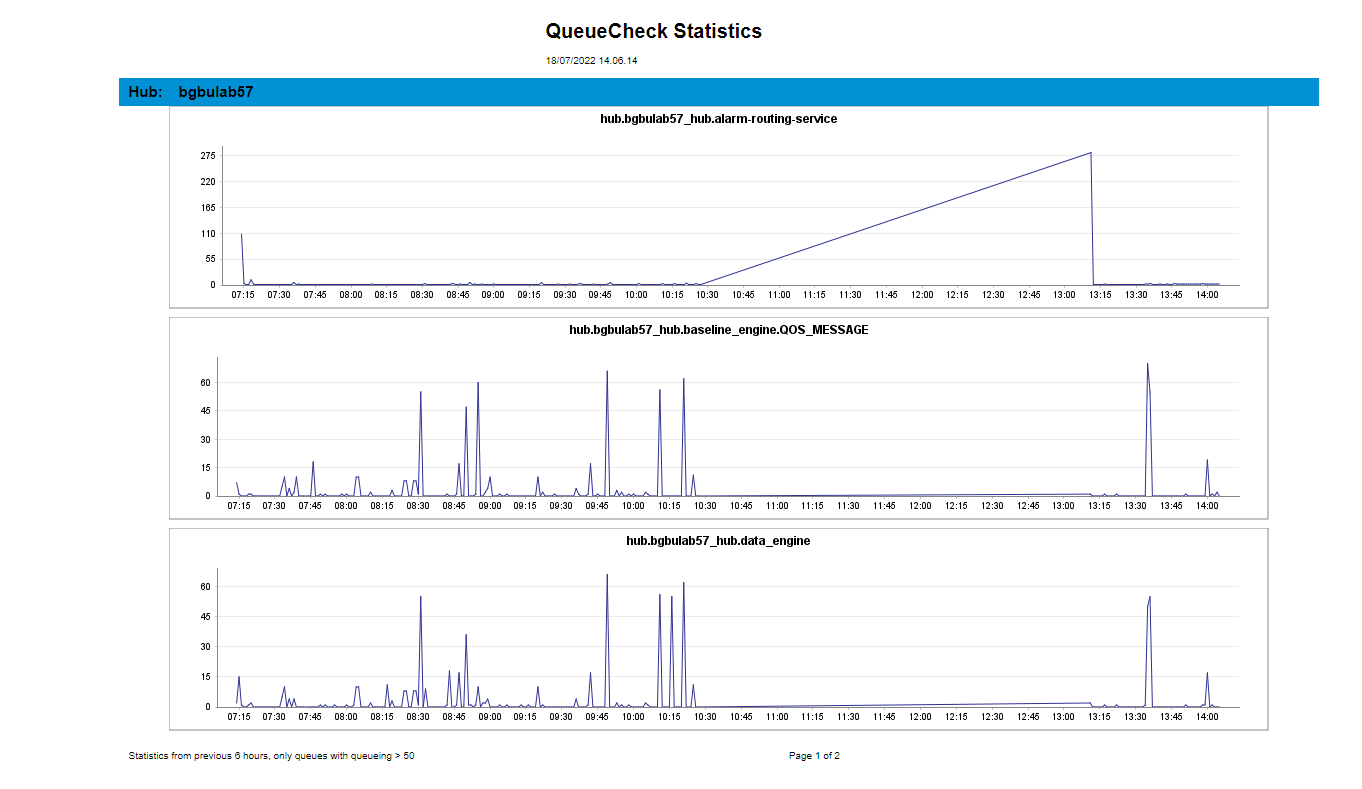
Your view with the 2 selected queues:



# - QOS Usage in CABI/Jaspersoft report

In the unzipped utilities folder you can find an example Cabi/Jaspersoft report: queuecheck\_statistics.jrxml.

Example report: (that will only list monitored queues that had a queuing > 50 the last 6 hours)

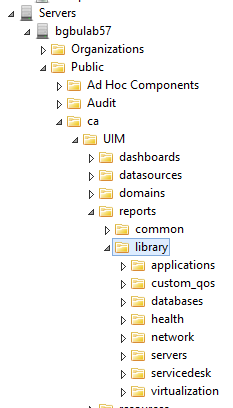


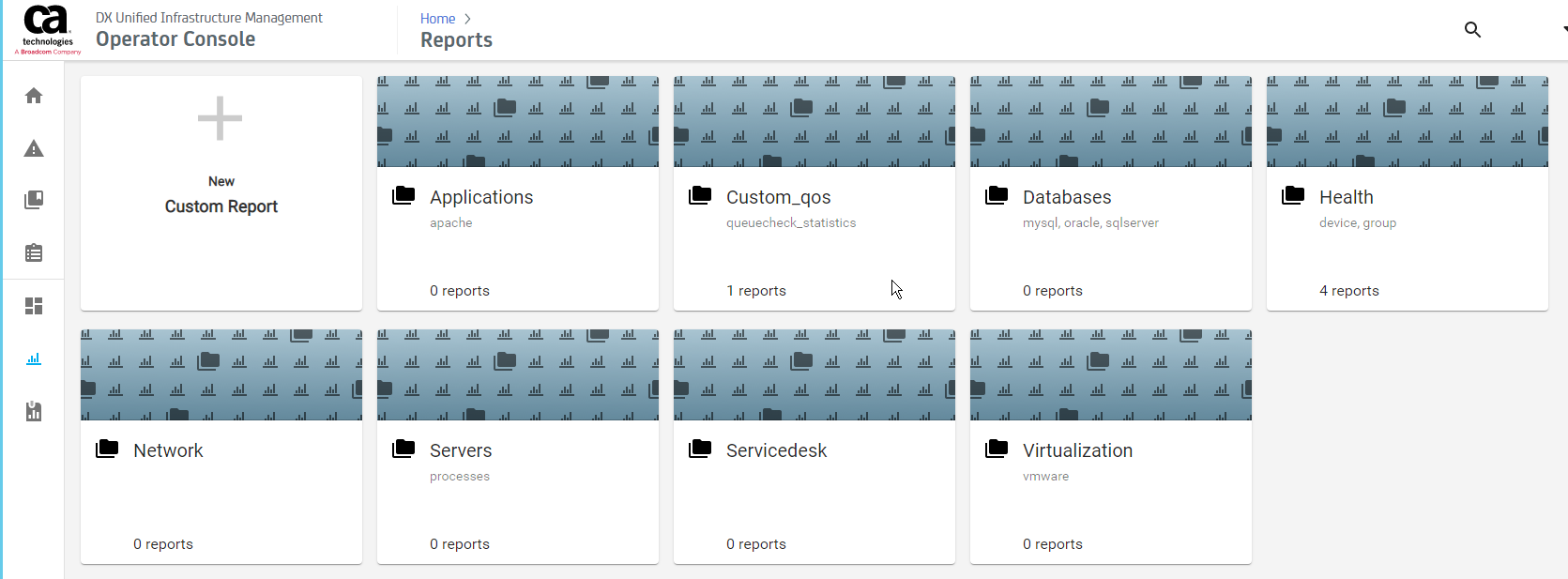
**Note**: this example report needs to be opened via Jaspersoft Studio and some parameters/definitions need to be altered, like the reference to your SQL server.

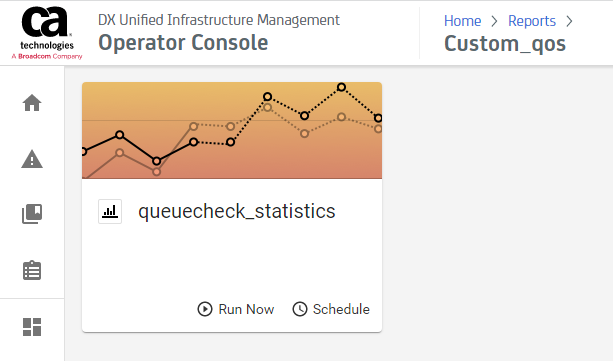
If you publish after the modification the report into a custom library under:

/Public/ca/UIM/reports/library

Your custom report will also be available for execution in the Operator Console – Reports GUI.







# - Probe Logic

* When the probe starts it will read the config file. If no queue section is defined, the probe will stop.
* Execute an sql query to get all open alarms for the probe: queuecheck
* Acknowledge each queuecheck alarm (only done when the probe is starting)
* The probe will start the loop based on the timer: interval (in seconds)
* Execute on local hub callback: **gethubs** (get list of known hubs)
* Loop in returned hublist table::
  + If hub is in hub\_exclude, bypass this hub + log
  + If hub is not active alarm/mail/log
  + Execute the hub callback: get\_info to check the field uptime. If the hub was recently (less than hub\_wait\_seconds seconds) started we will wait a probe interval to start checking the queues on that hub. This to avoid invalid alarms due to probes that are starting and not yet consuming the queues. A log message is written in that case.
  + Execute callback: **list\_subscribers** (get list of queues)
  + Loop in returned postroute table:
    - If queue address is in hub\_exclude, bypass this queue + log
    - If queue is in attach\_queue\_exclude or get\_queue\_exclude, bypass the queue + create log
    - If queue is not connected alarm/log. (possible with an alarm bypass for x intervals if: alarm\_on\_attach\_no\_consumer\_bypass or alarm\_on\_get\_not\_connected\_bypass is used)
    - Create qos metric count/queued
    - Calculate the difference between the previous count and current count and create, if activated by config parameter: qos\_on\_diff, a qos metric.
    - If threshold breached: alarm/mail/log (possible with an alarm bypass for x executions if: alarm\_on\_queued\_bypass is used)
* Is there a TYPE: NAS defined in queue section config file
  + Sql query to obtain all robots where nas probe is installed
  + Loop in returned nas address list
    - If nas inactive alarm/mail/log
    - If nas address is in nas\_exclude bypass this nas
    - Execute callback: **get\_info** (list of internal nas queues)
    - Loop in returned pub\_subscribers table:
      * Create qos metric queued
      * If threshold breached: alarm/mail/log (possible with an alarm bypass for x executions if: alarm\_on\_queued\_bypass is used)
* Write final log file line with totals: #alarm #mail #clear #hub #hub\_ok #hub\_nok #nas #nas\_ok #nas\_nok #attach #attach\_ok #attach\_nok #get #get\_ok #get\_nok #temp #nas\_queue

**Note**: for each alarm that the probe creates an internal variable is created so that the probe knows later if this alarm needs to be cleared.

During startup the probe will send a clear for all hubs, queues, nas that are good, this to clear possible previous alarms.

# - Example scenarios

This probe can be used in several different ways, but all scenarios are defined via the config file: queuecheck.cfg

## Only create Qos for queued with no alarms

A first approach can be to let the probe collect all info and create only the Qos metrics for the queuing with no alarm messages.

<setup>

<queues>

<0>

name = .\*

type = ATTACH

qos\_on\_queued = yes

</0>

<1>

name = .\*

type = GET

qos\_on\_queued = yes

</1>

<2>

name = .\*

type = NAS

qos\_on\_queued = yes

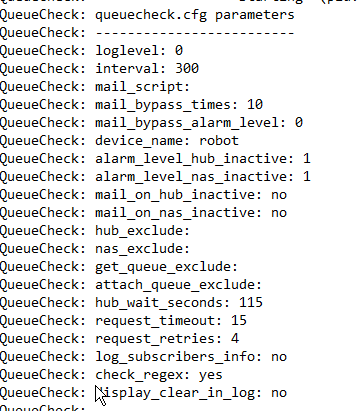
</2>

</queues>

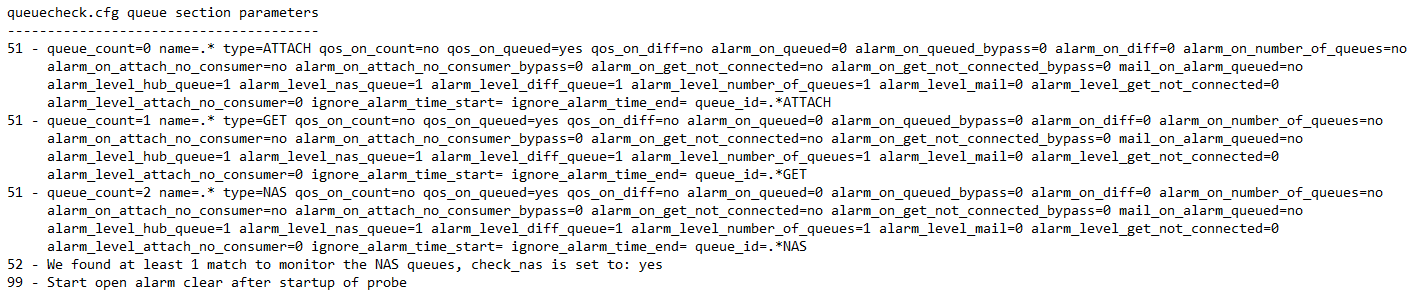
</setup>

In this very simple config file we only used the parameters that are different from the defaults. When you start the probe with this config set, you will see in queuecount.log the details of all used parameters:

Part one are the setup parameters:



Part two are the defined queue sections:



By default the probe will ONLY create an alarm for an Inactive hub or nas, this with a default level: 1. This behavior can be changed by adding in the <setup> section the parameters:

* alarm\_level\_hub\_inactive = 0
* alarm\_level\_nas\_inactive =0

After some intervals you can use the following SQL query to check if your Qos metrics are created correctly: (see file: doc\_list\_sql\_qos.txt)

SELECT qd.qos, cs.name, qd.target, qs.sampletime, qs.samplevalue, qd.ci\_metric\_id, qd.probe, ci.ci\_name, cim.ci\_metric\_type, cid.ci\_type, cid.ci\_description, cimd.unit\_type

FROM cm\_computer\_system cs with(nolock)

INNER JOIN

cm\_device d with(nolock) ON d.cs\_id = cs.cs\_id

INNER JOIN

cm\_configuration\_item ci with(nolock) ON ci.dev\_id = d.dev\_id

INNER JOIN

cm\_configuration\_item\_definition cid with(nolock) ON cid.ci\_type = ci.ci\_type

INNER JOIN

cm\_configuration\_item\_metric cim with(nolock) ON cim.ci\_id = ci.ci\_id

INNER JOIN

s\_qos\_data qd with(nolock) ON qd.ci\_metric\_id = cim.ci\_metric\_id

INNER JOIN

s\_qos\_snapshot qs with(nolock) ON qd.table\_id = qs.table\_id

LEFT JOIN

cm\_configuration\_item\_metric\_definition cimd with(nolock) ON ci\_metric\_type = cimd.met\_type

where qd.probe like 'queuecheck%'

order by qd.qos,qd.target

Now you can also use the example dashboard (oc\_queuecheck.zip) to see your Qos metrics in an overview table.



## Create Qos for queued with alarm if threshold is breached

Based on the minimal configuration in previous example we need to add some extra parameters with thresholds:

<setup>

<queues>

<0>

name = .\*

type = ATTACH

qos\_on\_queued = yes

**alarm\_on\_queued = 20000**

</0>

<1>

name = .\*

type = GET

qos\_on\_queued = yes

**alarm\_on\_queued = 20000**

</1>

<2>

name = .\*

type = NAS

qos\_on\_queued = yes

**alarm\_on\_queued = 1000**

</2>

</queues>

</setup>

The idea of running first the example of step 1 is that this would create some qos metrics that you can use as baseline/example to define the thresholds.

By default these alarms will be generated with level:1 (Info).

## Create Qos for queued with alarm and with exceptions

There are 2 ATTACH queues that will need extra attention and probably different threshold settings: NAS and DATA\_ENGINE.

These are the 2 ATTACH queues that are critical for your UIM performance and need a permanent attention.

We add these 2 queue sections to the previous scenario.

<3>

name = data\_engine

type = ATTACH

qos\_on\_queued = yes

qos\_on\_diff = yes

alarm\_on\_queued = 10000

alarm\_on\_queued\_bypass = 1

</3>

<4>

name = nas

type = ATTACH

qos\_on\_count = yes

qos\_on\_queued = yes

qos\_on\_diff = yes

alarm\_on\_queued = 1000

alarm\_on\_queued\_bypass = 1

</4>

Here we add:

* Different threshold for data\_engine and nas queue
* If a threshold is breached we will suppress/bypass the first occurrence of the alarm (= alarm\_on\_queued\_bypass)
* Create also qos: qos\_queuecheck\_diff. This qos will contain the # of alarms/qos that are processed in each probe interval

# - Nimsoft\_generic.dat (settings)

All custom reporting tools use a common parameter file: **nimsoft\_generic.dat**

# -- UMP server & port & http/https

**uim\_server**=ump\_server\_name

**uim\_port**=80

**uim\_https**=http

# --- Nimsoft userid and crypted password (via nimsoft\_crypt.exe)

**uim\_user**=administrator

**uim\_password**=gWL/M/ij/

# --- Nimsoft domain, hub and robot to create address to the main hub

**uim\_domain**=xxx\_domain

**uim\_hub**=xxx\_hub

**uim\_robot**=xxx

# --- Wasp realm (used by Rest and UIMAPI) pre 20.4: NimJAAS from 20.4: wasp-engine

uim\_realm=wasp-engine

# --- SQL server, userid, crypted password (via nimsoft\_crypt.exe) and databasename

**sql\_server**=sql\_server\_name

**sql\_user**=sa

**sql\_password**=gWL/M/ij/

**sql\_db**=CA\_UIM

**sql\_type**=mssql

# - sql\_driver: "SQL Server" (=default) or a manual installed newer driver, example: "ODBC Driver 17 for SQL Server" (\* no quotes around driver name \*)

**sql\_driver**=SQL Server

# - in case you need to use a not current logged on Windows user

**sql\_dsn**=

# --- end of parameters ---

The 2 passwords are stored in an encrypted form. To generate this encrypted password, use:

nimsoft\_crypt.exe your\_password

As output, you will receive the string that you can copy as encrypted password in the above file.

**Note1**: this nimsoft\_generic.dat is common between multiple tools. It is possible that the tool you are working with is not using all variables in this file.

**Note2**: if you use "sql\_user=trusted" and use also "sql\_password=" we will connect to MSSQL via a trusted connection. (=your logged on userid)

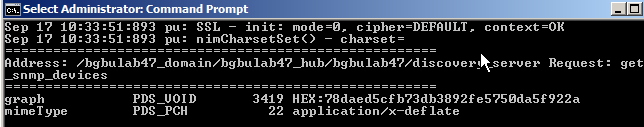
**Note3**: the uim\_robot must be defined in the format/case that UIM recognize them. (like it's displayed in IM)

**Note4:** sql\_driver: previously we used the hardcoded sql driver “SQL Server” that is by default installed in Windows systems. But when you want to connect to TLS 1.2 MSSQL you need to install (manually) a newer MSSQL driver (example: ODBC Driver 17 for SQL Server)

**Note5**: In case you receive a "communication error" while using a non-simulation execution of the tool, try to use the: /uim\_domain/uim\_hub/uim\_robot values like you use it in nimsoft\_generic.dat in the command:

pu -u administrator -p ??? /bgbulab47\_domain/bgbulab47\_hub/bgbulab47/discovery\_server get\_snmp\_devices

This commands must give a normal output like:



These names are case sensitive.

**Note6:** in case you want to use a user defined system DSN to perform the logon the MSSQL you can use the sql\_dsn parameter. Else keep this parameter blank.

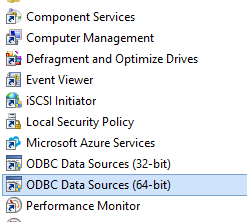
**Note7:** uimapi calls before 20.40 did use a realm: NimJAAS. But uimapi 20.4 introduced an undocumented change into realm: wasp-engine. This value is used by some of the utilities that use UIMAPI.

## SQL driver

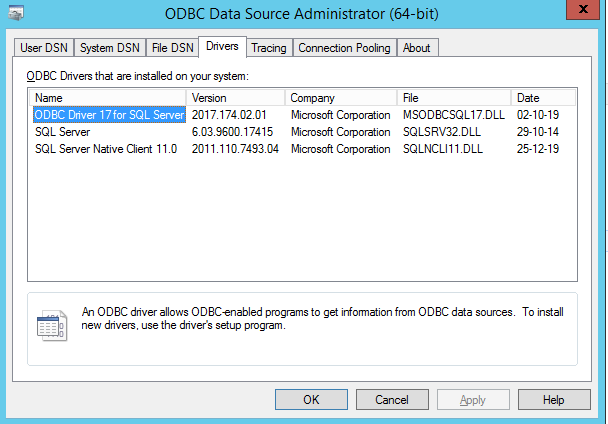
By default you will find the SQL driver: “SQL Server” installed on your Windows server.

But if you want/need to use TLS, it’s possible that you need a newer/updated version of the driver.

With the parameter: sql\_driver you can than enter the driver name, **without** surrounding quotes, that you installed.



Drivers can be found under: Administrative tools – ODBC Data Sources.



Under the tab: Drivers you can find the installed and available SQL drivers that you can use. You need to enter the “Name” as value of the parameter.

## ODBC System DSN

In nimsoft\_generic.dat you have several possibilities to define access to the CA\_UIM database.

The easiest way is to use an internal MSSQL user:

* sql\_user: sql userid
* sql\_password: password value encrypted with nimsoft\_crypt

If you want to access the CA\_UIM database with the current logged on user, you can use:

* sql\_user: trusted
* sql\_password:

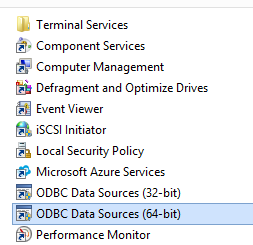
(the value kept empty)

If you want to use a Windows user to logon to SQL, not the current logged on user, you must use a pre-defined ODBC System DSN. The name you give to this System DSN is entered as value:

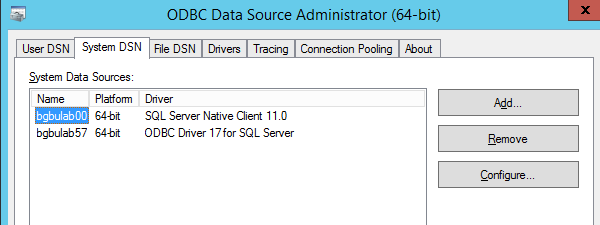
* sql\_user: windows user
* sql\_password: encrypted windows password
* sql\_dsn: name given to the ODBC System DSN

**Note1:** If you use one of the tools as probe or in a probe with a logon as Windows User, you **MUST** run the Nimsoft Service also with a Windows User (not needed to be the same Windows User)

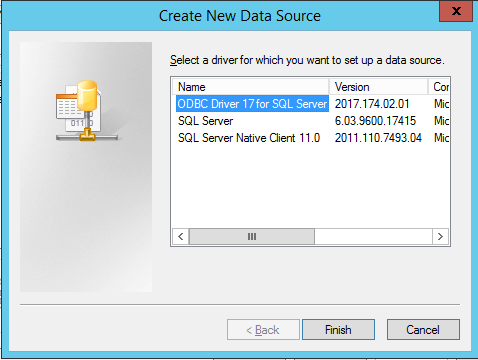
Under Administrative tools you can find:



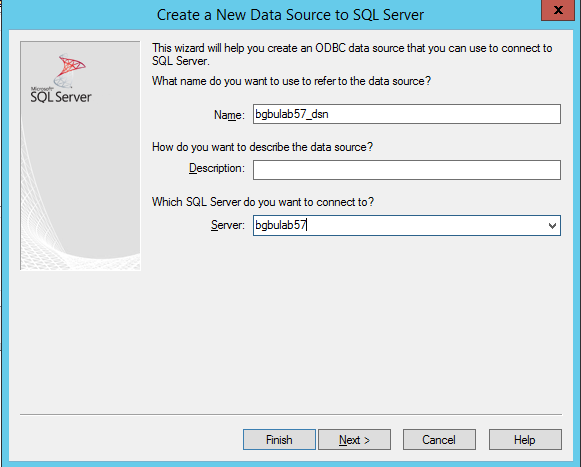
Select: ODBC Data Sources



Select the tab: System DSN - Add

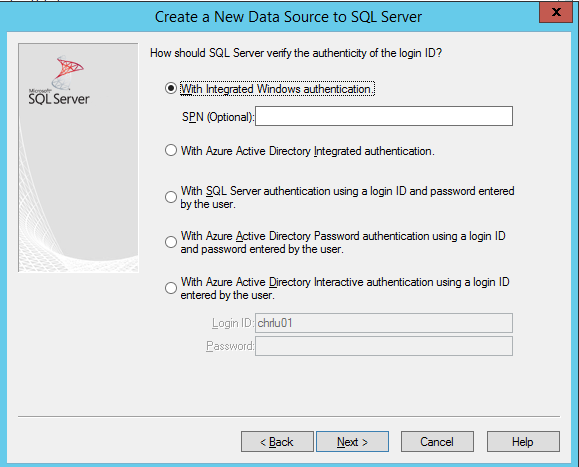


Select you ODBC driver and double click on that driver name:

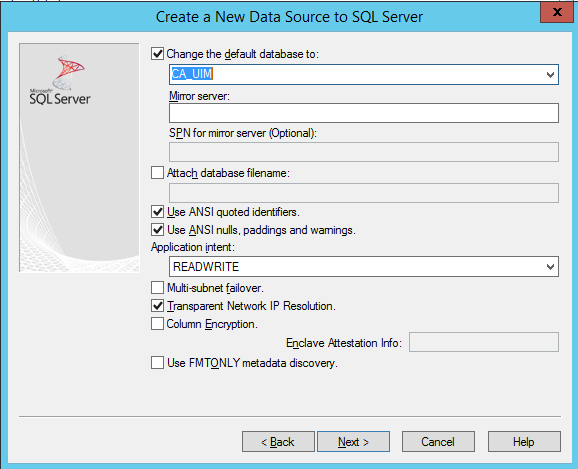


It’s the “name” that you will define, in this case: bgbulab57\_dsn, that you need to enter as value in the sql\_dsn parameter.

Select: next (and keep the defaults on that screen)

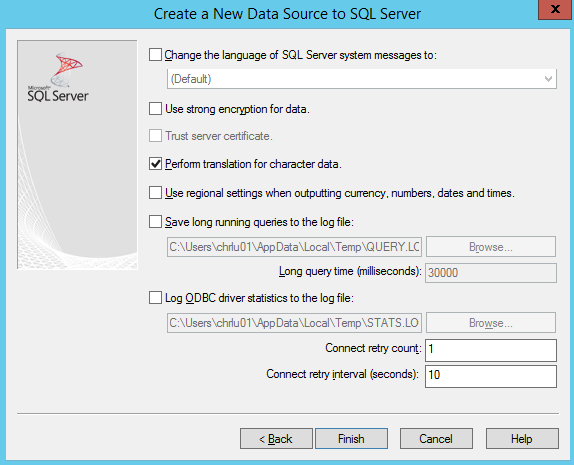


Select: next

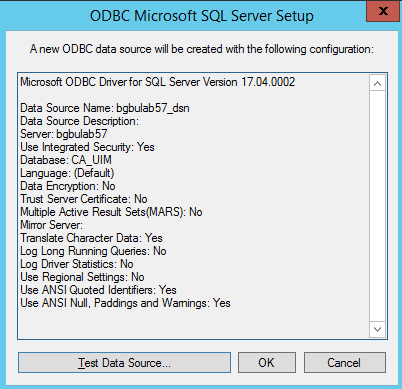


Change the default database to: **CA\_UIM**

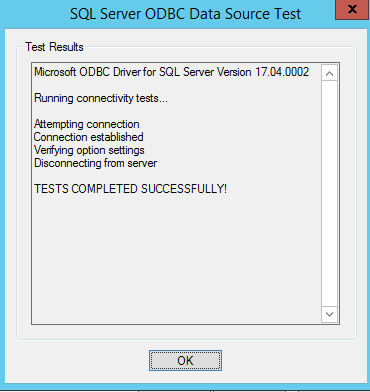
Select: next (and keep the defaults on the next screen)



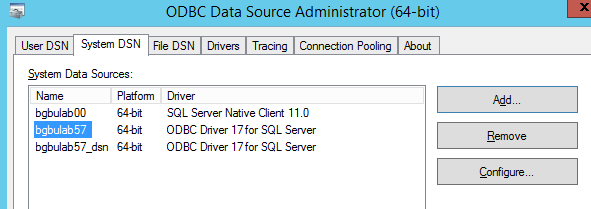
Select finish:



Select “test data source”:



Your new System DSN is now defined:



# - Create Windows Perl environment

This tool contains the Perl source and compiled Perl.

If you want to create a Perl environment that can run this Perl source & optionally compile the source yourself, you can follow the documented steps.

Once the Perl environment is created you can compile the Perl source:

pp -C -o c:\unibat\queuecheck.exe c:\unibat\queuecheck.pl

pp -C -o c:\unibat\queuecheck\_mysql.exe -l="C:\strawberry\c\bin\libmysql\_\_.dll" c:\unibat\queuecheck.pl

**Note1**: you must execute this PP command from a command prompt with as directory where you placed:

* nimsoft\_check\_package\_version.pl
* nimsoft\_generic.pm

**Note2:** the -l option is needed to include all dll modules to be able to execute the compiled module on an external server without Perl installed.

* + download from: http://strawberryperl.com/releases.html the file:
    - strawberry-perl-5.14.2.1-64bit.msi (it's a **must** that you download version 5.14.2)

**Note:** with UIM 20.40 a newer Perl SDK was released that support also Perl 5.32 (strawberry-perl-5.32.0.1-64bit.msi)

* + install the msi in: c:\Strawberry64 (as an example)
  + verify that the following directories are in the system path:
    - C:\strawberry64\perl\bin
    - C:\strawberry64\perl\site\bin
    - C:\strawberry64\c\bin
  + refresh the command prompt or reboot the activate the path
  + deploy the probe: SDK\_Perl to the main UIM server (this creates: C:\Program Files (x86)\Nimsoft\perllib)

**Note:** UIM package: SDK\_PERL version 20.40\_HF (or higher) can be used with Perl 5.32

* + copy the directories under: C:\Program Files (x86)\Nimsoft\perllib to C:\strawberry64\perl\lib
  + now you are ready to install additional packages via cpan (from command prompt): (you need internet access because these modules are get directly from cpan):
    - * cpan install Crypt::RC4
      * cpan install XML::Simple
      * cpan install HTTP::Request
      * cpan install DBD::ODBC (here you will receive some messages, but it's normal)
      * cpan install MIME::Base64
      * cpan install Time::Piece
      * cpan install Time::Seconds
      * cpan install LWP::UserAgent
      * (cpan install -f WWW::Mechanize) (generates messages and can take a long time to complete)
      * cpan install PAR::Packer
      * cpan install MIME::Lite
      * cpan install Data::GUID
      * cpan install Sys::HostAddr
      * (cpan install XML::LibXML)
      * cpan –fi Statistics::LineFit
      * cpan install Math::Spline
      * cpan install DBD::CSV
      * (cpan install DBD::Chart)
      * cpan install Net::SNMP
      * pip http://strawberryperl.com/package/kmx/perl-modules-patched/Crypt-OpenSSL-Random-0.04\_patched.tar.gz
      * (cpan -fi Net::SSL::ExpireDate) (generates a lot of messages and can take some time)
      * cpan install Win32::Console
      * cpan install Term::ANSIColor
      * cpan install Win32::Console::ANSI

Note: the lines between () are not needed to run the probe or utility but are packages that are also installed on the original Perl directory received from CA services. (and can be needed if you run other customized reports/tools/probes)

# – Possible Problems

## QoS data retention

By default the data retention for all QoS metrics is set relative high. But for QueueCheck qos data we only need these detail data for e very short period, in most cases only 1 day.

If you changed the qos retention in the data\_engine, like explained in chapter 1.12 you can force old queuecheck data to be deleted via the command:

perl nimsoft\_qos\_delete.pl -pi"queuecheck" -a"1" -k"y"

## S\_QOS\_Snapshot dtata retention

The table: S\_QOS\_SNAPSHOT contains the last value for each metric. But if your UIM environment is not very recent, there is a possibility that S\_QOS\_SNAPSHOT contains last values for metrics that don’t exist anymore.

With this query you can have a quick look what this table contains:

select \* from S\_QOS\_SNAPSHOT

order by sampletime

In most cases you will remark that you have a lot of data sampletimes that are very old.

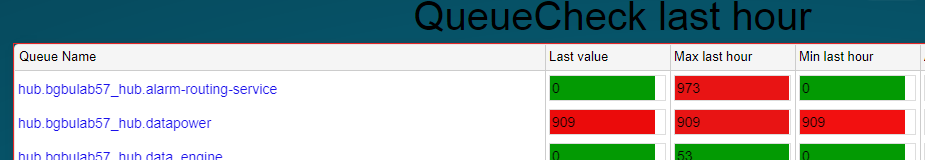
Example to delete all data older than 1 year. Remark that this table only contains the “last value” of each qos.

delete from S\_QOS\_SNAPSHOT

where sampletime < DATEADD(MONTH, -12, GETDATE())

## Trellis not processing alarms

In some exceptional cases the probe will detect abnormal queuing.



In this case Trellis was not processing alarms due to the unexpected deploy of a newer JAVA\_JRE package.

## log messages 71 &72

If you remark in the log file (queuecheck.log) that you receive for some hubs alarm and/or qos ATTACH queue the messages identified with # 71 and/or 72 this can indicate some possible problems.

**71:** this indicates that the remote hub is (probably) restarted. From the hub callback get\_info the probe will remember the previous uptime of that hub and compare that value with the current uptime. If the current (=last) uptime is lower than the previous uptime the probe will generate a log message identified with #71.

**72**: each run the probe remembers the count for each queue. Count is the number of qos/alarm message processed by this queue since the start. But sometimes the current count is less than the previous count. This means that something happened at the remote hub.

We remarked that this happened more frequently if the hub controller had the parameter: **suspend\_on\_loopback\_only** set to yes. When we changed this to no, the queue reset was gone.