CA Nimsoft Monitor

Probe Guide for Data Engine

data_engine v7.9 series

CA Nimsoft Monitor Copyright Notice

Contact CA

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

Send comments or questions about CA Technologies Nimsoft product documentation to nimsoft.techpubs@ca.com.

To provide feedback about general CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Overview 9

How the data_engine Probe Collects and Maintains QoS Data 9
RN_QOS_DATA Table Columns 10
RN_table Indexes 10
Data_engine Start Up 10
    Parallel Mode 11
    Serial Mode 12

Chapter 2: Configuration Tasks 13

Prerequisites for SQL Authentication on Microsoft SQL Server 13
Change the Database Connection Properties 14
Configure the Data Retention Settings 14
Override the Data Retention Settings on Individual QoS Objects 15
Set up Index Maintenance (MS SQL Server) 15
Set up Partitioning for Raw Sample Data (MS SQL Server) 16
Schedule Database Maintenance 16

Chapter 3: Best Practices 17

Chapter 4: Configuration Details 19

data_engine 19
Database Configuration 20
    MySQL 21
    Microsoft SQL Server 22
    Oracle 23
Quality of Service 24
Scheduler 25

Chapter 5: Troubleshooting 27

Viewing the Log File 27
Corrupted QoS Definition Values 28
Out of date information in the Quality of Service Type Status Table 30
    Table Statistics for Oracle 30
    Table Statistics for MySQL 31
    Table Statistics for Microsoft SQL Server 31

Check Partitioning Jobs 34

Documentation Changes

This table describes the version history for this document.

Version	Date	What's New?
7.9	June 2014	Added information for daily tables and Advanced Configuration items.
7.9	March 2014	Added information for new GUI items.
7.9	March 2013	Initial release of Admin Console probe GUI. (Previous versions of this probe are only configured using Infrastructure Manager).

Related Documentation

Documentation for other versions of the data_engine probe

The Release Notes for the data_engine probe

Monitor Metrics Reference Information for CA Nimsoft Probes (http://docs.nimsoft.com/prodhelp/en_US/Probes/ProbeReference/index.htm)

Chapter 1: Overview

The data_engine probe manages and maintains data that is collected by Quality of Service (QoS) enabled probes. The data_engine creates all tables and stored procedures necessary to manage the collected data.

Data that is produced by the QoS probes is stored in the NIS database, in tables named raw data tables (table prefix is RN_). Raw data is kept for a user-defined period, then compressed and aggregated into Hourly data (HN_tables). Hourly data is then compressed and aggregated into Daily data (DN_ tables).

How the data_engine Probe Collects and Maintains QoS Data

During data collection, data_engine performs the following actions:

1. The data_engine probe receives a QoS definition from a probe.
2. The data_engine probe queries the S_QOS_DEFINITION table to see if the QoS type in the message already exists in the table (for example, QOS_CPU_USAGE).
3. If the QoS type does not exist, a new entry is added to the S_QOS_DEFINITION table. New RN and DN tables are created to store monitoring data from the probe.
4. Once the first qos_message from the probe arrives, the data_engine probe adds QoS object data into the S_QOS_DATA table. The S_QOS_DATA table contains the object data for each unique combination of qos, source, and target attributes.
5. The data_engine probe inserts the raw data from the probes into the appropriate RN tables.
6. During the scheduled maintenance runs, the data_engine probe compresses and aggregates raw data from RN tables into hourly data that is stored in HN tables.
7. Older RN data is purged based on a user-defined period.
8. The data_engine probe then compresses and aggregates hourly data from HN tables into daily data that is stored in DN tables.
9. Older HN data is purged based on a user-defined period.
10. Last sample value coming from the probes for each qos object is populated in the S_QOS_SNAPSHOT table. This data is used to provide fast QoS data access for UMP portlets.

RN_QOS_DATA Table Columns

RN_QoS_Data_tables hold raw QoS data. QoS data is written once and never updated.

Column Name	Description
TableID	unique identifier; key for looking up time series data
Sampletime	time the sample was taken
Samplevalue	QoS value
Samplestdev	standard deviation of the sample
Samplerate	Rate of sampling
Tz_offset	time zone offset

RN_table Indexes

The default indexes in RN_tables are optimized for writing data:

Index	Description
Idx1	table_id, sampletime, samplevalue, samplerate, tz_offset
Idx2	table_id, sampletime

The RN_QoS_DATA_tables do not have primary keys, as both tableID and sampletime can be duplicated.

Data_engine Start Up

When the data_engine probe starts, it loads both S_QOS_DATA and S_QOS_DEFINITION into memory and establishes a bulk connection to the database.

The data_engine probe has two modes:

- Parallel Mode (see page 11)
- Serial Mode (see page 12)

Parallel Mode

Note: We recommend running the data_engine probe in parallel mode to improve data_engine probe performance.

If the thread_count_insert parameter is set to a number greater than one in the Raw Configure menu, the data_engine probe uses parallel mode.

In parallel mode, the data_engine uses multiple threads to do work in parallel. In this mode, the data_engine:

1. Continuously reads data from the hub.
2. Stores the data in a shared list and begins reading more messages.
3. Iterates over all lists using another thread to see if any need to be flushed to the database.

   Note: The actual writing of data does not happen now.
4. All objects are marked as ‘ready to commit’ and a reference is placed onto another list.
5. Concurrently, a thread continuously runs to validate, sort, and place messages into a list that is written to the database.
6. Writes any messages that have been marked as 'ready to commit data' to the database using a thread pool of worker threads.

This process allows the data_engine probe to write many rows to the database in parallel. Also, the data_engine is not as vulnerable to performance issues when inserting into one table is slow.

For more information about setting parallel mode, see Best Practices (see page 17).

Serial Mode

Serial mode is the original mode for the data_engine probe. If the thread_count_insert parameter is set to zero in the Raw Configure menu, the data_engine probe defaults to serial mode.

In this mode, the data_engine:

1. Reads messages from the hub for a given time period (default around 1 second).
    a. From 1 through 20 messages are read at a time, depending on how many are in the queue or if the hub queue size has changed (default 20).
    b. The read messages are then validated and sorted into lists that can be quickly inserted into the database.
2. Stops reading new messages and iterates over all the lists, checking to see if any are full. By default the list is flushed if it contains more than 5000 messages or if it has not been flushed in the last 5 seconds.
3. Goes back to reading messages from hub. If one bulk object takes too long to insert, then the writing of all data to the database is delayed.

Chapter 2: Configuration Tasks

If you are using SQL Authentication on Microsoft SQL Server, review the Prerequisites for SQL Authentication on Microsoft SQL Server. (see page 13)

In the Admin Console data_engine configuration GUI, you can:

- Change the Database Connection Properties. (see page 14)
- Configure the Data Retention Settings. (see page 14)
- Override the Data Retention Settings on Individual QoS Objects. (see page 15)
- Schedule Database Maintenance. (see page 16)
- Set up Partitioning for Raw Sample Data (MS SQL Server) (see page 16)
- Set up Index Maintenance (MS SQL Server) (see page 15)

To open the data_engine configuration GUI:

1. In the Admin Console navigation tree, click the down arrow next to the hub, then the robot the data_engine probe resides on.
2. Click the down arrow next to the data_engine probe, select **Configure**.

Prerequisites for SQL Authentication on Microsoft SQL Server

If you are not using the System Administrator (sa) login and require SQL Authentication on Microsoft SQL Server, your user account must have the following permissions:

- The db_owner database role for the NimsoftSLM database.
- Read and update permissions on the master and tmpdb system databases.
- The serveradmin database role to create and execute stored procedures properly.

Change the Database Connection Properties

Important! The database connection properties should only be changed in limited circumstances such as recovery operations. Changing the Database Vendor can cause connection issues. If you are changing database vendors, CA recommends reinstalling NMS.

Follow these steps:

1. In the data_engine probe configuration menu, click the **Database Configuration** folder.
2. Click the **Connection Information** drop-down list, select the Database Vendor for your database.

   The **Connection Information** field displays the connection options for the selected vendor.
3. Enter the connection settings. Settings are different for each database vendor. See Database Configuration (see page 20) for more information about the fields that are required for each vendor.
4. Click the **Test Connection** button.

   The **Test ADO Connection String** window appears. If the connection is good, the Connected and Ping values are set to yes.
5. Click **Save**.

   The configuration changes are saved and the probe is restarted.

Configure the Data Retention Settings

You can change the data retention settings to meet your auditing or security requirements.

Follow these steps:

1. In the data_engine probe configuration menu, click the **data_engine** heading.
2. Change the desired retention settings in the General Section. See data_engine (see page 19) for more information about each field.
3. Click **Save**.

   The configuration changes are saved and the probe is restarted.

Override the Data Retention Settings on Individual QoS Objects

You can override the data retention settings for individual QoS items.

Follow these steps:

1. In the data_engine probe configuration menu, click the **Quality of Service** folder.
2. In the Quality of Service Table, click the row of the QoS metric you want to modify.
3. Change the desired retention settings. See Quality of Service (see page 24) for more information about each field.
4. Click **Save**.

   The configuration changes are saved and the probe is restarted.

Set up Index Maintenance (MS SQL Server)

Important! It is not possible to rebuild the index for single partitions prior to SQLServer 2014. You can only reorganize individual partitions. Performing automatic indexing for large tables from the data_engine is discouraged, as indexing may not complete in a reasonable amount of time.

If you are using MS SQL Server, you can set up Index Maintenance to improve the speed of data retrieval operations.

Follow these steps:

1. Click the Database Configuration folder.
2. Select the **Index Maintenance** box.
3. Change the desired Index Maintenance options. See Microsoft SQL Server (see page 22) for more information about each field.
4. Click **Save.**

Index Maintenance will be performed during the next maintenance period

Set up Partitioning for Raw Sample Data (MS SQL Server)

If you are using MS SQL Server, you can set up partitioning to improve performance when accessing the raw sample data tables.

Follow these steps:

1. Click the Database Configuration folder.
2. Select the Partition data tables box.
3. Click **Save.**

Partitioning will be performed during the next maintenance period. The time required to execute the partitioning is dependent on both the amount of data and the performance of the disk subsystem. Partitioning can take up to several days on especially large installations.

Schedule Database Maintenance

You can schedule automatic database maintenance to optimize system performance.

Follow these steps:

1. In the data_engine probe configuration menu, click the **Scheduler** folder.
2. Enter a maintenance start date. You can set the start date for a future date or can start the maintenance schedule immediately.
3. Enter a maintenance end date. The end date can either be a calendar date or a set number of occurrences.
4. Select a Recurrence pattern.

   The additional time options appear for your selected duration pattern.
5. Enter the additional time options that are required for your selected duration. For example, if you select a daily duration pattern, enter the start hour and minute.
6. Click **Apply**.

   The configuration changes are saved. If the start now option was selected, the new schedule for maintenance begins.

Chapter 3: Best Practices

The data_engine performs most tasks with little or no interaction from the administrator. However there are some configurations that can improve performance.

data_engine Probe Location

- In order to reduce the network traffic, run the data_engine on a Hub as close to the database server as possible.

Thread Count

- Multi-threading is not enabled by default in the data_engine probe. To increase data_engine performance, you can increase the number of threads by using the thread_count_insert parameter in Raw Configure. The optimum thread count is highly dependent on several factors, including:
  - The number of CPUs running on the system.
  - The number of RN tables in the NIS database.
  - The size of your NMS deployment.

Hub Bulk Size

- By default, the hub bulk size is set to 20. A low bulk size is optimal for small environments with small message rate throughput in the data_engine. However, if your NMS deployment has a high message rate you can increase the hub_bulk_size parameter in Raw Configure to values in the 1800-2000 range. This action increases the number of QoS messages that are sent at once between the hub and data_engine.

Chapter 4: Configuration Details

This section describes the configuration information and options available through the Admin Console data_engine configuration GUI. The navigation pane organizes data_engine configuration into the following nodes:

- data_engine (see page 19)
- Scheduler (see page 25)
- Quality of Service (see page 24)
- Database Configuration (see page 20)

To access the data_engine configuration interface, select the robot that the data_engine probe resides on in the Admin Console navigation pane. In the Probes list, click the arrow to the left of the probe and select Configure.

data_engine

Navigation: data_engine

This section lets you view probe and QoS information, change the log level, and set data management values.

Probe Information

This section provides the basic probe information and is read-only.

General Configuration

This section provides general configuration details.

- **Log Level:** Sets the amount of detail that is logged to the log file.
- **Data Management default values:** The default settings for data maintenance. These settings apply to all QoS settings unless they have been individually overwritten in the Quality of Service settings.

Compress data before delete: If selected, then by default, compression is done on raw data which is then copied into historic tables. This is only completed before a delete is performed. In addition, historic data is compressed and aggregated into Daily Data before deletion from the HN tables.

- **Delete raw data older than:** Raw data older than the indicated number of days is deleted.
- **Delete historic data older than:** Historic table data older than the indicated number of days is deleted.
- **Delete daily average data older than:** Daily table data older than the indicated number of days is deleted.

Quality of Service Type Status

This section provides data regarding the QoS tables and is read-only.

Note: The status information is created based on statistics that are generated by the database provider. If incorrect information is displayed, you may need to update the table statistics. For more information, refer to Update the Table Statistics (see page 30).

Database Configuration

Important! The database connection properties should only be changed in limited circumstances such as recovery operations. Changing the Database Vendor can cause connection issues. If you are changing database vendors, CA recommends reinstalling NMS.

The Database Configuration section allows you to specify the database connection settings. These settings are different for each database vendor:

- MySQL (see page 21)
- Microsoft (see page 22)
- Oracle (see page 23)

To test the connection for all vendors, click **Actions**, **Test Connection** at the top of the screen.

MySQL

Navigation: data_engine>Database configuration>MySQL

This section lets you configure the connection options for a MySQL database.

- **Schema:** The database schema name.
- **Server Host:** The database server name or IP address.
- **Port:** The port number to connect to the database server.

  Default: 3306
- **Username:** The login user name.
- **Password:** The login user password.

  Restriction: The password does NOT contain any special characters (such as "**;**").

Microsoft SQL Server

Navigation: data_engine>Database configuration>Microsoft

This section lets you configure the connection options for a Microsoft SQL Server database.

- **Provider:** The SQL server provider.
- **Initial Catalog:** The database name.
- **Data Source:** The database server.
- **User ID:** The login user.
- **Password:** The login user password.

  Restriction: Password cannot contain any special characters (such as "**;**").
- **Parameters:** Other parameters for the OLEDB connection.

Compression mode: The method that is used for data compression:

- **None:** No compression occurs.
- **Page:** Optimizes storage of multiple rows in a page, a super-set of row compression.
- **Row:** Stores fixed-length data types in variable-length storage format.

- **Index Maintenance**: Perform table reindexing with other maintenance routines, which by default are executed every 24 hours.
- **Maintenance mode:** How the indexes are maintained:
  - **Dynamic**: Maintenance is performed based on the index statistics.
  - **Reorganize**: Maintenance is performed using the "alter index ... reorganize" SQL Server script.
  - **Rebuild**: Maintenance is performed using the "alter index ... rebuild" SQL Server script.

    Important! It is not possible to rebuild the index for single partitions prior to SQLServer 2014. You can only reorganize individual partitions. Performing automatic indexing for large tables from the data_engine is discouraged, as indexing may not complete in a reasonable amount of time.
- **Online mode**: The effect of maintenance on concurrent use of the QoS tables:
  - **Dynamic**: The maintenance is determined by the edition of SQL Server. If SQL Server is the Enterprise Edition, then Online mode is used for maintenance (if the chosen maintenance mode supports it); otherwise, offline mode is used.
  - **Online**: The QoS tables are available for update and query during the table maintenance period. Online mode offers greater concurrency but demands more resources.

  - **Offline**: The QoS tables are unavailable for update and query during the table maintenance period.
- **Fragmentation level: Low threshold:** If the fragmentation for an index is less than the low threshold percent value, then no maintenance is performed.
- **Fragmentation level: High threshold:** If dynamic maintenance mode is selected and fragmentation is between the low and high threshold percentages, then the Reorganize mode is used; otherwise the Rebuild mode is used.

  Note: This option is only available for Microsoft SQL Server Enterprise Edition. It is not available when using the Microsoft SQL Express edition.
- **Index name pattern:** The indexes that are maintained.

  Default: Blank (a blank entry results in all indexes being considered for maintenance).
- **Partition data table:** Option to partition the data tables.

  Note: This option is only available for Microsoft SQL Server Enterprise Edition. It is not available in the Microsoft SQL Express edition.

Oracle

Navigation: data_engine>Database configuration>Oracle

This section lets you configure the connection options for an Oracle database.

- **Hostname:** The database server name or IP address
- **Port:** The port number to connect to the database server
- **Username:** The login user name
- **Password:** The login user password.

  Restriction: Password cannot contain any special characters (such as "**;**").
- **Service Name:** The Oracle SID or Service name

Quality of Service

Navigation: data_engine>Quality of Service

The Quality of Service section displays the attributes for the QoS metrics.

- **Name:** The QoS type name.
- **Description:** Description of the QoS type.
- **QoS Group:** The QoS group is a logical group to which the QoS belongs (optional).
- **Unit:** The unit of the QoS data (The abbreviated form of the QoS data unit).
- **Has Max Value:** The data type has an absolute max.

  Example: disk size, memory usage.
- **Is Boolean:** The data type is logical (yes/no). Example: host is available/unavailable, printer is up/down.
- **Type:** Different data types:
  - **0** = Automatic (The sample value is read at fixed intervals, individually set for each of the probes).
  - **1** = Asynchronous (The sample value is read only each time the value changes, and the new value is read).
- **Override Raw Age:** Check this box to override the raw age of the QoS metric.
- **Raw Age**: The number of days you would like to retain the QoS metric information.
- **Override History Age:** Check this box to override the history age for the QoS metric.
- **History Age**: The number of days you would like to retain the history information.
- **Override Daily Average Age**: Check this box to override the daily average age for the QoS metric.
- **Daily Average Age:** The number of days you would like to retain the daily average information.
- **Override Compression**: Check this box to override compression settings for data in RN and HN tables.
- **Compress:** Raw data is compressed and aggregated into Hourly/Historic data before deletion from the RN tables. This Historic/Hourly data is then compressed and aggregated into Daily data before deletion from the HN tables.

Scheduler

Navigation: data_engine>Scheduler

This section allows you to schedule database maintenance.

- **Start time** - Select either **Now** or a specific date and time. Selecting now begins the new database maintenance schedule immediately.
- **Ending** - Select **Never**, **After** x occurrences, or **By** a specific date and time.
- **Recurring** - select one of the following occurrence patterns:
  - **Minutely**
  - **Hourly**
  - **Daily** (including a specific time)
  - **Weekly** (including a specific time and days of the week)
  - **Monthly** (including occurrence, calendar day, and specific time)
  - **Yearly** (including month and specific time)

Chapter 5: Troubleshooting

Troubleshooting topics:

- Viewing the Log File (see page 27)
- Corrupted QoS Definition Values (see page 28)
- Out of date information in the Quality of Service Type Status Table (see page 30)
- Check Partitioning Jobs (see page 34)

If your problem is not addressed here:

- Look for a solution or ask other users for help on the CA Nimsoft Community Forum.
- Contact Nimsoft Technical Support.

Send us feedback with the "rate this page" link below. We will strive to include a solution in the next release of this document.

Viewing the Log File

Advanced users may find it helpful to view the log file. To view the log file, click the data_engine probe and select **View Log**. You also can modify the log file settings so that it retains more data for troubleshooting.

Corrupted QoS Definition Values

Symptom:

I see corrupted QoS Definition values in QoS data within reports or dashboards, for example user-defined units such as degrees F or C, watts, etc. being labeled “variant". Customized QoS definition units have been incorrectly replaced (overridden) by data_engine version 7.85 (NM Server 5.60) or data_engine 7.86 (NM Server 5.61).

Solution:

Versions 7.87 through 7.90 of data_engine include a patch utility to recover and restore customized QoS definition units incorrectly replaced (overridden) by data_engine version 7.85 (NM Server 5.60) or data_engine 7.86 (NM Server 5.61). This patch utility uses a conditional override approach that corrects the issue.

Follow these steps:

1. From a command prompt, navigate to the directory that contains the utility:

   (Unix) cd /opt/nimsoft/probes/slm/data_engine/tools

   (Windows) CD C:\Program Files (x86)\Nimsoft\probes\slm\data_engine\tools

2. Execute the tool in “report” mode (-r flag set) with the java version installed with NM Server:

   (Unix) ../../../../jre/jre1.6.0_24/bin/java –jar qos_def_unit_repair_kit.jar –r

   (Windows)..\..\..\..\jre\jre1.6.0_24\bin\java –jar qos_def_unit_repair_kit.jar -r

3. The patch utility scans the S_QOS_DEF_SKIP_UNIT table in the database and finds QoS Definitions that are suspected to be corrupt.

   Important: The S_QOS_DEF_SKIP_UNIT table holds QoS definition values that should not override what is sent by a QoS probe. This table is pre-populated with three values:

   - variant
   - none
   - user defined

   If you have defined additional custom values, which have been incorrectly overridden by a previous data_engine version, add these values as new rows to the S_QOS_DEF_SKIP_UNIT table prior to step 5 below, so that the patch utility will find, report, and fix them as well. (Use standard database management tools to connect to the NIS database and add the rows and new values to the S_QOS_DEF_SKIP_UNIT table.)

The report generated by the utility shows corrupt QoS Definitions (if any) and the probe or probes associated with that data. Here is an excerpt from an example report:

```
Listing current problems…
S_QOS_DEFINITION { name=QOS_CPUSAMPLECOUNT, qosDefId=12,
group=QOS_VMWARE, unit=none,
unitShort=sc }
S_QOS_DATA { source=esxiqa1.i9.x, target=CPU sample count,
origin=w2k8-vm0hub,
host=10.0.0.1, robot=w2k8-vm0, probe=vmware }
S_QOS_DEFINITION { name=QOS_SNMP_VARIABLE, qosDefId=11,
group=QOS_SNMP_VARIABLE, unit=variant,
unitShort=value }
S_QOS_DATA { source=w2k8-vm0, target=interfaces.ifTable.ifREntry.ifOutOctets.1,
origin=w2k8-vm0hub, host=10.0.0.1, robot=w2k8-vm0, probe=snmpget }

S_QOS_DATA { source=w2k8-vm0, target=system.sysServices.0,
origin=w2k8-vm0hub,
host=10.0.0.1, robot=w2k8-vm0, probe=snmpget }
End of current problems list.
Running in report-only mode
```

4. Run the utility without the –r flag set to make the repairs:

   (Unix) ../../../../jre/jre1.6.0_24/bin/java –jar qos_def_unit_repair_kit.jar

   (Windows)..\..\..\..\jre\jre1.6.0_24\bin\java –jar qos_def_unit_repair_kit.jar

   The utility subscribes to the NimBUS to receive QoS Definitions and corrects any in the database that have been incorrectly replaced.

5. Restart each of the probes listed in the report. In the example above the vmware and snmpget probes would be restarted so that the correct units are received. Probes are identified at the end of each S_QOS_DATA listing in the report. For example:

   ```
   S_QOS_DATA { source=esxiqa1.i9.x, target=CPU sample count,
   origin=w2k8-vm0hub,
   host=10.0.0.1, robot=w2k8-vm0, probe=vmware }
   ```

Once the patch utility has repaired all QoS Definitions it does not need to be run again and no other action is required.

Out of date information in the Quality of Service Type Status Table

Symptom:

I see out of date information in the Quality of Service Type Status Table.

Solution:

Table statistics are collected by database software to create the best execution plan for a query. Some examples of collected statistics include:

- Rows that are stored in a table
- Available indexes
- How many pages store the rows

The data_engine probe uses these table statistics to generate the Quality of Service Type Status table. Table statistics can be manually updated if they become out of date.

The procedure for updating the table statistics is different for each database vendor:

- Table Statistics for Oracle (see page 30)
- Table Statistics for MySQL (see page 31)
- Table Statistics for Microsoft SQL Server (see page 31)

Important! Updating table statistics significantly impacts performance on all database platforms, especially on larger databases.

Table Statistics for Oracle

To receive correct statistics in Oracle, run one of the following queries on all RN_ and HN_ tables:

- ANALYZE table RN_QOS_DATA_XXXX ESTIMATE STATISTICS; - Samples from the table are taken and stored in the data dictionary.
- ANALYZE table RN_QOS_DATA_XXXX COMPUTE STATISTICS; - The entire table is analyzed using a full table scan and stored in the data dictionary.

Using the ANALYZE command in Oracle can be a time-consuming operation, especially for large databases. Only perform the command sporadically and do not use Automated Maintenance Tasks.

For more information, refer to the following Oracle documentation:

- http://docs.oracle.com/cd/B28359_01/server.111/b28310/general002.htm#ADMIN11524

Table Statistics for MySQL

To receive correct statistics in MySQL, run the following query on all RN_ and HN_ tables:

- ANALYZE table RN_QOS_DATA_XXXX;

In MySQL, the ANALYZE command holds a read lock on tables, which can negatively impact database performance.

For more information, refer to the following MySQL documentation:

- http://dev.mysql.com/doc/refman/5.5/en/tables-table.html
- http://dev.mysql.com/doc/refman/5.6/en/analyze-table.html

Table Statistics for Microsoft SQL Server

Normally table statistics are automatically managed by the SQL Server. However, this functionality does not work in some cases.

For example, if you are performing bulk inserts using the OLE DB FastLoad API, the statistics for data tables are not automatically updated. This can lead to poor performance and extra work for SQL Server.

The data_engine probe contains options that can automatically update statistics for Microsoft SQL Server. The code is in a stored procedure in the SLM database that is named 'spn_de_UpdateStatistics'.

Updating tables statistics in Microsoft SQL server causes queries to recompile. For more information, refer to the following Microsoft SQL documentation:

- http://msdn.microsoft.com/en-us/library/ms187348.aspx
- http://msdn.microsoft.com/en-us/library/ms187348(v=sql.100).aspx

The behavior can be controlled with the following variables in the raw configure menu:

Key name	Default value	Type
statistics_age	24	Time in hours. This means when the stored procedure is called, statistics that are older than this number will be updated. This value is used by the stored procedure, not data_engine itself. If this number is set to 0 (zero), statistics will be disabled and not be run at all by the data_engine.

statistics_pattern	RN_QOS_DATA%	String pattern to which data tables will be updated.
statistics_loglevel	0	Numbers 0 to 5 used by the stored procedure when logging to tbnLogging.
statistics_time_pattern	<not set>	The scheduling string that determines when to run statistics. If this key is empty or not set, the same schedule that is defined for data management will be used. This means the statistics will be run when data_engine has finished index maintenance and data management. If this value is specified to a different schedule, the statistics will be updated independently of when data management is scheduled. The string will be used by the Nimsoft calendar scheduling library, which is used by various Nimsoft components. It supports RFC2445. See short example below.

Some string examples that are copied from the library help file.

```
/*****************************************************************************
** nimCalCreate - Creates a handle to a nimCal structure
*****************************************************************************
** PARAMETERS:
**      char    *pattern       - RFC2445 ,'weekly' or 'dts'
**  char       *start      - startdate: yyyy-mm-dd [hh:mm:ss] || NULL
**                                                    : weekly format 1 or 2
**
**      start = 'yyyy-mm-dd [hh:mm:ss]' will expect the 'pattern' to comply with RFC2445.
**        = NULL results in setting start to 'now'
**      e.g.
**              h = nimCalCreate("DTSTART:19970610T090000|RRULE:FREQ=YEARLY;COUNT=10",NULL);
**              h =
nimCalCreate("DTSTART:19970610T090000|RRULE:FREQ=YEARLY;COUNT=10","2007-07-25");
**
**  pattern = 'weekly' handles two 'start' formats:
**              1. 0123,10:00,14:00 [,NOT]                                      (old NAS
format)
**              2. MO=12:00-14:00,15:30-17:00;TU=08:00-16:00 (new, allow 8-16)
**
**              h = nimCalCreate("weekly","012,10:00,14:00");
**              h = nimCalCreate("weekly","MO=12:00-14:00,15:30-17:00;TU=08:00-16:00");
**              h = nimCalCreate("dts","2007-08-20 08:00,2007-08-27 08:00,2007-09-03
08:00,2007-09-10"
**
** Note: Free the handle using nimCalFree.
*****************************************************************************/
```

You can also create a schedule in NAS and use the resulting string from there or use data_engine scheduler to create a string.

Check Partitioning Jobs

Use the following SQL statement to determine the partitioning jobs that are running:

```
SELECT sqltext.TEXT,
req.session_id,
req.status,
req.command,
req.cpu_time,
req.total_elapsed_time
FROM sys.dm_exec_requests req
CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS sqltext;
```

If the results display multiple "CREATE nonclustered index" statements, you have more than one partitioning job running.

To stop a partitioning job:

```
KILL [session_id]
```