UMP (USM) Slow Performance Guide and Troubleshooting Checklist 1.1

Updated: June 29, 2016
Author: Steve Danseglio

Summary:

This Knowledge Base Article provides a guide and troubleshooting checklist for analyzing some of the most common UMP/USM performance issues such as slow response time, slow USM loading, slow alarm view loading, pages loading indefinitely, slow data retrieval, and other slow performance response and delay-related issues.

Note that a 'generally acceptable' response time for UMP/USM is 0-10 seconds. This range is based on Support and customer feedback. If loading USM consistently takes more than 10 seconds then please run through the following checklist items.

Background:

'Slow' UMP/USM performance can be caused by a variety of factors which can differ from one environment to another. This guide provides items to check for narrowing down the potential cause(s) contributing to slow performance. It also provides suggestions to help alleviate issues which may be contributing to slow performance.

Environment:

- CA UIM 8.2 or higher
- All relevant UIM/UMP hotfixes should be applied. Hotfixes are downloadable from:

  http://www.ca.com/us/support/ca-support-online/product-content/recommended-reading/technical-document-index/ca-unified-infrastructure-management-hotfix-index.aspx
- Database: Microsoft SQL Server 2012 or higher
- Database Server machine specifications
  For CA UIM Database Server requirements based on deployment size, please refer to this link:

  https://docops.ca.com/display/UIM84/Prepare+Your+Server+Hardware#PrepareYourServerHardware-DatabaseServerRequirements

WARNING:

Your DBA should be consulted before running any of the queries listed below or referenced by this Article. Moreover, to be safe, queries should be tested within your CA UIM Test/Dev environment before use in Production.

Instructions:

☐ **Determine what version of Microsoft SQL Server database you are running.**

To do so, execute the query

```
Select @@VERSION
```

Then expand the column to view and copy the entire resultant string showing the version:

```
Microsoft SQL Server 2012 - 11.0.2100.60 (X64)
Feb 10 2012 19:39:15
Copyright (c) Microsoft Corporation
Enterprise Edition (64-bit) on Windows NT 6.0 <X64> (Build 6002: Service Pack 2) (Hypervisor)
```

IMPORTANT:
For CA UIM we highly recommend using Microsoft SQL Server 2012 or higher, and Enterprise Edition, NOT Standard. The Enterprise Edition takes advantage of all of the benefits of *true partitioning* where SQL Standard edition does not such as:

a) Online partitioning
b) Partition Switching
c) Table and Index partitioning
d) Partition Table Parallelism
e) Allows you to take advantage of the related data_engine configuration settings

We do not recommend using Microsoft SQL Server *Standard* Edition in medium to large installs given the limited feature set.

☐ **Determine the total size of your CA UIM database. A query to check current database size and space available for the CA UIM/Nimsoft SLM database is shown below. Replace the name of the database with the name of your own CA UIM database and then run it.**

```
use CA_UIM
go
sp_spaceused
go
```

Is your CA UIM database size greater than 1 TB? If so, please note that running one or more of the queries listed in this document may take a longer time than expected. If your database is not being administered by a DBA, and fragmentation is not being checked and defragmented on a regular daily/weekly basis, or partitioning has not been enabled, this may be also contributing to 'slow' performance. Note also that CA UIM databases that are > 1 TB require regular periodic administration and tuning.

☐ **Determine the level of database fragmentation (for key USM tables fragmentation should be less than 30%). This is one of the first steps to take in the process and it is a key step.**

```
S_QOS_DATA
CM_COMPUTER_SYSTEM
CM_CONFIGURATION_ITEM
```

CM_CONFIGURATION_ITEM_METRIC
CM_DEVICE
CM_CONFIGURATION_ITEM_DEFINITION
CM_CONFIGURATION_ITEM_METRIC_DEFINITION
NAS_ALARMS
NAS_TRANSACTION_SUMMARY
NAS_TRANSACTION_LOG

Listed below is a query to check the level of fragmentation for all tables or some of the key USM tables using a modified WHERE clause:

```
SELECT dbschemas.[name] as 'Schema',
dbtables.[name] as 'Table',
dbindexes.[name] as 'Index',
indexstats.avg_fragmentation_in_percent,
indexstats.page_count
FROM sys.dm_db_index_physical_stats (DB_ID(), NULL, NULL, NULL, NULL) AS
indexstats
INNER JOIN sys.tables dbtables on dbtables.[object_id] = indexstats.[object_id]
INNER JOIN sys.schemas dbschemas on dbtables.[schema_id] = dbschemas.[schema_id]
INNER JOIN sys.indexes AS dbindexes ON dbindexes.[object_id] =
indexstats.[object_id]
AND indexstats.index_id = dbindexes.index_id
WHERE indexstats.database_id = DB_ID()
ORDER BY indexstats.avg_fragmentation_in_percent desc
```

```
--This query can be modified to focus on specific tables by appending the table name to the 'where' clause:

-- WHERE indexstats.database_id = DB_ID() AND dbtables.[name] like '%<table_name_string>%'
```

Please also refer to the following Knowledge Article *"Defragmentation of NimsoftSLM database"* for more information at this link:

http://ecm.ca.com/KB/Pages/authoring/create-KB-Article.aspx?kbid=TEC000004098

Note that executing a fragmentation check for all or some large tables may take a long time depending on the number of rows. You may choose to run this query after hours or on the weekend if/when the database is not as busy.

It is also important to note the page count of the query as well. A table that has very few rows is more efficiently read in as a heap than incurring the cost of an index seek. Therefore, you'll always see 99% fragmentation on indexes that are built against small tables.

☐ **Determine how long the data is actually being retained in the tables**
Determine how 'old' the raw data is in the RN tables and make a decision whether or not you need to execute a *purge* of old unwanted data manually (versus using the data_engine) so you can bring your tables up to date. You may have to manually purge the data because there may be one or more reasons why the data_engine retention settings are not taking effect, e.g., the nightly data maintenance job is not completing.

See SQL query statement example below, 'purging of data by date'

```
--Purge of data by date.
--See date (as per sampletime below which you need to adjust before running this)
```

Script to check what you'll be deleting:

```
--------------------

declare table_id_cursor cursor for select table_id from s_qos_data
declare @sql varchar(1000)
declare @r_table varchar(64)
declare @h_table varchar(64)
declare @table_id int

OPEN table_id_cursor
FETCH NEXT FROM table_id_cursor into @table_id
WHILE @@FETCH_STATUS = 0
BEGIN
set @r_table = (select r_table from S_QOS_DATA where table_id = @table_id)
set @h_table = (select h_table from S_QOS_DATA where table_id = @table_id)
set @sql = 'select COUNT(*) from ' + @r_table + ' where table_id = ' +
CAST(@table_id AS varchar(10)) + ' and sampletime < ''2014-01-01
00:00:00'''
EXECUTE(@sql)
set @sql = 'select COUNT(*) from ' + @h_table + ' where table_id = ' +
CAST(@table_id AS varchar(10)) + ' and sampletime < ''2014-01-01
00:00:00'''
EXECUTE(@sql)
FETCH NEXT FROM table_id_cursor into @table_id
END
CLOSE table_id_cursor
DEALLOCATE table_id_cursor

--------------------
```

Script to actually delete the data (WARNING-> BACK UP YOUR DATABASE FIRST!!!)

```
declare table_id_cursor cursor for select table_id from s_qos_data
declare @sql varchar(1000)
declare @r_table varchar(64)
declare @h_table varchar(64)
declare @table_id int

OPEN table_id_cursor
FETCH NEXT FROM table_id_cursor into @table_id
WHILE @@FETCH_STATUS = 0
BEGIN
set @r_table = (select r_table from S_QOS_DATA where table_id = @table_id)
set @h_table = (select h_table from S_QOS_DATA where table_id = @table_id)
set @sql = 'delete from ' + @r_table + ' where table_id = ' +
CAST(@table_id AS varchar(10)) + ' and sampletime < ''2014-01-01
00:00:00'''
EXECUTE(@sql)
set @sql = 'delete from ' + @h_table + ' where table_id = ' +
CAST(@table_id AS varchar(10)) + ' and sampletime < ''2014-01-01
00:00:00'''
EXECUTE(@sql)
FETCH NEXT FROM table_id_cursor into @table_id
END
CLOSE table_id_cursor
```

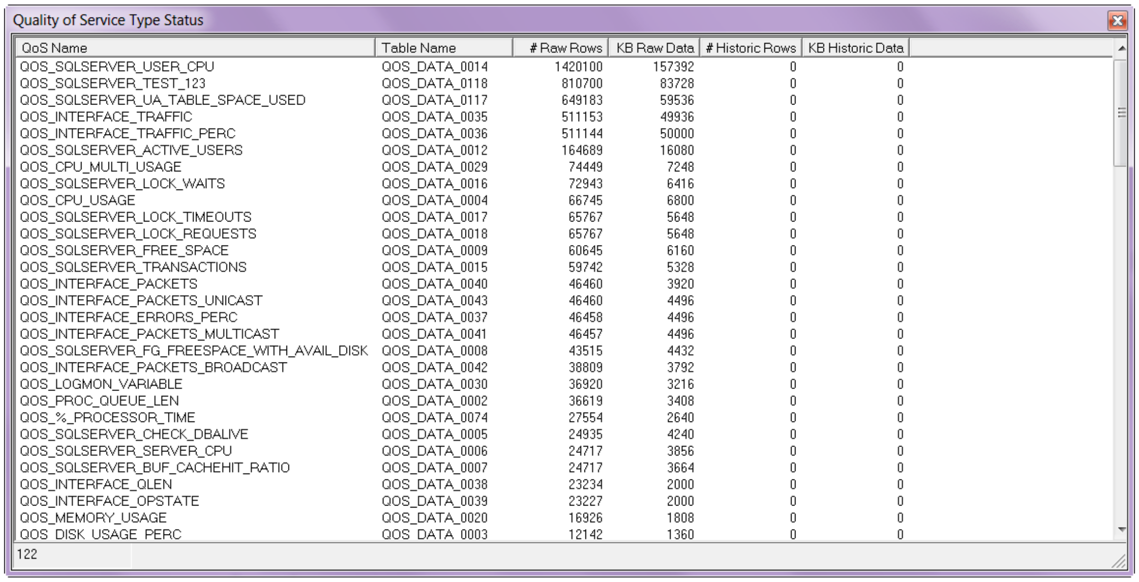
```
DEALLOCATE table_id_cursor
--------------------
```

Depending on how much data is in the database you may have to use a slightly different approach/query than the one described above. See the section "**How to purge data without running out of memory or resources"** towards the end of this document.

- Note that when you reduce data retention using the data_engine settings, "Delete raw data older than 'x' days" it must be done on smaller time periods of data at a time in a stepped fashion of 15-20 days at a time until you reach your goal, otherwise the data maintenance will most likely not have a chance to finish in time during the nightly maintenance period.

  That being said, once again this should be tested in a Test/DEV environment first and reviewed and approved by your CA UIM DBA BEFORE using it in Production. Your DBA may have their own preferred method to purge old data from the database.

☐ **Determine if there are very large tables from specific probes**
For example, probes such as exchange_monitor, ntperf, interface_traffic, vmware, etc., could be collecting tons of data, and the QOS_* tables may contain millions of rows. To quickly see the number of rows for the QOS being collected in the database you can click on the Status button in the data_engine:



Quality of Service Type Status

QoS Name	Table Name	# Raw Rows	KB Raw Data	# Historic Rows	KB Historic Data
QOS_SQLSERVER_USER_CPU	QOS_DATA_0014	1420100	157392	0	0
QOS_SQLSERVER_TEST_123	QOS_DATA_0118	810700	83728	0	0
QOS_SQLSERVER_UA_TABLE_SPACE_USED	QOS_DATA_0117	649183	59536	0	0
QOS_INTERFACE_TRAFFIC	QOS_DATA_0035	511153	49936	0	0
QOS_INTERFACE_TRAFFIC_PERC	QOS_DATA_0036	511144	50000	0	0
QOS_SQLSERVER_ACTIVE_USERS	QOS_DATA_0012	164689	16080	0	0
QOS_CPU_MULTI_USAGE	QOS_DATA_0029	74449	7248	0	0
QOS_SQLSERVER_LOCK_WAITS	QOS_DATA_0016	72943	6416	0	0
QOS_CPU_USAGE	QOS_DATA_0004	66745	6800	0	0
QOS_SQLSERVER_LOCK_TIMEOUTS	QOS_DATA_0017	65767	5648	0	0
QOS_SQLSERVER_LOCK_REQUESTS	QOS_DATA_0018	65767	5648	0	0
QOS_SQLSERVER_FREE_SPACE	QOS_DATA_0009	60645	6160	0	0
QOS_SQLSERVER_TRANSACTIONS	QOS_DATA_0015	59742	5328	0	0
QOS_INTERFACE_PACKETS	QOS_DATA_0040	46460	3920	0	0
QOS_INTERFACE_PACKETS_UNICAST	QOS_DATA_0043	46460	4496	0	0
QOS_INTERFACE_ERRORS_PERC	QOS_DATA_0037	46458	4496	0	0
QOS_INTERFACE_PACKETS_MULTICAST	QOS_DATA_0041	46457	4496	0	0
QOS_SQLSERVER_FG_FREESPACE_WITH_AVAIL_DISK	QOS_DATA_0008	43515	4432	0	0
QOS_INTERFACE_PACKETS_BROADCAST	QOS_DATA_0042	38809	3792	0	0
QOS_LOGMON_VARIABLE	QOS_DATA_0030	36920	3216	0	0
QOS_PROC_QUEUE_LEN	QOS_DATA_0002	36619	3408	0	0
QOS_%_PROCESSOR_TIME	QOS_DATA_0074	27554	2640	0	0
QOS_SQLSERVER_CHECK_DBALIVE	QOS_DATA_0005	24935	4240	0	0
QOS_SQLSERVER_SERVER_CPU	QOS_DATA_0006	24717	3856	0	0
QOS_SQLSERVER_BUF_CACHEHIT_RATIO	QOS_DATA_0007	24717	3664	0	0
QOS_INTERFACE_QLEN	QOS_DATA_0038	23234	2000	0	0
QOS_INTERFACE_OPSTATE	QOS_DATA_0039	23227	2000	0	0
QOS_MEMORY_USAGE	QOS_DATA_0020	16926	1808	0	0
QOS_DISK_USAGE_PERC	QOS_DATA_0003	12142	1360	0	0

122

Run this query in MS SQL Server studio to display row count and data size:

```
USE [CA_UIM]
GO
CREATE TABLE #temp (
table_name sysname ,
row_count INT,
reserved_size VARCHAR(50),
    data_size VARCHAR(50),
    index_size VARCHAR(50),
    unused_size VARCHAR(50))
    SET NOCOUNT ON
    INSERT #temp
    EXEC sp_msforeachtable 'sp_spaceused ''?'''
    SELECT a.table_name,
    a.row_count,
    COUNT(*) AS col_count,
    a.data_size
    FROM #temp a
    INNER JOIN information_schema.columns b
    ON a.table_name collate database_default
    = b.table_name collate database_default
    GROUP BY a.table_name, a.row_count, a.data_size
    ORDER BY CAST(REPLACE(a.data_size, ' KB', '') AS integer) DESC
    DROP TABLE #temp
```

Once you have this information, this will help guide you how to proceed. For example if you find the largest tables are RN_QOS_DATA_* tables then you should look to find out which probe's QOS is associated with those tables and find out if an excessively large amount of monitoring was enabled.

In some cases it is best to disable any unnecessary monitoring and collection of QOS where the tables are building too fast and the rows added to the tables per hour or day are very high, e.g., greater than 2000. Note that some probes enable a ton of monitoring 'out of the box,' e.g., exchange_monitor performance (perfmon) monitors, and hence those tables can build very quickly. You may want to prune the data/truncate the tables if your response times are slow due to these tables. Note that as of the writing of this document, there is currently a new version of the perfmon probe. The old perfmon probe was erroneously creating CI entries in the CI_CONFIGURATION_ITEM table for each dynamic monitoring profile that was created in response to the monitoring requested from the Exchange Monitoring probe. No QOS data is ever generated for these CI's so they should not be created as CI's. The new perfmon build no longer creates the CI records hence the table is not constantly updated with a ton of new entries each hour. If you need further information about this please contact CA support for more information.

Microsoft SQL Server profiler and other tools may help you determine which jobs are taking the longest/most resources. SQL Server profiles can e started from the Tools Menu within SQL Server Management studio.

https://msdn.microsoft.com/en-us/library/ms173799%28v=sql.110%29.aspx

Note also that enabling UMP/USM debug mode will reveal which tables are being accessed during performance issues.

You should also check to see if the samples are being deleted properly from the RN Tables, by looking at the oldest 'sampletime' value in the raw data tables and make sure it is not older than the amount of data that should be retained (as per the data_engine settings).

You can check the data_engine retention settings ("Delete raw data older than" GUI option, or delete_raw_samples value in Raw Configure mode) and compare it to the results of a query such as:

```
SELECT * from RN_QOS_DATA_0122 ORDER BY sampletime ASC
```

The data should not be older than the retention settings allow.

Also make sure the database is set to use a "SIMPLE" recovery model and not "FULL," but more importantly the best practices listed here in the Article: *"CA UIM (Nimsoft) Database Best Practices for MS SQL Server"* should be reviewed and followed:

http://ecm.ca.com/KB/Pages/authoring/create-KB-Article.aspx?kbid=TEC000003224

☐ **Disk I/O on the MS SQL Server**

Determine and Analyze I/O Wait
I/O Wait is the percentage of time your processors are waiting on the disk. On UNIX/Linux you can check your I/O wait percentage via `top`. If your I/O wait percentage is greater than (1/# of CPU cores) then your CPUs are waiting a significant amount of time for the disk subsystem to catch up. On Windows you could use the Resource Monitor Disk stats or specific queries such as below.

DMV - sys.dm_io_virtual_file_stats

This DMV will give you cumulative file stats for each database and each database file including both the data and log files. Based on this data you can determine which file is the busiest from a read and/or write perspective.

The output also includes I/O stall information for reads, writes and total. The I/O stall is the total time, in milliseconds, that users waited for I/O to be completed on the file. By looking at the I/O stall information you can see how much time was waiting for I/O to complete and therefore the users were waiting.

Note that the data that is returned from this DMV is cumulative data, which means that each time you restart MS SQL Server, the counters are reset. Since the data is cumulative you can run this once and then run the query again in the future and compare the deltas for the two time periods. If the I/O stalls are high compared to the length of the time period then you may have an I/O bottleneck. Here is a sample query you can use. Note you can use the ORDER BY and sort the results differently, e.g., ORDER BY io_stall DESC, and so on.

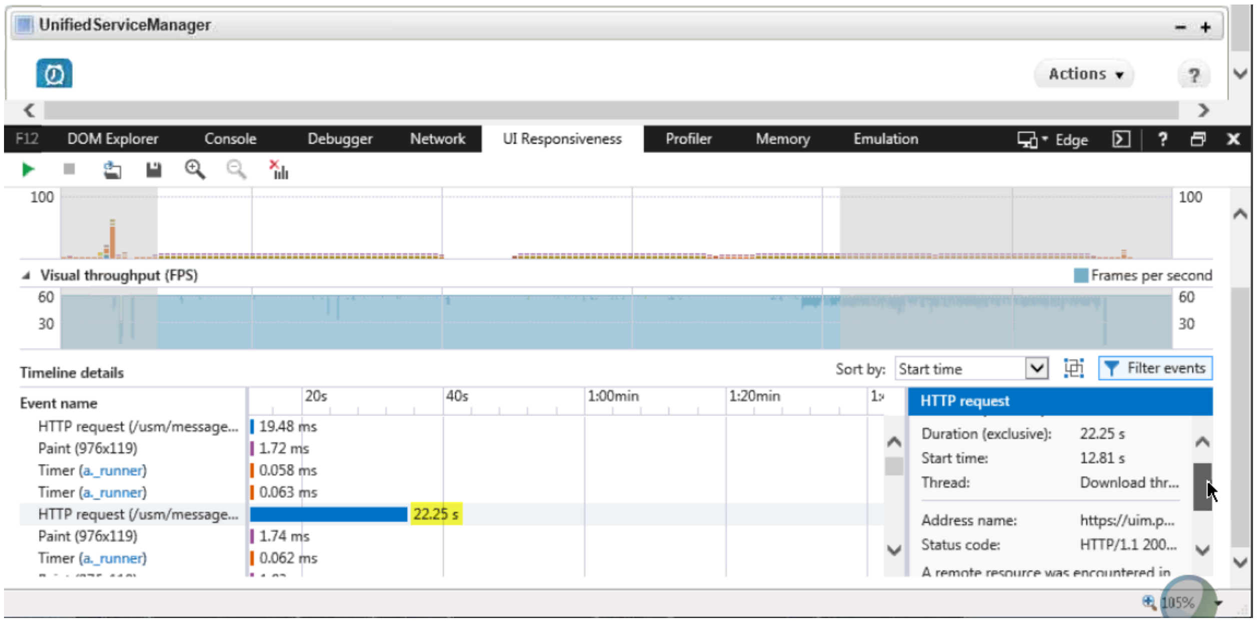
```
SELECT
cast(DB_Name(a.database_id) as varchar) as Database_name,
b.physical_name, *
```

```
FROM
sys.dm_io_virtual_file_stats(null, null) a
INNER JOIN sys.master_files b ON a.database_id = b.database_id and a.file_id =
b.file_id
ORDER BY Database_Name
```

☐ **Network speed/available bandwidth**
Check connection speed (upload/download) from the specific clients/browser and also on/from the local UMP machine itself, e.g., http://www.speedtest.net/

☐ **Determine the size of the nas transactionlog.db file on the primary hub.**
It should not be larger than ~300 MB

☐ **Compare local versus remote connections and performance**
Check performance/response times using the specific browser developer tools or Fiddler2 and identify the longest timings – see example screen shot below.



Timeline details

Event name	
HTTP request (/usm/message...	19.48 ms
Paint (976x119)	1.72 ms
Timer (a._runner)	0.058 ms
Timer (a._runner)	0.063 ms
HTTP request (/usm/message...	22.25 s
Paint (976x119)	1.74 ms
Timer (a._runner)	0.062 ms

HTTP request

Duration (exclusive):	22.25 s
Start time:	12.81 s
Thread:	Download thr...
Address name:	https://uim.p...
Status code:	HTTP/1.1 200...

A remote resource was encountered in

☐ **Perform network tests**
You can use ping/tracrt/traceroute or other tools such as iPerf and determine if there is any network latency to/from the UMP machine(s) and database being accessed. Ask your network team to check using their own tool if necessary.

☐ **Verify if there is a proxy in front of UMP and verify its configuration**.

☐ **Multi-UMP environment**
In the case of multi-UMP instances, verify if the UMP instances have been installed and configured correctly as per the Help documentation and if there is a Load Balancer in front of UMP, verify its configuration as well. You may want to test each UMP instance by accessing each node directly to see if they respond better than going through the VIP.

☐ **UMP machine specifications**

For CA UIM Database Server requirements based on deployment size, please refer to this link:

https://docops.ca.com/display/UIM84/Prepare+Your+Server+Hardware#PrepareYourServerHardware-UMPServerRequirements

☐ **Key wasp settings**

wasp memory

Increase the maximum Java memory allocated to the wasp probe**.** Minimum recommendation for med-large environment:

```
java_mem_max = -Xmx8192m
java_mem_init = -Xms2048m
```

nimpool_timeout

If you are getting errors like connection I/O error, socket error, connection timeout then try increasing nimpool timeout in the wasp configuration to 300 seconds.

☐ **Database partitioning and status**

Is the data_engine database partitioning option enabled and has partitioning been completed?

How to determine if tables were partitioned/partitioning was completed:

This query will show you the partitions for a given table (just change the number in the RN table):

```
declare @lTableName varchar(max) = 'RN_QOS_DATA_0001';

select
'Table Name' = o.name,
'Partition Number' = p.partition_number,
Rows = p.Rows,
'Partition Range' = case when prv.value is null then '< ' +
convert(varchar(10),(select convert(datetime, min(prv.value)) from sys.objects
inner join sys.indexes i on o.object_id = i.object_id and i.index_id = 1 inner
join sys.partitions p on i.index_id = p.index_id inner join sys.partition_schemes
s on i.data_space_id = s.data_space_id left outer join sys.partition_range_values
prv on s.function_id = prv.function_id where prv.boundary_id = 1), 120)
else '>= ' + convert(varchar(10),prv.value,120) + ' and < '+ convert(varchar(10),
dateadd(DD, 1, convert(datetime, prv.value)), 120) end,
'Days From Today' = isnull(case convert(nvarchar(5), datediff(day, sysdatetime(),
convert(datetime, prv.value))) when '0' then '0 *** TODAY ***' else
convert(nvarchar(5), datediff(day, sysdatetime(), convert(datetime, prv.value)))
end, nchar(8734))
from
sys.objects o
inner join sys.indexes i on o.object_id = i.object_id and i.index_id = 1
inner join sys.partitions p on p.object_id = i.object_id and p.index_id=i.index_id
inner join sys.partition_schemes ps on i.data_space_id=ps.data_space_id
```

```
left outer join sys.partition_range_values prv on ps.function_id=prv.function_id
and prv.boundary_id+1=p.partition_number
where o.name like @lTableName
group by o.name, o.object_id, p.partition_number, p.rows, prv.value
order by o.name asc;
```

***Tables are generally partitioned in sequential order, so if you run this in the last table, you will know if it completed. That is how you can tell for sure.

The other place to look is the tbnlogging table. It should have an entry for spn_de_DataMaint indicating that the data_engine maintenance has completed. If that table is empty:

Set the table_maintenance_loglevel in the data_engine to log to the tbnLogging table. We can then see what specifically the data_engine does on table maintenance.

1. Deactivate the data_engine probe
2. Using Raw Configure mode (SHIFT + right click->Raw Configure), set the loglevel to '6' and set the table_maintenance_loglevel to '6'
3. set the "logsize" key to 50000 so you can also watch/search the log.

To check the table results/activity, you can execute this query but note that table maintenance may take some time to finish and you'll see rows being added in the query output as the maintenance run continues when you run:

```
select * FROM tbnLogging where Event like '%spn_de_DataMaint%' order by id DESC
```

Below is an example from the tbnlogging table during data_engine partitioning activity:

```
> spn_de_PartitionAdmin__CreateIndexes(DN_QOS_DATA_0007 DN_QOS_DATA_0007 DN
psnDN_QOS_DATA_0007)
- create partition scheme psnDN_QOS_DATA_0007 as partition pfnDN_QOS_DATA_0007
all to ([primary])
- create partition function pfnDN_QOS_DATA_0007(datetime) as range right for
values
('20160306','20160307','20160308','20160309','20160310','20160311','20160312','
20160313','20160314','20160315','20160316','20160317','20160318','20160319','20
160320','20160321','20160322','20160323','20160324','20160325','20160326','2016
0327','20160328','20160329','20160330','20160331','20160401','20160402','201604
03','20160404','20160405','20160406','20160407','20160408','20160409','20160410
','20160411','20160412','20160413','20160414','20160415','20160416','20160417',
'20160418','20160419','20160420')
- Final RetentionTime = 30 Extra partitions = 15
- tbn_de_DataMaintConfig RetentionTime = 30 Extra partitions = 15
> spn_de_PartitionAdmin__PartitionTable(DN_QOS_DATA_0007 DN)
> spn_de_DataMaint_DeleteOldData(DN, 7, Apr  5 2016 12:00AM)
< spn_de_DataMaint_DeleteOldData(HN, 7, Apr  5 2016 12:00AM) ---> OK
< spn_de_PartitionAdmin__ShiftTable(HN_QOS_DATA_0007,HN,2016-04-05 00:00:00)
deleted null rows ---> OK
```

What you will see in the tbnlogging table is an entry like this: “spn_de_DataMaint <” and “spn_de_DataMaint >”. The “<” and “> indicate the start and end of maintenance. It may have an entry for each qos id, and the ids will be sorted in ascending order, so when the last one completes, it is done.

Note that there is no 'final' log message or entry.

It will do the partitioning on the tables one by one in ascending order. When the last table is done, it is completed.

☐ **data_engine Indexing**

- Ensure that the data_engine option for index maintenance, is enabled for you database. This allows more efficient database maintenance. For more information please refer to:

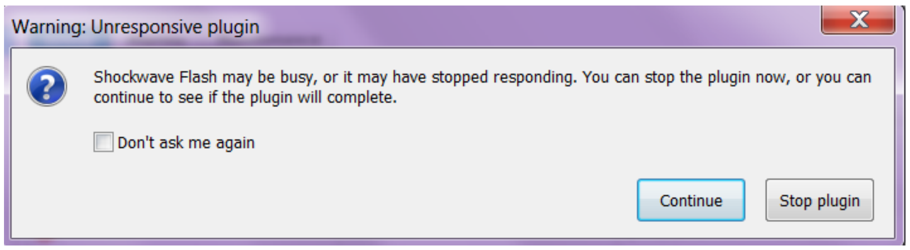
  https://docops.ca.com/ca-unified-infrastructure-management-probes/en/alphabetical-probe-articles/data_engine/data_engine-versions-8-0-8-3/v8-0-data_engine-im-configuration

☐ **Maintenance Tables**

- Check the row count for maintenance_* tables and see if it has > 500 records. This may contribute to performance problems. These tables may need to be truncated.

```
select count(*) from MAINTENANCE_SCHEDULE
select count(*) from MAINTENANCE_SCHEDULE_MEMBERS
select count(*) from MAINTENANCE_WINDOW
```

The effects of too many entries in these tables may case USM slow performance or you may see the flash plugin crashes/hangs the browser. This could apply to other scenarios where there are very large tables or tables that contain a lot of corrupt data.


Warning: Unresponsive plugin

Shockwave Flash may be busy, or it may have stopped responding. You can stop the plugin now, or you can continue to see if the plugin will complete.

☐ Don't ask me again

First take note of the number of rows in each of these maintenance tables:

select count(*) from maintenance_window;
select count(*) from maintenance_schedule;

A high number of expired maintenance schedules can slow down USM performance, e.g., > 300. You can check for those via the USM if it is responsive enough or in the database under the table MAINTENANCE_SCHEDULE, and also for schedule entries in the table MAINTENANCE_WINDOW that are not linked to a SCHEDULE_ID.

These entries are being checked when you open the USM, and this can cause slow performance as well. Unfortunately, at this moment the deletion of expired schedules does not cause the entries in the MAINTENANCE_WINDOW to be deleted, and we found that when you do this manually, it will improve USM performance (removing expired schedules from MAINTENANCE_SCHEDULE and orphaned entries from MAINTENANCE_WINDOW.

Even if the row counts are above 300, this could slow down USM performance/response time.

You could then delete from maintenance_window using the END_TIME timestamp as a filter and/or maintenance_schedule using SCHEDULE_NAME or START_TIME as a filter to reduce the number of rows. You can delete based on date, Ad Hoc schedules, etc. to minimize the entries.

Moving forward, UIM development may institute cleanup of the maintenance tables but that has not yet been implemented.

Also pay close attention to the CM_CONFIGURATION_ITEM table as too many rows, e.g., > 1.5M can slow down USM display rendering.

Check this table for dirty data using this query:

```
declare @str varchar(255)
declare @i int
set @str = ''
set @i = 32
while @i <= 127
begin
set @str = @str + '|' + char(@i)
set @i = @i + 1
end

select * from cm_configuration_item
where ci_name like '%[^' + @str + ']%' escape '|'

--then if you see corrupt data, use DELETE FROM to delete those rows.

--delete from cm_configuration_item
--where ci_name like '%[^' + @str + ']%' escape '|'
```

☐ **Check nas table row counts**

- Check the nas tables in the database, e.g., nas_alarms, nas_transaction_summary etc. to see if they have very high row counts, e.g., if they contain more than 3000 rows, it could take time for UMP/USM to load alarms. You can check the tables directly by running a select count(*) from <nas_table> from MS SQL Server Studio to see if it takes a long time to return the results.

☐ **Check UMP Machine resources**

- Check overall resource utilization and resource availability in detail, e.g., CPU, Memory, I/O. Keep in mind that you can implement some self-health monitoring by using probes such as cdm, processes, and other probes to monitor the UMP machine and create performance reports so you can see trends over time, e.g., for CPU, Memory etc. Also note that in some cases a simple restart of the primary hub and UMP can resolve slow performance which may be due to resource utilization issues.

☐ **Check Microsoft SQL Server Memory**

- To support a *large* UIM environment, **you need a minimum of 64 GB of RAM and likely 128 GB RAM with SQL dynamically managing its own memory (not limited).** See Memory->Server Properties in SQL Server. Make sure you are <u>not</u> consistently using all of the allocated SQL Memory, and/or there is no significant room to expand.

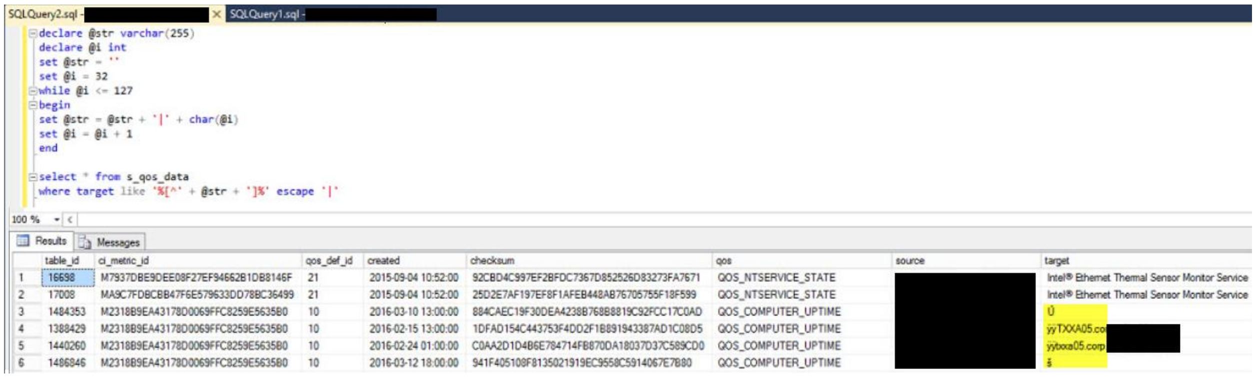
  Helpful links:

- Server Memory Server Configuration Options
  https://msdn.microsoft.com/en-us/library/ms178067.aspx?f=255&MSPPError=-2147217396

- Optimizing Server Performance Using Memory Configuration Options
  https://technet.microsoft.com/en-us/library/ms177455%28v=sql.105%29.aspx

☐ **UMP/USM slowness due to data 'Loading' issues**

- Check for *corrupt* data. Sometimes it is necessary to search for and eliminate corrupt data in tables such as S_QOS_DATA, NAS_ALARMS, CM_COMPUTER_SYSTEM, etc. especially when the UMP/USM cannot display the data or the page displays and remains at "*Retrieving data...*" or the interface/module simply never finishes 'loading' data or hangs. The queries referenced in the Article links below can be run in the SLM portlet or a database tool such as MS SQL Server studio. See examples of corrupt/bad data below highlighted in yellow.



SQLQuery2.sql - [redacted] SQLQuery1.sql - [redacted]

```
declare @str varchar(255)
declare @i int
set @str = ''
set @i = 32
while @i <= 127
begin
set @str = @str + '|' + char(@i)
set @i = @i + 1
end

select * from s_qos_data
where target like '%[^' + @str + ']%' escape '|'
```

	table_id	ci_metric_id	qos_def_id	created	checksum	qos	source	target
1	16698	M7937DBE9DEE08F27EF94662B1DB8146F	21	2015-09-04 10:52:00	92CBD4C997EF2BFDC7367D852526D83273FA7671	QOS_NTSERVICE_STATE	[redacted]	Intel® Ethernet Thermal Sensor Monitor Service
2	17008	MA9C7FDBCBB47F6E579633DD78BC36499	21	2015-09-04 10:52:00	25D2E7AF197EF8F1AFEB448AB76705755F18F599	QOS_NTSERVICE_STATE	[redacted]	Intel® Ethernet Thermal Sensor Monitor Service
3	1484353	M2318B9EA43178D0069FFC8259E5635B0	10	2016-03-10 13:00:00	884CAEC19F30DEA4238B768B8819C92FCC17C0AD	QOS_COMPUTER_UPTIME	[redacted]	Û
4	1388429	M2318B9EA43178D0069FFC8259E5635B0	10	2016-02-15 13:00:00	1DFAD154C443753F4DD2F1B891943387AD1C08D5	QOS_COMPUTER_UPTIME	[redacted]	ÿÿTXXA05.co [redacted]
5	1440260	M2318B9EA43178D0069FFC8259E5635B0	10	2016-02-24 01:00:00	C0AA2D1D4B6E784714FB870DA18037D37C589CD0	QOS_COMPUTER_UPTIME	[redacted]	ÿÿtxxa05.corp [redacted]
6	1486846	M2318B9EA43178D0069FFC8259E5635B0	10	2016-03-12 18:00:00	941F405108F8135021919EC9558C5914067E7B80	QOS_COMPUTER_UPTIME	[redacted]	š

- *How to search for bad/special/non-printable characters in MS SQL Server*
  http://ecm.ca.com/KB/Pages/authoring/create-KB-Article.aspx?kbid=TEC000005335

- *How to search for bad/special/non-printable characters in MySQL* http://ecm.ca.com/KB/Pages/authoring/create-KB-Article.aspx?kbid=TEC000005283

☐ **Anti-Virus**

- Check to make sure that antivirus or network security is not interfering with the connection between UMP and the client browser (Check UMP machine and local browser session.) The UMP Machine should have a FULL exception for all CA UIM (Nimsoft) programs in the Anti-Virus configuration. In some rare cases we have seen AV blocking/scanning when it shouldn't due to a bug in the vendor's AV software.

Additional Information:

UMP Debug Mode

You can enable UMP's debug mode for USM or other portlets exhibiting slow performance/loading. This allows you to reveal the underlying queries being run when the data is being fetched from the database/tables. You can then test/manipulate and test them manually to see how they perform.

Please refer to this Knowledge Article, *"How can I enable DEBUG mode for UMP, USM, ListViewer or Performance Report displays?" at the following link,* for more detailed information:

http://ecm.ca.com/KB/Pages/authoring/create-KB-Article.aspx?kbid=TEC000003890

Poor performing queries
To identify slow running/poor performing queries in general without using debug mode you can run a query like this to generate a table of stats and the poor performing queries:

```
SELECT  top 15
         creation_time
        ,last_execution_time
        ,total_physical_reads
        ,total_logical_reads
        ,total_logical_writes
        , execution_count
        , total_worker_time
        , total_elapsed_time
        , total_elapsed_time / execution_count avg_elapsed_time
        ,SUBSTRING(st.text, (qs.statement_start_offset/2) + 1,
         ((CASE statement_end_offset
          WHEN -1 THEN DATALENGTH(st.text)
          ELSE qs.statement_end_offset END
            - qs.statement_start_offset)/2) + 1) AS statement_text
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
ORDER BY total_elapsed_time / execution_count DESC;
```

Diagnostic Queries

Listed below are a few diagnostic queries to help you assess performance of the backend Microsoft SQL Server as it relates to UMP/USM response times, data/portlet loading and/or performance.

Identify current users, sessions and processes

```
exec sp_who2
```

Provides information about current users, sessions, and processes in an instance of the Microsoft SQL Server Database Engine. The information can be filtered to return only those processes that are not idle, that belong to a specific user, or that belong to a specific session.

Wait Stats

```
SELECT * FROM sys.dm_os_wait_stats ORDER BY wait_time_ms desc
```

Identify current requests

```
select * from sys.dm_exec_requestsfs
```

Returns information about each request that is executing within SQL Server.

How to purge data without running out of memory or resources

There are various ways of effectively purging unnecessary/outdated data from a large database (> 1 TB) but here is one that I found to be effective. This approach should not be used/executed during business hours as it can take significant time to complete.

Make sure that the date is NOT set too far out to delete any records; this (the sampletime value) is a “check valve” to control what data should be deleted.

The sampletime value can be tested and then adjusted to delete a small amount of data of course, and then increased once successful.

-----   purge script   -----
-- Note that the TOP (<value>) should be increased each time until all of the unwanted data has been purged

```
DECLARE qos_def_cursor CURSOR READ_ONLY FAST_FORWARD FOR SELECT DISTINCT
qos_def_id FROM S_QOS_DATA;
DECLARE @sql VARCHAR(1000);
DECLARE @r_table VARCHAR(64);
DECLARE @h_table VARCHAR(64);
DECLARE @qos_def_id VARCHAR(4);

OPEN qos_def_cursor

FETCH NEXT FROM qos_def_cursor INTO @qos_def_id WHILE @@FETCH_STATUS = 0
```

```
BEGIN
      SET @r_table = 'RN_QOS_DATA_' + RIGHT('0000' + @qos_def_id, 4);
      SET @h_table = 'HN_QOS_DATA_' + RIGHT('0000' + @qos_def_id, 4);

      SET @sql = 'DELETE TOP (3000) FROM ' + @r_table + ' WHERE sampletime
< ''2016-06-20 00:00:00.000''';
      PRINT CHAR(13) + CHAR(13) + 'Deleting from ' + @r_table;
      EXECUTE(@sql)


      SET @sql = 'DELETE TOP (3000) FROM ' + @h_table + ' WHERE sampletime
< ''2016-06-20 00:00:00.000''';
      PRINT CHAR(13) + 'Deleting from ' + @h_table;
      EXECUTE(@sql)

      FETCH NEXT FROM qos_def_cursor INTO @qos_def_id;
END

CLOSE qos_def_cursor;
DEALLOCATE qos_def_cursor;

-- check results against a given table
-- select * from rn_qos_data_0001 order by sampletime DESC


-----   end script   -----
```

The script needs to be run with the 'top' argument, and then each time it finishes, increase the top <value> and run it until ALL of the older data has been deleted, e.g., change the 'top' value to:

1000
3000
5000
10000
50000
100000
500000
1000000
5000000

Below are some examples from a production system of how long it may take to purge the data.

3000 records – 4min
10000 records – 7min
50000 records – 25min
100000 records – 40min
500000 records – 1hr30min
1000000 records – 2hr30min
5000000 records – 3hr

As you can see, each run with a higher top value will take longer than the previous run.

You can check the results before and then after the query is executed using:

select count(*) from RN_QOS_DATA_0001 order by sampletime ASC
(and check the date)

Misc:

There is a known UMP/USM performance issue when the browser is displayed on a Mac's Retina Display screen (effective resolution 2880x1800). When the browser is displayed on a non-retina display (resolution 1920x1080), USM performance is very good, as expected. UIM 8.3.1. There is an available hotfix for this issue. If you need it please contact CA UIM Support.