SAP Knowledge Base Article

1840954 - Alerts related to HANA memory consumption

Version 6 **Validity:** 26.06.2013 - active **Language** English

Symptom

Statisticserver alerts are triggered on HANA side:

HANA alert 1: Host physical memory usage
HANA alert 43: Memory usage of services
HANA alert 44: Licensed memory usage

Environment

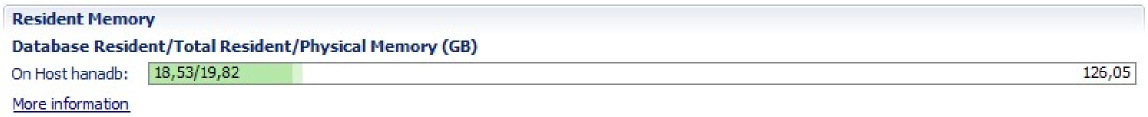
HANA

Resolution

1. Operating System vs. Database

Before proceeding with analysing the different HANA specific memory areas, it has to be clarified that indeed the database and not processes runnin resident memory is used by the HANA database. On the HANA appliance, resident memory used outside HANA (OS, 3rd party processes) is typicall different views on the resident memory:

HANA Studio:



Resident Memory
Database Resident/Total Resident/Physical Memory (GB)
On Host hanadb: 18,53/19,82 126,05
More information

The following SQL statements are behind this output:

'Database Resident':

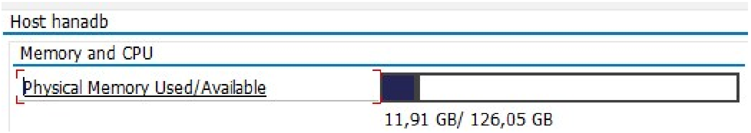
```
SELECT SUM(PHYSICAL_MEMORY_SIZE) FROM M_SERVICE_MEMORY
```

'Total Resident':

```
SELECT T1.HOST, T1.USED_PHYSICAL_MEMORY + T2.SHARED_MEMORY_ALLOCATED_SIZE
FROM M_HOST_RESOURCE_UTILIZATION AS T1
JOIN (SELECT M_SERVICE_MEMORY.HOST, SUM(M_SERVICE_MEMORY.SHARED_MEMORY_ALLOCATED_SIZE) AS SHARED_ME
         FROM SYS.M_SERVICE_MEMORY
         GROUP BY M_SERVICE_MEMORY.HOST) AS T2
ON T2.HOST = T1.HOST;
```

Since the difference between 'Total Resident' and 'Database Resident' is well below 2 GB, there is no indication that processes outside the database

DBACockpit:



Host hanadb
Memory and CPU
Physical Memory Used/Available
11,91 GB/ 126,05 GB

This is effectively the output of the following statement:

```
SELECT HOST,
ROUND(USED_PHYSICAL_MEMORY/1024/1024/1024, 2) AS "Resident GB",
ROUND((USED_PHYSICAL_MEMORY + FREE_PHYSICAL_MEMORY)/1024/1024/1024, 2) AS "Physical Memory GB"
FROM PUBLIC.M_HOST_RESOURCE_UTILIZATION
```

DBACockpit does currently not consider shared memory as part of the resident memory, hence the difference to what *HANA Studio* reports.

If processes outside the database are not contributing significantly to memory usage, 2 cases have to be distinguished in general:

1. An 'out of memory' error has already occurred.
2. An 'out of memory' error was not yet triggered, but further investigation on the current memory usage of the database might be required.

For case (1), the foundation for further analysis is the oom trace file which follows the naming convention <processname>_<hostname>.<number>.rte indexserver_hdbnode1.39503.rtedump.72290.oom.trc).

In case (2), such a trace file is not yet available and an rte dump has to be generated which has a structure similar to the oom trace. SAP note 181302 (2), further analysis can also be done 'top-down', making use of the information provided by the different administrative frontends (HANA Studio / DBA

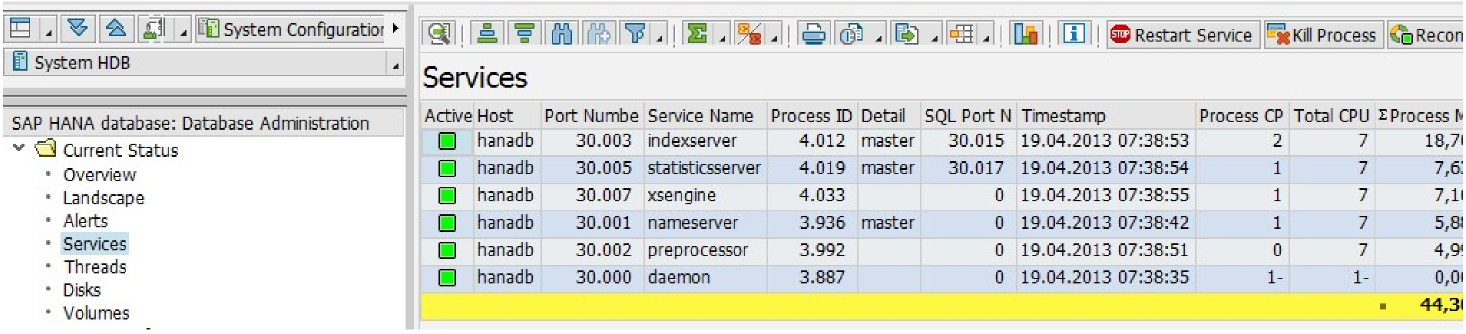
2. Top down approach

2.1 Which HANA process on which host is using most of the memory?

SQL Statement:

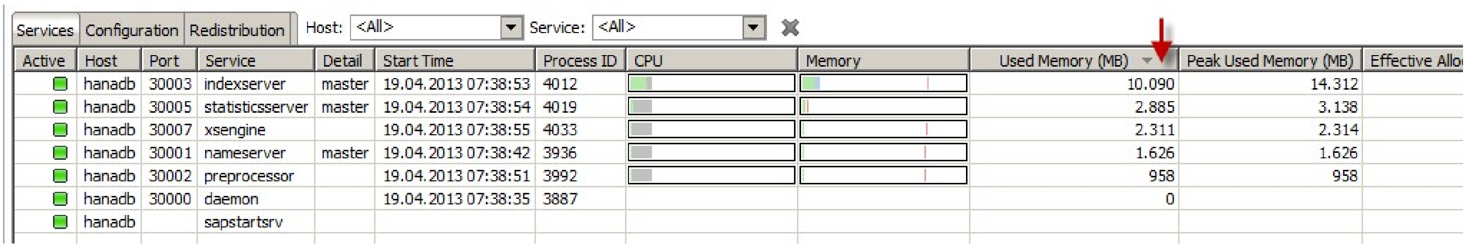
```
SELECT TOP 3 HOST, PORT, SERVICE_NAME, TOTAL_MEMORY_USED_SIZE
FROM M_SERVICE_MEMORY
ORDER BY TOTAL_MEMORY_USED_SIZE DESC
```

DBACockpit:



Active	Host	Port Numbe	Service Name	Process ID	Detail	SQL Port N	Timestamp	Process CP	Total CPU	ΣProcess M
🟩	hanadb	30.003	indexserver	4.012	master	30.015	19.04.2013 07:38:53	2	7	18,7
🟩	hanadb	30.005	statisticsserver	4.019	master	30.017	19.04.2013 07:38:54	1	7	7,6
🟩	hanadb	30.007	xsengine	4.033		0	19.04.2013 07:38:55	1	7	7,1
🟩	hanadb	30.001	nameserver	3.936	master	0	19.04.2013 07:38:42	1	7	5,8
🟩	hanadb	30.002	preprocessor	3.992		0	19.04.2013 07:38:51	0	7	4,9
🟩	hanadb	30.000	daemon	3.887		0	19.04.2013 07:38:35	1-	1-	0,0
										44,3

HANA Studio:



Active	Host	Port	Service	Detail	Start Time	Process ID	CPU	Memory	Used Memory (MB)	Peak Used Memory (MB)	Effective Allo
🟩	hanadb	30003	indexserver	master	19.04.2013 07:38:53	4012			10.090	14.312	
🟩	hanadb	30005	statisticsserver	master	19.04.2013 07:38:54	4019			2.885	3.138	
🟩	hanadb	30007	xsengine		19.04.2013 07:38:55	4033			2.311	2.314	
🟩	hanadb	30001	nameserver	master	19.04.2013 07:38:42	3936			1.626	1.626	
🟩	hanadb	30002	preprocessor		19.04.2013 07:38:51	3992			958	958	
🟩	hanadb	30000	daemon		19.04.2013 07:38:35	3887			0		
🟩	hanadb		sapstartsrv								

2.1.1 On 'Used Memory'

The values in the 'Used Memory' column are calculated in the following way:

```
SELECT ( (CODE_SIZE + SHARED_MEMORY_ALLOCATED_SIZE + HEAP_MEMORY_USED_SIZE ) / 1024 / 1024) AS "USE
FROM M_SERVICE_MEMORY
```

It is important to note that even though the column is called 'Used Memory', the overall allocated shared memory is used for the calculation and not the shared memory segment (see point 4) can only be completely attached to the address space of the process and not just the used part.

Due to changes in the calculation of CODE_SIZE, there might be cases where the used memory increased after an revision upgrade. This can be see

```
SELECT SERVICE_NAME, CODE_SIZE
FROM M_SERVICE_MEMORY
```

rev49:

SERVICE_NAME	CODE_SIZE
nameserver	627.855.360
preprocessor	687.079.424
indexserver	733.585.408
statisticsserver	727.781.376
xsengine	725.540.864

rev54 and later:

SERVICE_NAME	CODE_SIZE
nameserver	4.498.731.008
preprocessor	4.526.575.616
indexserver	5.256.380.416
statisticsserver	4.956.565.504
xsengine	5.230.637.056

Before HANA revision 54, the code size was incorrectly reported. This problem is explained in SAP note 1826139. The majority of the CODE_SIZE i memory is consumed by CODE_SIZE is therefore not correct. Starting with HANA SP 06, an instance wide CODE_SIZE will be reported. Additionall (which is expected to be around 5 GB).

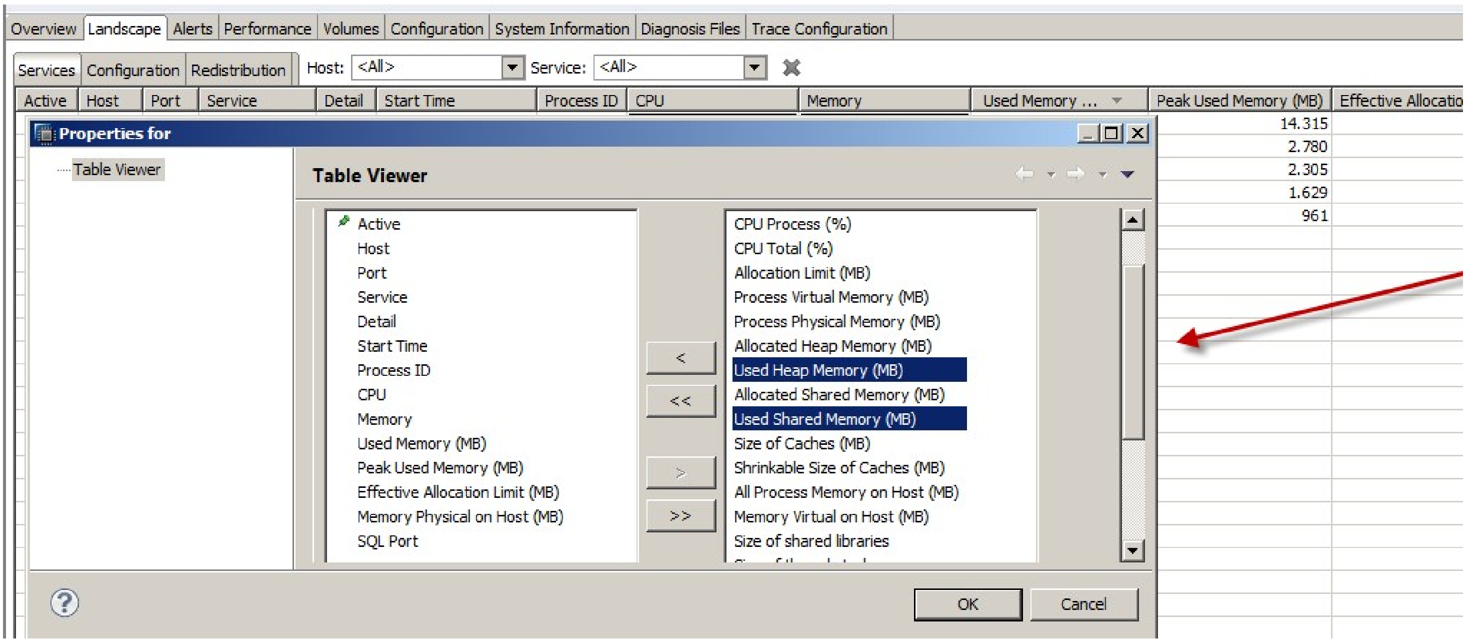
2.2 Is shared or heap memory using the largest share?

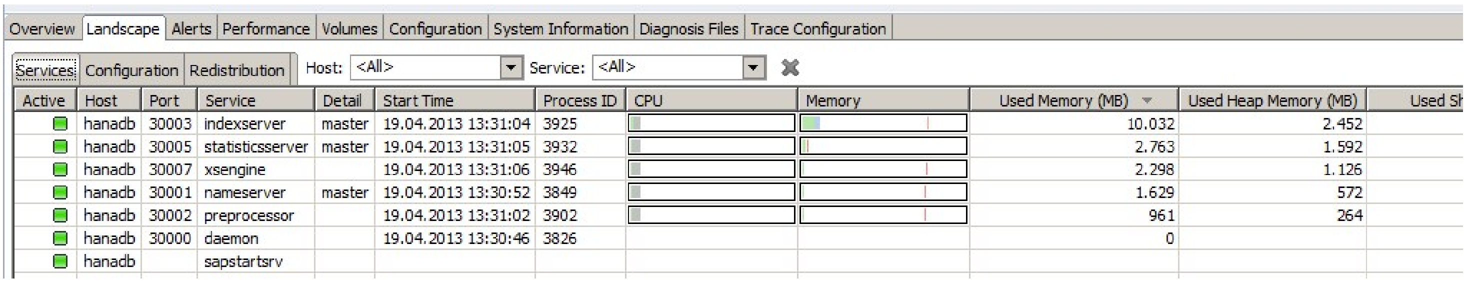
So far, the sum of used heap and shared memory has been determined. Now, this total has to be split:

SQL Statement:

```
SELECT TOP 3 HOST, PORT, SERVICE_NAME, HEAP_MEMORY_USED_SIZE, SHARED_MEMORY_USED_SIZE, TOTAL_MEMORY
FROM M_SERVICE_MEMORY
ORDER BY SHARED_MEMORY_USED_SIZE DESC
```

HANA Studio:





Active	Host	Port	Service	Detail	Start Time	Process ID	CPU	Memory	Used Memory (MB)	Used Heap Memory (MB)	Used Sh
🟩	hanadb	30003	indexserver	master	19.04.2013 13:31:04	3925			10.032	2.452	
🟩	hanadb	30005	statisticsserver	master	19.04.2013 13:31:05	3932			2.763	1.592	
🟩	hanadb	30007	xsengine		19.04.2013 13:31:06	3946			2.298	1.126	
🟩	hanadb	30001	nameserver	master	19.04.2013 13:30:52	3849			1.629	572	
🟩	hanadb	30002	preprocessor		19.04.2013 13:31:02	3902			961	264	
🟩	hanadb	30000	daemon		19.04.2013 13:30:46	3826			0		
🟩	hanadb		sapstartsrv								

If it is shared memory, proceed as outlined in section 'Shared Memory Usage', otherwise, go to section 'Heap memory usage'.

3. Trace file approach

The following sections of those trace files are usually relevant:

[IPMM_MEMORY]

The first part of this section lists the local (heap) memory the processes that make up the HANA database are currently using:

```
[0] PID=34884, SId=42309944, compactors active, alive, process name: hdbnameserver
AB=2220466176b (2.06gb), UA=0b, U=2015851859b (1.87gb), FSL=0b, PAL=487738426982b (454.24gb), TPAL=
[1] PID=35049, SId=42310545, compactors active, alive, process name: hdbpreprocessor
AB=365772800b (348.82mb), UA=0b, U=362430594b (345.64mb), FSL=0b, PAL=487738426982b (454.24gb), TPA
[3] PID=35094, SId=42310846, compactors active, alive, process name: hdbstatisticsse
AB=17623138051b (16.41gb), UA=0b, U=14624613181b (13.62gb), FSL=268435456b (256mb), PAL=27096579276
[4] PID=35114, SId=42310947, compactors active, alive, process name: hdbxsengine
AB=2270855168b (2.11gb), UA=0b, U=2136436039b (1.98gb), FSL=0b, PAL=487738426982b (454.24gb), TPAL=
[5] PID=33976, SId=171197412, compactors active, alive, process name: hdbindexserver
AB=240495694077b (223.97gb), UA=0b, U=260528715346b (242.63gb), FSL=0b, PAL=487738426982b (454.24gb
```

For each process, the allocation limit (PAL), the amount of memory currently allocated (AB) and used (U) are displayed. An AB value that is significant allocated, but not used memory will be released. Of particular interest are lines where AB, U and PAL have approximately the same value. This partic output of M_HEAP_MEMORY in section [PROCESS_INFO] has to be done then.

The second part of section [IPMM_MEMORY] contains information regarding the shared memory:

```
GLOBAL_MAX_ALLOCATION_LIMIT=487738426983b (454.24gb), cached sum of allocation limits=487604209255b
#checks for cached allocation limit sum=838859, #terminated processes since last reset=7
#processes=5, sum of temp process allocation limits=292676603651b (272.57gb), cached sum=4876042092
SHARED_MEMORY=194927605604b (181.54gb), temp shared memory allocation limit=194927605604b (181.54gb
IPMM reservation=0b, emergency reservation=134217728b (128mb)
112 early exits since creation of IPMM, provide-memory-counter=50279
Provide memory 50267 failed, only 109174531b were free.
Users respected in shm calculation: [302]
```

Shared Memory is used for various purposes. For practical purposes, it is sufficient to assume that mainly row store tables occupy shared memory. In the inability to further increase the TPAL (temporary allocation limit) of the failing process. The allocation limit of a process is not set to the maximum case, 'sum of temp process allocation limits' + SHARED_MEMORY exceeds the GLOBAL_MAX_ALLOCATION_LIMIT which consequently causes t

[MEMORY_OOM]

Before any detailed analysis is done, the information from section [MEMORY_OOM] has to be reviewed as well. It might be the case that the request

```
[MEMORY_OOM]  Information about current out of memory situation:
OUT OF MEMORY occurred.
Failed to allocate 2405843009873693952 byte.
```

A known issue regarding such large request is solved with revision 50. In case this occurs in newer revisions, please contact SAP Support.

The following checks can then be done based on the information available so far:

4. Shared Memory Usage

Contrary to heap memory, which is allocated using the 'malloc' system call, shared memory is provided using the 'shmget' call. The results of shared m following approach can be used to display the shared memory of one particular HANA process on operating system side:

1. Get the process pid: ps -ef | grep <HANA process>
2. ipcs -p | grep <pid> then displays all segments that were created by this particular process:

```
ipcs -p | grep 4221
86999065   hanadm     4221       4221
87064602   hanadm     4221       4221
87130139   hanadm     4221       4221
87195676   hanadm     4221       4221
87261213   hanadm     4221       4221
87359519   hanadm     4221       4221
88309819   hanadm     4221       4221
88375356   hanadm     4221       4221
88440894   hanadm     4221       4221
```

3. The size of a particular segment can then be further examined using the command ipcs -m -i <id>

```
ipcs -m -i 86999065
Shared memory Segment shmid=86999065
uid=60000       gid=100 cuid=60000      cgid=100
mode=01600      access_perms=0600
bytes=8929752   lpid=4221       cpid=4221       nattch=1
att_time=Tue May 14 14:09:13 2013
det_time=Tue May 14 14:09:13 2013
change_time=Tue May 14 14:09:13 2013
```

The sum of all those shared memory segments is then equivalent to the output of the statement:

```
SELECT SHARED_MEMORY_ALLOCATED_SIZE
FROM M_SERVICE_MEMORY
WHERE PROCESS_ID = '4221'
```

4.1 Memory usage of the row store

1. The row store is organized in 64MB segments. Deleting large number of rows can lead to sparse segments and unnecessary use of memory. S
2. Indirectly, high memory usage of the row store can cause problems when parameter *client_distribution_mode* is set to *'off'* in distributed enviror where table is located. With the setting 'off', the statement might then also be directed to the master node. Since all row store tables are usually l materializing a large amount of data on the master node (from a table that is actually located on another node) can then simply be too much. The parameter *client_distribution_mode* should not be turned off, but instead the recommendations from SAP note 1780950 are to be followed.
3. Too many large tables were created as row store tables. The largest tables currently contained in the row store can be retrieved by the following

```
SELECT TOP 50 *
FROM M_RS_TABLES
ORDER BY (ALLOCATED_FIXED_PART_SIZE +
ALLOCATED_VARIABLE_PART_SIZE) DESC
```

It is however important to keep in mind that in many scenarios, switching a table from rowstore to columnstore and vice versa must not be done. settings for different table types. At any case, it makes sense to check whether the system is currently adhering to the recommended configuratio rowstore and a check that is based on this list is part of report *RSDU_TABLE_CONSISTENCY*. This report also contains a repair option when

5. Heap memory usage

If it can be ruled out that the rowstore is reponsible for high memory consumption, the heap memory allocation of the individual processes has to be a usually allocates the greatest share of the available memory. The following possibilities exist to do this:

1. If an oom error has already occurred or a rte dump was explicitly triggered, the content of the system view M_HEAP_MEMORY (section[PROCE
2. If an adhoc analysis of the heap memory consumption should be done, the view M_HEAP_MEMORY can be queried directly, for example using

In case of (1), the output can be directly copied into a csv file and further be analyzed for example using excel. Each line represents an allocator, whicl used on M_HEAP_MEMORY then:

```
SELECT TOP 15 MS.HOST, MS.SERVICE_NAME,MH.CATEGORY, MH.INCLUSIVE_SIZE_IN_USE, MH.EXCLUSIVE_SIZE_IN_
FROM M_HEAP_MEMORY MH, M_SERVICES MS
```

```
WHERE MH.PORT = MS.PORT AND
           MH.HOST = MS.HOST AND
           MS.SERVICE_NAME = 'indexserver'
ORDER BY 4 DESC
```

Of interest are those pools that are on the top of the list and have an inclusive size close to the exclusive size. Depending on the pool names, the follo

Pool/parallel/ihm

There are cases where this allocator used 90% of the available memory. If such a problem is observed, the system should be upgraded to revision 54

Pool/itab

This allocator is needed for the intermediate result of join operation and translation table used for join operation. This could indicate use of a suboptim

Pool/FemsCompression/CompositeFemsCompressionPool/FemsCompression/CompositeFemsCompression

A known issue is associated with high memory usage that is solved with revision 53. FEMS compression is used in BW systems to keep the size of t

Pool/ValueArray

This is the allocator which keeps the complete resultset in uncompressed form. A known issue has been solved with revision 55 which addresses me increasing over time. The details are documented in SAP note 1852425.

Pool/JoinEvaluator/TranslationTable

Translation tables are created for caching join column entries of join engine queries. The number of translation tables that are kept in memory is deter

```
[joins]
translator_cache_size = n
```

The default value is 2000. In case of excessive memory usage for this allocator, the parameter can be set to 500. From SP6 onwards it will be possib

Pool/malloc/libhdbcsapi.so

A memory leak was introduced with revision 54 that can lead to a huge growth of this allocator when BW queries containing lots of OR terms instead implementation of SAP note 1786777. From HANA side, the problem is solved starting with revision 60 (HANA SP6). See also SAP note 1871239.

6. Memory Usage of the Statisticsserver

Even though the statisticsserver can also run into oom errors, it should not consume a large share of the available memory. The statisticsserver is sup following two configuration parameters are relevant:

```
statisticsserver.ini
  -> memorymanager
         allocationlimit
         minallocationlimit
```

The allocation limit of the statisticsserver is either a percentage of the availably RAM or just the value of minallocationlimit, depending on what is the b reached its process allocation limit, it can make sense to increase the allocationlimit to 10% or 15% to have at least a workaround for the time being. (see above). In the oom trace file, such a situation would look like this:

```
[3] PID=23840, SId=320543005, compactors active, alive, process name: hdbstatisticsse
    AB=6755876864b (6.29gb), U=7135415505b (6.64gb), FSL=0b, PAL=6772669235b (6.30gb), TPAL=6772669
```

AB, PAL and TPAL are on the same values. Also in this case, further investigation regarding the involved pools is necessary, either using the informa

```
SELECT TOP 15 MS.HOST, MS.SERVICE_NAME,MH.CATEGORY, MH.INCLUSIVE_SIZE_IN_USE, MH.EXCLUSIVE_SIZE_IN_
FROM M_HEAP_MEMORY MH, M_SERVICES MS
WHERE MH.PORT = MS.PORT AND
         MH.HOST = MS.HOST AND
         MS.SERVICE_NAME = 'statisticsserver'
ORDER BY 4 DESC
```

Header Data

Released On	26.06.2013 12:35:06
Release Status	Released to Customer
Component	BC-DB-HDB SAP HANA database
Priority	Normal
Category	Problem
Database	HDB 1.0

Product

This document is not restricted to a product or product version

References

This document refers to:

SAP Knowledge Base Articles

1813020	How to generate a runtime dump on SAP HANA

CSS SAP Notes

1813245	SAP HANA DB: Row store reorganization
1852425	SAP HANA appliance: Revision 55 of SAP HANA database
1780950	Connection problems due to host name resolution
1659383	RowStore list for SAP NetWeaver in SAP HANA database