Service Manager Diagnostics and Tuning

Best Practices for diagnosing issues in Service Manager and tuning Service Manager for best performance

Introduction

Service Manager®, the premier consolidated service desk offering from HP Software, is a comprehensive and fully integrated IT Service Management (ITSM) software product that enables IT professionals to improve service levels, balance resources, and optimize costs. Along with the ability to have a custom-made service desk come both the need to ensure that tailoring does not negatively impact the system's performance, and the ability to diagnose potential issues.

When collecting requirements for an implementation, it is important to remember that the business process drives the tool, and not the other way around. To simplify implementation, Service Manager contains pre-existing, ITIL®-certified best practice guidelines for the Service Desk, Incident Management, Change Management, and Problem Management modules.

When you approach an implementation, begin with a detailed process diagram. Present the out-of-the-box settings as an option to enhance your business processes. Conduct an exercise that includes input from members of all areas that will use the system to make sure that business processes flow smoothly. Then, Service Manager will be readily accepted when implemented; and only a one-time setup is required, so that you do not have to make adjustments and enhancements after every user acceptance test.

When system features are customized without proper planning and careful implementation, there are associated costs, including system errors and performance concerns. It is important to remember that most performance issues will only manifest themselves under high load.

The purpose of this guide is to raise awareness of those factors that can cause performance concerns if they have not been considered during the design and implementation phases of a project. Moreover, the goal of this document is to outline best practices to ensure the smooth operation of your system and to set the stage for a stable, scalable, and efficient system. Finally, this white paper discusses both diagnostic tools and best practices to ensure that the tailored system performs properly.

Prerequisites

You must have extensive knowledge of Service Manager and System Administrator access to the Service Manager system.

Planning, design and implementation

Size the server hardware appropriately

For specific information for Service Manager server hardware configurations please refer to the Service Manager sizing reference guide. In addition, the following guidelines provide a usable metric:

1. Add up the requirements for each and every application on the server including the operating system.
2. Review the performance information and memory usage for each product to determine minimum requirements based on use.
3. Determine what expectations there will be for each application. If one application talks to another, consider each as a separate user/process rather than a single one.

Remember that additional servers are needed to act as development and test environments. The hardware requirements for those environments need not be the same as those for the production environment unless these environments need to be an exact duplicate of the production environment. If stress testing is to be done, it is recommended that the load/stress testing environment mimics the production environment to provide more accurate test results.

Prepare for database growth

Make sure to allow for growth of the database according to your projected data volume, allow for dynamic allocation of additional space in the Service Manager table space.

Service Manager Diagnostics

This section discusses Service Manager parameters that can be used to diagnose an issue, debug a process, or get an overview of the software's health. If an issue occurs that needs detailed diagnosis by HP Software Customer Support, send your representative all requested information, which usually includes the `sm.log` file, captured outputs, and the `sm.alert.log` file.

Reports

Service Manager offers multiple reports that help you monitor the state of your system. To generate these reports enter the command as listed below in a command prompt (such as the DOS command prompt) from the Service Manager Server RUN directory. The reports, and the parameters that generate them, are listed and discussed below:

sm -reportcache

The `sm -reportcache` command generates a report that displays the Service Manager Cache statistics. Sample output:

```
---------------------- Cache Statistics      ----------------------

Slot use: 66%; Average Slot Depth: 1; Maximum Slot Depth: 8
```

The cache is organized as a tree. The slot depth indicates the amount of levels deep within the tree. If the tree has too many levels, finding a single branch takes too long. Increasing the tree width to minimize the number of levels deep will make searches against the cache more efficient.

If the average slot depth is higher than 5, HP recommends that you increase the cache_slots parameter in the `sm.ini` file. The cache slots describe the width of the tree. The default cache_slots value is 2003 and this parameter should always be a prime number for optimum performance. If the slot depth is a lot larger than the recommended depth, set the cache_slots parameter to a significantly larger number, until the average slot depth arrives at a number less than 5. Another related parameter is the cache_clean_interval parameter. This parameter determines the interval at which the cache_slots will be cleaned up for reuse.

sm -reportdbstats

The `sm -reportdbstats` command generates a report that displays database usage statistics. To gather database statistics, include the dbstats parameter in the `sm.ini` file and restart the Service Manager server. Sample output (excerpt):

```
---------------------- Database Statistics      ----------------------

Filename  Selects  Inserts  Updates  Deletes  Counts  Sorts  Finds
Cache Inits  Cache Terms  Cache Finds
format          0        0        0        0       0      0     27
          0        1              150
displayoption   0        0        0        0       0      0      0
          0        1                0
info           88        1        1        0       0      0      0
         80          94             0
triggers       69        0        0        0       0      0      0
          0           1             0
stathistory     8        8        0        0       0      0      0
         40          56             0
```

sm -reportipc

The `sm –reportipc` command is an alias for the `sm –reportsem` command. See the section *sm -reportsem* below for detailed information about the report that both commands generate.

sm -reportlbstatus

The `sm –reportlbstatus` command generates a report that displays the LoadBalancer status. It returns information only when running in vertical or horizontal scaling mode. If not running in scaling mode, it returns the message "Couldn't obtain loadBalancer info." Sample output:

```
Load Balancer Status:Fri Aug 29 09:41:13 PDT 2008
HP Service Manager LoadBalancer Running on Host:server.domain.com
Port:13701
List of Hosts:

HostName: server.domain.com
        httpPorts:null
        httpsPorts:null
        maxprocesses:5
        threadsperprocess:50
---------------ServletNodes-----------------------------------------------
 Process  ClusterAddress    Http   Https  Sess  Debug  Quiesce  Load
 ID                         Port   Port   ions  Mode   Mode     Balancer
 1128     192.168.1.1:2361  13704  13705  0     N      N        N

 4296     192.168.1.1:2355  13702  13703  0     N      N        N

 5372     192.168.1.1:2351  13701  13702  0     N      N        Y

---------------ClassicNodes-----------------------------------------------
 ProcessID     ClusterAddress
    4752      127.127.1.1:2367
    520       127.127.1.1:2372
```

sm -reportlic

The `sm –reportlic` command generates a report that displays the Service Manager license information, such as its expiration date, its licensed platforms and features, and license usage both in total numbers and per licensed module at the time of the report. If this report indicates that all licenses are in use, no more users can log into the system.

This report helps you troubleshoot problems you may encounter trying to use certain features. If a feature is not listed in the licensed modules, the user is not authorized to use that feature. However, SysAdmin users may sometimes have access to features for which they are not licensed. Sample output:

```
--- HP Service Manager License Report ---

Permanent License.
Server Quiesced State    :  Allow All Logins

Licensed Module Useage                          Named(Licensed)     Float(Licensed)
 IR Expert(Foundation)                             Enabled
 Configuration Management(Foundation)            0(      25)         0(      25)
 Desktop Administration(Foundation)              0(      25)         0(      25)
 Self Service Ticketing(HelpDesk)                 UnLimited
 Incident Management(HelpDesk)                   0(     100)         0(     100)
 Service Desk(HelpDesk)                          0(     100)         0(     100)
 Problem Management(HelpDesk)                    0(     100)         0(     100)
 Scheduled Maintenance(HelpDesk)                 0(     100)         0(     100)

 RAD Compiler                                    0(       0)         0(      25)
 Service Catalog                                 0(     100)         0(       0)
 Change Management                               0(     100)         0(     100)
```

```
 Request Management                              0(     100)          0(     100)
 Service Level Management                        0(     100)          0(     100)
Contract Management                              0(     100)          0(     100)
Asset Contracts Management                       0(     100)          0(     100)
Knowledge Management                             0(     100)          0(     100)
Knowledge Management ESS                         0(     100)          0(       0)

 HP SCAuto SDK for MVS(SCAuto)                      Enabled
 HP SCAuto SDK for Unix/Windows(SCAuto)             Enabled
```

The Server Quiesced State can have the following values:

Value	Description
Allow All Logins	No restrictions
Allow System administrators only	Only users with the SysAdmin capability word can log in. Users already in the system are not affected.
Allow none	No additional logins are allowed. Users already in the system are not affected.

sm -reportlanguages

This report lists all the supported language settings:

```
Language              CP    Description
==================== ===== ======================================
iso8859-1             819   ISO 8859-1 (Western European Latin-1)
8859-1                819   ISO 8859-1 (Western European Latin-1)
iso8859-2             912   ISO 8859-2 (Central European Latin-2)
8859-2                912   ISO 8859-2 (Central European Latin-2)
iso8859-5             915   ISO 8859-5 (Cyrillic)
8859-5                915   ISO 8859-5 (Cyrillic)
iso8859-7             813   ISO 8859-7 (Greek)
8859-7                813   ISO 8859-7 (Greek)
iso8859-9             920   ISO 8859-9 (Turkey Latin-5)
8859-9                920   ISO 8859-9 (Turkey Latin-5)
iso8859-11            8741  ISO 8859-11 (Thai)
8859-11               8741  ISO 8859-11 (Thai)
iso8859-15            922   ISO 8859-15 (Euro, Finnish, Estonian Latin-9)
8859-15               922   ISO 8859-15 (Euro, Finnish, Estonian Latin-9)
koi8-r                921   Kod Obmena Informatsiey, Russian, Bulgarian
koi8-u                1124  Kod Obmena Informatsiey, Ukranian
utf-8                 884   UTF-8
utf8                  884   UTF-8
mswin874              874   MS cp874 Thai
mswin932              9932  MS cp932 Japanese (almost Shift-JIS)
mswin936              936   MS cp936 Simplified Chinese
mswin949              949   MS cp949 Korean
mswin950              950   MS cp950 Traditional Chinese
mswin1250             1250  MS cp1250 Central European
mswin1251             1251  MS cp1251 Russian, Bulgarian, Serbian
mswin1252             1252  MS cp1252 Latin/Western European
mswin1253             1253  MS cp1253 modern Greek
mswin1254             1254  MS cp1254 Turkish
mswin1257             1257  MS cp1257 Estonian, Latvian, Lithuanian
sjis                  932   Shift-JIS
```

sm -reportlocks

The `sm -reportlocks` command generates a report that displays current resource locks in the system. If a single resource has multiple lock requests, then users have to wait for the first exclusive

lock to be released. Usually this situation resolves itself within a few milliseconds. If the situation does not resolve itself, the user holding the exclusive lock may have to be forced off the system by issuing a `kill` command from the system status screen. If multiple users have a lock on an IR (Information Retrieval) file, this may indicate either a very inefficient query being executed or a corrupt IR file that must be repaired with an IR regen. Sample output:

```
--- Resource Locks ---

Resource Name: ocml;1002                          Number: 0
PID    Session TID       Lock Request Time    Type      Killable Waiting
005780 000022  00001276  03/21/2007 07:39:52  Exclusive        Y       N

Resource Name: ocmq;Q1001                         Number: 0
PID    Session TID       Lock Request Time     Type     Killable Waiting
005780 000022  00001276  03/21/2007 07:39:52   Exclusive        Y       N

Resource Name: probsummary;IM1007                 Number: 0
PID    Session TID       Lock Request Time     Type     Killable Waiting
005780 000022  00001276  03/21/2007 07:39:15   Exclusive        Y       N

Resource Name: agent:KMUpdate                     Number: 0
PID    Session TID       Lock Request Time     Type     Killable Waiting
001508 000021  00004332  03/21/2007 06:29:07   Shared           Y       N

Resource Name: agent:linker                       Number: 0
PID    Session TID       Lock Request Time     Type     Killable Waiting
001508 000014  00004168  03/21/2007 06:29:00   Shared           Y       N

Resource Name: agent:lister                       Number: 0
PID    Session TID       Lock Request Time     Type     Killable Waiting
001508 000013  00005304  03/21/2007 06:28:59   Shared           Y       N

Resource Name: agent:problem                      Number: 0
PID    Session TID       Lock Request Time     Type     Killable Waiting
001508 000008  00004196  03/21/2007 06:28:53   Shared           Y       N
```

sm -reportsem

The `sm -reportsem` command generates a report that displays Service Manager semaphore information. The `sm -reportipc` command generates the same output. The report indicates whether a semaphore is available and, if so, how often it was used. Each semaphore is responsible for one or many shared memory categories. To release a hung semaphore, use the following command: `sm -releasesem:`*n*, where *n* is the number of the semaphore. (For example, the User chain semaphore has number 6.) Sample output:

```
pid (3444) HP Service Manager diagnostic report follows:
--- Reportsem ---

[0]System               Available  count(22)
[1]Application cache    Available  count(1108)
[2]Shared memory        Available  count(6889)
[3]IR Expert            Available  count(2)
[4]Licensing            Available  count(677)
[5]Resource manager     Available  count(2424)
[6]User chain           Available  count(71)
[7]Cache manager        Available  count(73276)
[8]Publish/Subscribe    Available  count(10)
[9]Database Services    Available  count(45146)
[10]Alert Services       Available  count(0)
[11]User Stats Chain     Available  count(40)
```

sm -reportshm

The `sm -reportshm` command generates the Shared Memory report, which shows the Service Manager server's release information, the current size of shared memory as defined in the `sm.ini` file, segment and large block allocation, total unused and free space, as well as total allocations and detailed allocation information per shared memory category. The unused space and free space totals are usually not identical, but should be very close. A large discrepancy between free and unused space (>20 %) indicates a problem. If the unused space falls below 25%, HP Software recommends that you increase the amount of shared memory defined in the `sm.ini` file. To adjust the shared memory in the sm.ini the Service Manager server must be down. Refer to the Service Manager Shared Memory Guide whitepaper for an initial shared memory size recommendation. To get the best shared memory size for your system, start with this recommendation and then generate a Shared Memory report at a time of highest load after the system has been running for a few days. Adjust the shared memory setting so that at the highest load free space is between 35% and 45%. Sample output:

```
pid (4328) HP Service Manager diagnostic report follows:

------ Shared Memory ------

Shared Memory Release     7.1

Current Size              32000000

Segment Allocation         3837400
Large Block Allocation     4024064

Unused Space              24138536  (75%)
Free Space                24357104  (76%)


Shared Memory Type  Allocations       Frees   Allocated
------------------  -----------  ----------- -----------
Not named                   299         191       56128
User blocks                   0           0           0
Messages                      0           0           0
Resource locks              165          84        7696
Database Services            78           0       10224
Cache overhead                3           0       10256
Application cache          1291         163     1523728
DBDICT cache               3653           1     4833824
SQL descriptor cache        166           0      308112
Join/ERD/Type cache         590           0      779904
String Type cache           180          10       23008
JavaScript Members            7           0         720
IR Expert cache             370           0       89248
Publish/Subscribe             2           0          48
Web cache                     0           0           0
Threads                       0           0           0
```

sm -reportpub

The `sm -reportpub` command generates the Publish House report, which lists the Marquees that are regularly updated by the system and their values. Although the report still exists in Service Manager 7, the output will say -- Publish House is empty --, since Marquees are no longer used in Service Manager.

sm -reportstatus

The sm –reportstatus command has three sections: Shared Memory, Semaphores and Processes. The Shared Memory section displays information on shared memory such as the size, the starting address (Mapped at), and the Owner of system resources. In the Semaphores section, it displays the

Semaphore Name, Owner, and Grant information. The Processes section lists the owner and user of each process, the user ID of the owner (UID), the process thread ID (PTID), the process ID (PID), the NTID, a name for the process, idle time, whether it is started locally or remotely, and the command that started the process. Sample output (excerpt):

```
pid (7096) HP Service Manager diagnostic report follows:

---- Shared Memory ----

Release  : 7.1.0
Mapped at: 0x03020000
Size     : 0x01E84800 - 32000000 bytes

Resource Name: SM.SHRM.13701

Owner: DOMAIN\user
Group: DOMAIN\Domain Users

Grant: NT AUTHORITY\Authenticated Users rw-      ReadControl WriteDacl
WriteOwner
Grant: NT AUTHORITY\SYSTEM       rw-     ReadControl WriteDacl WriteOwner

---- Semaphores ----

Semaphore Name: SM.LOCK.13701.0

Owner: DOMAIN\user
Group: DOMAIN\Domain Users

Grant: NT AUTHORITY\Authenticated Users ---     ReadControl WriteDacl
WriteOwner
Grant: NT AUTHORITY\SYSTEM       ---     ReadControl WriteDacl WriteOwner

---- Processes ----

Owner   User   UID PTID    PID   NTID  Name                     Idle
Lc/Rmt Command
user    user     1   -1   5372   5412  ThreadControllerId-13701 02:09:03
Local  sm -loadBalancer -httpPort:13701
user    user     2   -1   4296   5536  ThreadControllerId-13702 02:09:04
Local  sm -httpPort:13702 -httpsPort:13703
user    user     3   -1   1128   5148  ThreadControllerId-13704 02:09:00
Local  sm -httpPort:13704 -httpsPort:13705
user    user     4   -1   4752   6048  system.start             02:08:39
Local  sm system.start
user    user     6 6936   4752   6936  spool                    00:03:57
Local  sm system.start
```

sm -reportvirtualmap:pid

The `sm -reportvirtualmap:pid` command generates a report that displays the virtual memory map for a single process, on Windows systems only. It writes output to the `sm.log` file. This report can be used to determine the shared_memory_address parameter if Service Manager fails to start due to an address conflict. To find the best address, first convert your shared memory size to a hexadecimal number, such as 0x01E84800 – 32000000. Then go through the list as shown below and search for an area that has at least as much free space. The same technique can be used on a single process ID (PID) if that PID has issues loading into memory. For example:

```
    0x7C9C1000-0x7CBBD000 (0x001FC000) r-x   i
C:\WINDOWS\system32\VERSION.dll
    0x7F6F0000-0x7F6F7000 (0x00007000) r-x   s UNKNOWN
```

Calculation: `0x7cbc0000 (end of shell32.dll + padding) + 0x01E84800 (size needed for shared memory) = 0x7EA44800`

Based on this calculation, the end address of shared memory, if we set the start at 0x7cbc0000 behind the shell32.dll will still leave room to grow until we reach the next used memory segment at 0x7F6F0000.

Sample output (excerpt):

```
 DUMPING VIRTUAL MEMORY MAP FOR PID 1776: sm system.start
   **Process Memory Map** - User address space extends from 0x00010000 to
0x7FFEFFFF
     Start      End        (Size      ) Name
     0x00010000-0x00011000 (0x00001000) rw-     UNKNOWN
     0x00260000-0x00276000 (0x00016000) r--   s
\Device\HarddiskVolume1\WINDOWS\system32\unicode.nls
     0x00280000-0x002BD000 (0x0003D000) r--   s
\Device\HarddiskVolume1\WINDOWS\system32\locale.nls
     0x002C0000-0x00301000 (0x00041000) r--   s
\Device\HarddiskVolume1\WINDOWS\system32\sortkey.nls
     0x00310000-0x00316000 (0x00006000) r--   s
\Device\HarddiskVolume1\WINDOWS\system32\sorttbls.nls
     0x00350000-0x00351000 (0x00001000) r--   i
C:\scs\sm701\server\RUN\pthreadVC2.dll
     0x00370000-0x00373000 (0x00003000) r--   s
\Device\HarddiskVolume1\WINDOWS\system32\ctype.nls
     0x00380000-0x0038A000 (0x0000A000) rw-     UNKNOWN
     0x00400000-0x00401000 (0x00001000) r--   i
C:\scs\sm701\server\RUN\sm.exe
     0x00410000-0x00411000 (0x00001000) r--   i
C:\scs\sm701\server\RUN\LIBEAY32.dll
     0x00411000-0x004B2000 (0x000A1000) --x   i
     0x01330000-0x01430000 (0x00100000) rw-     UNKNOWN
    Size of IMAGES          :  73637888
   Size of shared memory   :  35368960
   Size of readonly memory :      8192
   Size of writable memory :  79667200
   Size of other memory    :  16297984
   TOTAL SIZE              : 204980224
RTE D Total number of recorded stacks: 0
```

The memory indicated here is used by all threads in the process. The shared memory number indicates the amount of shared_memory in the `sm.ini` file.

sm.alert.log parameters

The `sm.alert.log` file can be found in the Service Manager server logs directory or in the path specified by the -alertlog:PATHSPEC parameter. It contains information about system performance such as queries that take a long time to execute, and system health such as shared memory shortage.

The following alert categories are entered into the `sm.alert.log` file:

- Limits
- Mapping
- Performance
- Stalled

Each of these alert categories has a list of alert types that describe the conditions that triggered the alert (missing numbers are obsolete in Service Manager). These alert types are:

- Limits
    - Limits-2: A user's virtual memory usage exceeds the alertvirtuallimit parameter. The alert item lists the name of the user.
    - Limits-3: A user's CPU usage exceeds the alertcpulimit parameter. The alert item lists the name of the user.
    - Limits-4: The system shared memory is critically close to full.
- Mapping
    - Mapping-1: A database field has been truncated and mapped in a compressed format. The alert item lists the truncated file and field names.
    - Mapping-2: There is a duplicate mapping for a single SQL field. The alert item lists the file name with the duplicate mapping. The alert text lists the field names with the duplicate mapping.
    - Mapping-3: A query could not be translated into SQL. The alert item lists the file name. The alert text lists the query and the function that could not be translated into SQL.
    - Mapping-4: There is a field that cannot be used in an SQL query because of its data type. The alert item lists the file and field names.
- Performance
    - Performance-1: There has been a non-keyed query request that exceeds the alertquerylimit parameter. The alert item lists the name of the file that was the target of the query. You can avoid this alert by creating a key to satisfy the query.
    - Performance-2: There has been a partially keyed query request that exceeds the alertquerylimit parameter. The alert item lists the name of the file that was the target of the query. You can avoid this alert by creating a key to satisfy the query.
    - Performance-3: The system has been waiting for a query to return a result but the query has exceeded the alertwaitlimit parameter. The alert item lists the name of the lock.
    - Performance-4: The system has been waiting for a lock to release a resource, but the lock has exceeded the alertholdlimit parameter. The alert item lists the name of the lock.
    - Performance-5: There has been a query request that exceeds the alertquerylimit parameter. The alert item lists the name of the file that was the target of the query. You can avoid this alert by creating a key to satisfy the query.
    - Performance-6: There has been a query request that exceeds the alerthitratio parameter. The alert item lists the name of the file that was the target of the query. You can avoid this alert by creating a key to satisfy the query.
- Stalled
    - Stalled-3: The IR irqueue is stalled. The number of records in the irqueue exceeds the alertirqueuelimit parameter.
    - Stalled-6: The IR irqueue is stopped. The first record in the irqueue is not changing.

sm -alertquerylimit:nnnn

If the number of milliseconds a Service Manager query takes to execute exceeds the number defined in this parameter, a message is written to the `sm.alert.log` file. The default value is 0, which turns off this type of alert. The message will look like the following:

```
RTE I Performance-5-probsummary, Query (open.time>'01/01/01 00:00:00' and
open.time<'01/01/08 00:00:00' and update.time>'01/01/01 00:00:00' and
update.time<'01/01/08 00:00:00') took 328 milliseconds to complete;
user(falcon), application(cc.search), panel(select)
```

sm -alerthitratio:nn%

If the hit ratio for records inspected to records selected during a query exceeds the percentage specified, an alert will be written to the `sm.alert.log` file. The default value is 90%. The message will look like the following:

```
RTE I Performance-6-problemtype, Hit Ratio not achieved on file
problemtype and query (active=true and limited.given.level2#"enquiry" and
company="DEFAULT"):  Of 157 records checked, 154 did not match the query
; user(falcon), application(display), panel(fdisp.1)
```

*sm -alertwaitlimit:*nnnn

If a user had to wait longer than *nnnn* milliseconds for a lock to be released, an alert is written to the `sm.alert.log` file. The default value for this type of alert is 0, which means that no alerts are being issued.

```
RTE I Performance-4-probsummary;IM1001, Held resource
(probsummary;IM1001) for 34860 milliseconds ; user(falcon),
application(se.unlock.object), panel(unlock)
```

*sm -alertholdlimit:*nnnn

This parameter defines the number of milliseconds (*nnnn*) a lock can be held before writing an alert to the `sm.alert.log` file. By default, the alert is turned off (value 0).

```
RTE I Performance-4-probsummary;IM1001, Held resource
(probsummary;IM1001) for 34860 milliseconds ; user(falcon),
application(se.unlock.object), panel(unlock)
```

The following three parameters monitor the size of the Service Manager internal queue files:

*sm -alertirqueuelimit:*nnnn

If the number of records in the `irqueue` file exceeds *nnnn*, an alert is written to the `sm.alert.log` file. By default, this type of alert is disabled (with default value 0).

*sm -alertcpulimit:*n

This parameter defines the number of standard deviations from the mean CPU usage that cause the Service Manager server to issue an alert message. The default value is 5.

*sm -alertvirtuallimit:*nnn*K*

This parameter defines the number (*nnn*) of kilobytes of virtual memory used that cause the Service Manager server to issue an alert message. This alert is turned off by default (value 0).

sm -alertfilters

This parameter defines the names of alerts that you want the Service Manager server to filter out of alert messages so that Service Manager no longer posts alert messages from these alert categories into the `sm.alert.log` file. By default, all activated alerts are written to the `sm.alert.log` file, with no filtering.

You can use an asterisk (*) at the end of the alert name to specify all alerts within that category. For example, to filter out all performance alerts, type: alertfilters:Performance*.

You can use a semicolon to list additional alert names. For example, to specify all mapping alerts and a long running query alert against the `contacts` file, type:

alertfilters:Mapping*;Performance-5-contacts

sm –util

The Service Manager Database utilities can be used to perform actions similar to the Database Manager utility within Service Manager. Use only as directed by customer support

Important: The reset option will delete all records within the table, the remove option will delete all records as well as the dbdict. Use these options only when directed to by customer support.

```
HP Service Manager Database Exerciser
(Version: 7.01.053 Build: 053)  [09/15/2008 14:48:47]


 add: add        cls: close      cnt: count
 del: delete     dis: display    fnd: find
 get: get        nxt: next       opn: open
 pat: patch      prv: previous   qbe: qbe        rck: read keys
 reg: ir regen   rst: reset
 rmv: remove     upd: update     vrir: verify IR

x: EXIT

Enter your choice:
```

Minimizing debugging output

The secret to successful debugging is to debug only the error situation, avoiding log files that are too big to process. To debug only the error situation, first find out if the issue occurs on a user process or a background process.

Debugging Foreground / User processes

If the issue is on a user process, add the debugging parameters to the server `sm.ini` file, and then ask the user to connect with the Windows client and immediately remove the debugging parameters from the `sm.ini` again. Ask the user to perform the steps causing the issue, and then log out immediately. This minimizes the number of user connections being traced. Another possibility is to start a separate debugnode for that user to connect to in a load balanced environment. This separate node will have to be started with the debugnode switch, so that the loadBalancer does not distribute processes to that port and the node is only available for direct connections. The listener would be started by entering the following command:

```
sm –httpPort:<port#> -httpsPort:<port#> -debugnode -log:debug.log –RTM:3
```

Then the user can change the connection settings on the client to the new httpPort parameter and re-create the issue.

Debugging background processes

If the issue occurred in a background process, it is best to create a debug background process that can be reused when needed. Follow these steps:

1. Go to Database Manager.
2. Select the Format info.startup.
3. Search for the background process you want to debug, such as alert.startup.
4. Add the word DEBUG to the end of the name; for example, alert.startupDEBUG.
5. Click the Add button.
6. In the array to the right, enter the debug parameters, such as –RTM:3, –debugdbquery:999, and –log:debug.log.
7. Click Save.
8. Go to System Status.

9. Stop the existing background process by using k in the command column in front of the process, or by using s. Click on execute command. If you used s click Stop Scheduler. Using k kills the process immediately, and using s followed by Stop Scheduler stops the process once it is safe (with no more I/O or locks).
10. Click Start Scheduler and select the debug version of the background schedule from the list to start the debug process.
11. Execute the steps leading up to the issue. Stop the debug background process again and start the regular background process as discussed in the previous steps.

These steps limit the number of processes that are traced.

Important: Use the debug parameters necessary, while avoiding "over-debugging" the issue (where you trace additional steps or users that do not help troubleshoot the issue).

Debugging a Web Services connection

Web Services connect to Service Manager via the SOAP API. Debugging Web Services can add a lot of information into the `sm.log` file. To minimize that, you can start a separate node for the Web Service in a load balanced environment and debug only that servlet, similar to debugging a foreground user.

```
sm –httpPort:<port#> -httpsPort:<port#> -debugnode -log:WSdebug.log –
debughttp –debugdbquery:999
```

Once the servletcontainer is started, redo the WSDL2JS (Web Services Description Language to JavaScript tool) from that new port number, recompile all JavaScript®, and recreate the issue.

Which parameters to use

Running debug on the system can have a performance impact and should be used cautiously as described above. This paragraph lists situations commonly encountered, where tracing a single process will help to fix the issue.

- Performance issues:
  - RTM:3
  - debugdbquery:999
  - sqldebug:1
- RAD application issues (such as unrecoverable errors)
  - RTM:3 (if in a trigger, use dbtriggertrace:3,<*filename*> instead)
  - debugdbquery:999
- Issues with data retrieval from a relational database
  - debugdbquery:999
  - sqldebug:1
- Issues with Service Manager server startup
  - debugstartup
  - debugdbquery:999, if the startup issue may be data related
  - For a vertical or horizontal scaling implementation, to check nodes joining the group use logdebuglevel:0 (this trace generally creates large output).

Debugging parameters

Please use these parameters only as directed by HP Software Customer Support. Debugging without specific indications for use of the parameters will negatively impact performance.

Database debugging

debugdbquery

The debugdbquery tracing parameter is called as follows:

- debugdbquery:*n*: Lists all queries in the `sm.log` file that took longer than *n* seconds to complete.
- debugdbquery:999: Lists all queries, and database access in the `sm.log` file.

The debugdbquery trace is most often used to determine the cause of performance issues, or to trace incorrect application behavior.

Sample output:

```
D (0x012B9540) DBACCESS - Cache Init   against file datadict
D (0x012B9540) DBACCESS - Cache Find   against file datadict found 1
record, query: name="info"
D (0x0131D4D0) DBACCESS - Select against file globallists
D (0x012B9540) >DBACCESS - Cache Init   against file datadict
D DBQUERY^F^probsummary(P4)^10^1.000000^F^1^0.341000^"flag#true"^
^0.000000^0.000000
D DBQUERY^F^probsummary(P4)^10^1.000000^I^100^8.462000^
"action#"printer""^ ^0.000000^0.000000 ( , )
```

Explanation:

What was executed	Always the value of DBQUERY, DBACCESS DBFIND or DBCOUNT
Where was it executed	F or B (Foreground or Background)
Which file/table and what database type	The name of the file/table and the type of database in which the file/table resides.
Which index was used to retrieve the data	The number of the index that was selected (position in dbdict) **Note**: An asterisk (*) behind the key number indicates that the key was chosen because of the sort fields and not because of the query fields. The system always favors using a key that satisfies the sort over the query because then a re-sort of the data is not required.
Weight assigned to the chosen Index	The weight calculation is: 1/(pos(field in query) * pos(field in key)^2) The result of the calculation is used to determine which index in the dbdict is most efficient for the query.
Query type	Full (F), Partial (P), True (T), Non keyed (N), or IR key (I)
Record count	The number of records returned by the query. The DBQUERY entry is put into the log after processing the SELECT panel.
Number of seconds to return result	The amount of time it took to return the query results
Query	The query submitted by the user or application
Sort fields	Sort criteria for the returned data (entered by the user or application)
Time to extract the results	The amount of time it took to find all the records that match the query criteria (without sort time)
Time to sort the results	If a re-sort was necessary because an index was not present that matched the requested sort, then the sort time contains the amount of time it took to sort all the retrieved records.

*-ir_trace:*n

This parameter, usually called as **ir_trace:801**, provides detail for IR searches, adds, and updates. It writes the name and path of the IR file, the query or update / add statement, and weighting information into the `sm.log` file.

The parameter takes parameter values from 0 to 900, where values between 850 and 900 should not be used since they produce too much output that affects performance negatively.

- 0: No trace
- 1: Log *select* response times
- 5: Same as 1 plus logs changes in access to IR files (open, close, and reopen)

- 101: Same as 5 plus traces high level accesses to IR (search, insert, update, delete)
- 200: Same as 101 plus logs some stats on terms used by the query
- 201: Same as 200 plus tracing on frequencies of TERMS and weighing of documents
- 801: Same as 201 plus tracing of low-level access to IR (information on terms and keys, for example)
- 850: Same as 801 plus low-level space management calls (allocating space, creating new TERM/DOC, and so forth)
- 900: Same as 850 plus, if using the ir_max_shared parameter, includes shared memory management tracing

Sample output of the ir_trace:801 parameter – Search (excerpt):

```
RTE D IR: Just opened file ..\DATA\ir.probsummary with value 00000000
RTE D IR: Data for file ..\DATA\ir.probsummary loaded using 0x00000000
RTE D IR Query against file ir.probsummary - 0
RTE D parsing test printer
RTE D Dump of PARSING
RTE D 0000: 74657374 20707269 6E746572          [printer    ]
RTE D SearchTerm: printer
RTE D Dump of hex key value
RTE D 0000: 7072696E 746572                     [printer         ]
RTE D SearchTerm: return code = 0 -
RTE D Dump of hex return value
RTE D 0000:                                     [                ]
RTE D Query data: printer
RTE D Dump of Query
RTE D 0000: 74657374 20707269 6E746572          [printer    ]
RTE D Parsed token: printer
RTE D Dump of Token
RTE D 0000: 7072696E 746572                     [printer         ]
RTE D Term value printer located
RTE D ....it is in 1 documents with total frequency of 2
RTE D ....that would give it an IDF-MF weight of 5.955827
RTE D ....that would give it an IDF-ND weight of 5.356709
RTE D ....It has an ADL MF weight of 0.000000
RTE D Normalized values for term printer  mf = 0.565855 nd = 0.574315
RTE D IR: It took 62 milliseconds to read the terms
RTE D Starting to spread term printer with weight 0.565855 to its 1
documents
RTE D document 4630226 weight increased by 0.282927 as it contains term
printer
RTE D document 4630226 weight increased by 0.282927 as it contains term
printer
RTE D document at offset 4630226 added to set with weight 0.565855
RTE D Query with 1 term found 1 documents
RTE D ...which spread to 0 other terms resulting in 0 documents
```

sqldebug:n

This parameter enables a trace of the SQL statements sent to the RDBMS, including settings and performance data in the sql diff statements. It accepts numbers 1 through 4 as parameters, with the higher numbers giving additional information:

sqldebug:1: Logs all the SQL statements executed, logs most calls into the RDBMS API, and queries for defined indexes for any table that is being opened

sqldebug:2: In addition to the sqldebug:1 output, it also dumps out all the bind variables that are not binary (not BLOB, RAW, IMAGE, VARBINARYetc.)

sqldebug:3: In addition to the sqldebug:2 output, this also dumps up to the first 256 bytes of binary bind variables

sqldebug:4: Same as sqldebug:3 with the exception that this option increases the dump to the first 100,000 bytes of binary bind variables

Sample output:

```
RTE   InitHash called for file scldapconfig with 12 fields
RTE   sqociInitSqlda: COUNTER, type: 4-FLOAT (was 2-NUMBER), len: 8
RTE   sqociInitSqlda: LDAPHOST, type: 9-VARCHAR (was 1-VARCHAR2), len: 72
RTE   sqociInitSqlda: LDAPPORT, type: 9-VARCHAR (was 1-VARCHAR2), len: 72
RTE   sqociInitSqlda: LDAPBASE, type: 9-VARCHAR (was 1-VARCHAR2), len: 72
RTE   sqociInitSqlda: LDAPSSL, type: 96-CHAR (was 96-CHAR), len: 1
RTE   sqociInitSqlda: LDAPSSLDBPATH, type: 9-VARCHAR (was 1-VARCHAR2),
len: 72
RTE   sqociInitSqlda: LDAPSSLCLIENTAUTH, type: 96-CHAR (was 96-CHAR),
len: 1
RTE   sqociInitSqlda: LDAPSSLKEYPATH, type: 9-VARCHAR (was 1-VARCHAR2),
len: 32
RTE   sqociInitSqlda: SYSMODTIME, type: 5-STRING (was 12-DATE), len: 21
RTE   sqociInitSqlda: SYSMODUSER, type: 9-VARCHAR (was 1-VARCHAR2), len:
62
RTE   sqociInitSqlda: SYSMODCOUNT, type: 4-FLOAT (was 2-NUMBER), len: 8
RTE   PREPARE SELECT m1."COUNTER" FROM SCLDAPCONFIGM1 m1 ORDER BY
m1."COUNTER" ASC
RTE   sql diff 0.031  total 0.031  call#:1 --> Connecting to Oracle
server 'xxx' as user 'xxx'
RTE   Connected to Oracle Version 10.2.0.1.0
RTE   Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production
RTE   OCI Client settings for language, territory and character set:
AMERICAN_AMERICA.AL32UTF8
RTE   PREPARE SELECT PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS WHERE
PARAMETER IN ( 'NLS_TERRITORY', 'NLS_LANGUAGE', 'NLS_CHARACTERSET' )
RTE   sql diff 0.000  total 0.031  call#:2 --> Prepare Direct SQL::SELECT
PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER IN (
'NLS_TERRITORY', 'NLS_LANGUAGE', 'NLS_CHARACTERSET' )
RTE   sql diff 0.000  total 0.031  call#:3 --> Fetch next Direct
SQL::SELECT PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER
IN ( 'NLS_TERRITORY', 'NLS_LANGUAGE', 'NLS_CHARACTERSET' )
RTE   sql diff 0.000  total 0.031  call#:4 --> Fetch next Direct
SQL::SELECT PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER
IN ( 'NLS_TERRITORY', 'NLS_LANGUAGE', 'NLS_CHARACTERSET' )
RTE   sql diff 0.000  total 0.031  call#:5 --> Fetch next Direct
SQL::SELECT PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER
IN ( 'NLS_TERRITORY', 'NLS_LANGUAGE', 'NLS_CHARACTERSET' )
RTE   sql diff 0.000  total 0.031  call#:6 --> Fetch next Direct
SQL::SELECT PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER
IN ( 'NLS_TERRITORY', 'NLS_LANGUAGE', 'NLS_CHARACTERSET' )
```

sqldebug:3 - Sample output:

```
RTE   sqInitRelation()
RTE   sqmkDoInit(): tuserp->sqmk_nSqlDebug set level=3
RTE   sqmkbuildFieldTab()
RTE   sqmkFindField()
RTE   sqInitialize(): return=0
RTE   sqFetch()
RTE   GetList()
RTE   sqQbe()
RTE   sqmkFetchCommon()
RTE   sqmkbuildFieldTab()
```

```
RTE   sqmkKeyNames()
RTE   sqmkFieldName(): return=0
RTE   sqmkKeyNames(): keyNames=m1.FILENAME
RTE   createSortOrder()
RTE   createSortOrder(): return=0
RTE   sqmkKeyNames()
RTE   sqmkFieldName(): return=0
RTE   sqmkKeyNames(): keyNames=m1.FILENAME ASC
RTE   where()
RTE   whereCondition()
RTE   where(): return=0
RTE   sqmkEquiJoin()
RTE   tableNames()
RTE   tableNames(): return=0
RTE   Connected to Oracle Version 10.2.0.1.0
RTE   OCI Client settings for language, territory and character set:
AMERICAN_AMERICA.WE8MSWIN1252
RTE   sqQbeCallback(): return=0
RTE   sqQbeCallback()
RTE   sqmkbuildFieldTab()
RTE   sqmkFindField()
RTE   sqSetKey()
RTE   sqQbeCallback(): return=0
RTE   sqQbe(): return=0
RTE   GetList(): return=0
RTE   EvalRecord()
RTE   sqmkSelect()
RTE   searchCondition(): return=0
RTE   sqmkSelect()
RTE   sqPrimaryKey()
RTE   sqFetch()
RTE   GetList()
RTE   GetList(): return=9
RTE   EvalRecord()
RTE   sqmkbuildFieldTab()
RTE   sqmkCheckForFld()
RTE   sqmkCheckForFld(): return=TRUE
RTE   sqmkCheckForFld(): return=FALSE
RTE   sqmkFieldName()
RTE   sqmkFindField()
RTE   sqmkFieldName(): fldName=FILENAME
RTE   sqmkFieldName(): return=0
RTE   sqPrimaryKey()
```

debugadhocsql

This parameter enables debugging of the adhocsql functionality, which is used for example when using the multi-level field chooser from the advanced filter in Service Manager search screens. Sample output (excerpt):

```
RTE D adhocOpenCursor: original SQL query: SELECT file.alias01,*,
file.alias02,wdManagerName AS
probsummary_assignment_assignment_name_assignment_wdManagerName FROM
probsummary alias01 LEFT OUTER JOIN assignment alias02 ON  (
file.alias01,assignment = file.alias02,name )  WHERE ( index("MA",
file.alias02,wdManagerName)>0 )
  RTE D Alias provided: alias01
  RTE D JOIN CLAUSE provided: LEFT OUTER JOIN assignment alias02 ON  (
file.alias01,assignment = file.alias02,name )
  RTE D Alias provided: alias02
  RTE D Dump of Original Query
```

```
  RTE D 0000: 66696C65 2E616C69 61733031 2C617373 [file.alias01,ass]
  RTE D 0010: 69676E6D 656E743D 66696C65 2E616C69 [ignment=file.ali]
  RTE D 0020: 61733032 2C6E616D 65                [as02,name       ]
  RTE D Dump of Validated Query
  RTE D 0000: 66696C65 2E616C69 61733031 2C617373 [file.alias01,ass]
  RTE D 0010: 69676E6D 656E743D 66696C65 2E616C69 [ignment=file.ali]
  RTE D 0020: 61733032 2C6E616D 65                [as02,name       ]
  RTE   the original dbdict
  RTE D Field Name                L  Idx T    RC Alias    SQL Fieldname
Type                 DBMERGE Options
  RTE D descriptor                0  1   9     F **NULL** **NULL**
**NULL**                   **NULL**             **NULL**
  RTE D number                    1  1   2     F m1       NUMBER
VARCHAR2(90)               **NULL**             **NULL**
  RTE D number.vj.slo             1  1   2     F **NULL** **NULL**
**NULL**                   **NULL**             **NULL**
  RTE D category                  1  2   2     F m1       CATEGORY
VARCHAR2(50)               **NULL**             **NULL**

  RTE   the NEW dbdict
  RTE D Field Name                L  Idx T    RC Alias    SQL Fieldname
Type                 DBMERGE Options
  RTE D descriptor                0  1   9     F **NULL** **NULL**
**NULL**                   **NULL**             **NULL**
  RTE D number                    1  1   2     F **NULL** **NULL**
**NULL**                   **NULL**             **NULL**
  RTE D number.vj.slo             1  1   2     F **NULL** **NULL**
**NULL**                   **NULL**             **NULL**
  RTE D category                  1  2   2     F **NULL** **NULL**
**NULL**                   **NULL**             **NULL**
  RTE D Dump of Original Query
  RTE D 0000: 696E6465 7828224D 41222C20 66696C65 [index("MA", file]
  RTE D 0010: 2E616C69 61733032 2C77644D 616E6167 [.alias02,wdManag]
  RTE D 0020: 65724E61 6D65293E 30                [erName)>0       ]
  RTE D Dump of Validated Query
  RTE D 0000: 696E6465 7828224D 41222C20 66696C65 [index("MA", file]
  RTE D 0010: 2E616C69 61733032 2C77644D 616E6167 [.alias02,wdManag]
  RTE D 0020: 65724E61 6D65293E 30                [erName)>0       ]
  RTE D Dump of _splitQuery - Query 0 - 0
  RTE D 0000: 74727565                            [true            ]
  RTE D Dump of _splitQuery - Query 0 - 0
  RTE D 0000: 74727565                            [true            ]
  RTE D Dump of _splitQuery - Query 1 - 1
```

ldapstats:1

This parameter is used to debug LDAP connections. Sample output:

```
[OpenLDAP] ldap_search_ext
[OpenLDAP] put_filter: "(&(cn=joe employee)(sn=*))"
[OpenLDAP] put_filter: AND
 [OpenLDAP] put_filter_list "(cn=joe employee)(sn=*)"
[OpenLDAP] put_filter: "(cn=joe employee)"
[OpenLDAP] put_filter: simple
[OpenLDAP] put_simple_filter: "cn=joe employee"
[OpenLDAP] put_filter: "(sn=*)"
[OpenLDAP] put_filter: simple
[OpenLDAP] put_simple_filter: "sn=*"
[OpenLDAP] ldap_send_initial_request
[OpenLDAP] ldap_new_connection
[OpenLDAP] ldap_int_open_connection
```

```
[OpenLDAP] ldap_connect_to_host: TCP hostname:389
[OpenLDAP] ldap_new_socket: 19
[OpenLDAP] ldap_prepare_socket: 19
[OpenLDAP] ldap_connect_to_host: Trying 127.0.0.1:389
[OpenLDAP] ldap_connect_timeout: fd: 19 tm: -1 async: 0
[OpenLDAP] ldap_ndelay_on: 19
[OpenLDAP] ldap_is_sock_ready: 19
[OpenLDAP] ldap_ndelay_off: 19
[OpenLDAP] ldap_open_defconn: successful
[OpenLDAP] ldap_send_server_request
[OpenLDAP] ldap_result msgid 1
[OpenLDAP] ldap_chkResponseList for msgid=1, all=1
[OpenLDAP] ldap_chkResponseList returns NULL
[OpenLDAP] wait4msg (infinite timeout), msgid 1
[OpenLDAP] wait4msg continue, msgid 1, all 1
[OpenLDAP] ** Connections:
[OpenLDAP] * host: hostname  port: 389  (default)
[OpenLDAP]   refcnt: 2  status: Connected
[OpenLDAP]   last used: Wed Feb  9 22:02:54 2005
[OpenLDAP] ** Outstanding Requests:
[OpenLDAP]  * msgid 1,  origid 1, status InProgress
[OpenLDAP]    outstanding referrals 0, parent count 0
[OpenLDAP] ** Response Queue:
[OpenLDAP]    Empty
[OpenLDAP] ldap_chkResponseList for msgid=1, all=1
[OpenLDAP] ldap_chkResponseList returns NULL
[OpenLDAP] ldap_int_select
[OpenLDAP] read1msg: msgid 1, all 1
[OpenLDAP] ldap_read: message type search-entry msgid 1, original id 1
[OpenLDAP] wait4msg continue, msgid 1, all 1
[OpenLDAP] ldap_free_connection: refcnt 1
[OpenLDAP] adding response id 1 type 101:
[OpenLDAP] ldap_parse_result
Ldap query: (&(cn=joe employee)(sn=*)) Base Directory: t=dom return 1
rows.  query time: 0.051000   sort time: 0.000000
[OpenLDAP] ldap_get_dn
LDAP fetch: DN returned for row: cn=joe employee,ou=PRGN,ou=HP,o=DOM
[OpenLDAP] ldap_explode_dn
[OpenLDAP] => ldap_bv2dn(t=dom,0)
[OpenLDAP] <= ldap_bv2dn(t=dom,0)=0
[OpenLDAP] ldap_explode_dn
[OpenLDAP] => ldap_bv2dn(cn=joe employee,ou=PRGN,ou=HP,o=DOM,0)
```

dbmonitorfiles

This parameter is used to monitor updates on files. It is called in the following format

dbmonitorfiles:<*filename1*>,<*filename2*>, ...

with at least one filename passed in. Sample output:

```
DBMONITOR(Update) file:(operator) key:(name=falcon) Application:(login)
Label:(set.last.login.1) User:(falcon)

DBMONITOR(Insert) file:(schedule) key:(schedule.id=2571707)
Application:(apm.problem.change.state) Label:(sched.add) User:(falcon)

DBMONITOR(Update) file:(probsummary) key:(number=IM10001)
Application:(apm.save.problem) Label:(update.go) User:(falcon)
```

```
DBMONITOR(Insert) file:(activity) key:(number=IM10001,negdatestamp=70412
08:21:31,thenumber=001A658) Application:(sm.activity)
Label:(add.activity) User:(falcon)
```

debugdbtypes

This trace is used during SQL to SQL mapping. This parameter is used to document data type conflicts while converting to an RDBMS, to help with troubleshooting unexpected binary type mappings.

dbstats

This parameter starts collecting database statistics after the next restart of the server. To print out the information gathered with this parameter, run **sm –reportdbstats**. Refer to a sample output in the section *sm -reportdbstats* on page 4.

Application debugging

*-RTM:*n *or rtm:*n

RTM stands for *Response Time Monitor*. This trace can be used to check performance data as well as for detailed application tracing.

Valid parameter variations for *n* are:

- 2: Provide performance feedback. Client Response, Server Response, Network Response as well as CPU delta on display of a form. The response times are accumulative since the last form was displayed.
- 3: Same information as on RTM:2 plus detailed RAD application tracing (application name, panel name, CPU delta)
- 4: Same information as RTM:3 plus which RAD function was called how often from the last format response to this one
- 5: Same information as RTM:4, just the count of RAD functions are counted per panel rather than from format to format.

RTM:2 and RTM:3 will produce output at the end of each transaction that indicates response time and memory usage for the transaction. For example:

```
RTE D Total: 2.578 -- RAD: 2.468  JS: 0.110 Database: 0.000  LDAP: 0.000
LoadManager: 0.000 (CPU 2.312)
RTE D Memory: D(1344857) S(1965247) O(1239296)
```

The memory numbers indicate:

- D: Memory delta from the previous transaction. This value could be positive or negative.
- S: Number of bytes of storage allocated by Service Manager. This does not include shared memory or memory that is not directly allocated by Service Manager.
- O: Overhead associated with the internal memory manager.

A complete RTM trace will contain the following types of information:

RTM Level	Sample Output:
3	`RTE D RADTRACE 0 [ 0] login check.version decision CPU( 0 15 )`
3	`RTE D RADTRACE 0 [ 0] login prompt.for.password rio CPU( 0 15 )`
2	`RTE D Response for format: login.prompt.g in application: login,prompt.for.password, option:0`
2	`RTE D Total: 0.000 -- RAD: 0.000 JS: 0.000 Database: 0.000 LDAP: 0.000 LoadManager: 0.000 (CPU 0.015)`
4	`RTE D ... 2 calls made to ';' (1)`
4	`RTE D ... 63 calls made to '=' (2)`

```
4  RTE D ... 7    calls made to ' and ' (4)
4  RTE D ... 8    calls made to 'not ' (5)
4  RTE D ... 3    calls made to '<=' (7)
4  RTE D ... 15   calls made to '=' (8)
4  RTE D ... 5    calls made to '~=' (9)
4  RTE D ... 2    calls made to '>=' (10)
4  RTE D ... 9    calls made to '+' (12)
4  RTE D ... 12   calls made to 'if ' (19)
4  RTE D ... 4    calls made to 'if ' (20)
4  RTE D ... 1    calls made to 'configure' (115)
4  RTE D ... 1    calls made to 'loop' (119)
4  RTE D ... 6    calls made to 'parse' (122)
4  RTE D ... 6    calls made to 'evaluate' (123)
4  RTE D ... 2    calls made to '+=' (128)
4  RTE D ... 2    calls made to 'param2' (134)
4  RTE D ... 4    calls made to ' isin ' (215)
4  RTE D ... 1    calls made to 'scmsg' (238)
2  RTE D - End of Transaction -
3  RTE D RADTRACE      0 [ 0] login
   call.user.login           user.login   CPU(   16     31 )
3  RTE D Calling RAD trigger trigger.operator.check for file operator
3  RTE D RADTRACE      0 [ 0] trigger.operator.check              start
   process      CPU(    0     31 )
3  RTE D RADTRACE      0 [ 0] trigger.operator.check
   init.company              rinit        CPU(    0     31 )
3  RTE D RADTRACE      0 [ 0] trigger.operator.check
   select.company            select       CPU(    0     31 )
3  RTE D RADTRACE      0 [ 0] trigger.operator.check
   process.rules             process      CPU(    0     31 )
3  RTE D RADTRACE      0 [ 0] trigger.operator.check
   do.password.check         decision     CPU(    0     31 )
3  RTE D RADTRACE      0 [ 0] trigger.operator.check
   RADReturn                 Unknown      CPU(    0     31 )
3  RTE D Finished RAD  trigger trigger.operator.check for file operator
3  RTE D Calling RAD trigger trigger.operator.stamp for file operator
3  RTE D RADTRACE      0 [ 0] trigger.operator.stamp              start
   process      CPU(    0     31 )
3  RTE D RADTRACE      0 [ 0] trigger.operator.stamp
   RADReturn                 Unknown      CPU(    0     31 )
3  RTE D Finished RAD  trigger trigger.operator.stamp for file operator
3  RTE D Calling RAD trigger trigger.apm for file operator
```

dbtriggertrace

The dbtriggertrace parameter is used to debug code executed in the Service Manager triggers. You call it as follows:

dbtriggertrace:<*trace level*>,<*filename1*>,<*filename2*>, ...

The trace level defaults to 3. The different trace levels are:

1 or 2: Information given is Calling Trigger, Finished trigger, or trigger not defined on the file.

```
D Calling RAD trigger trigger.operator.check for file operator
D Finished RAD  trigger trigger.operator.check for file operator
D Trigger type 1 not defined for file schedule
D Trigger type 2 not defined for file schedule
```

3 or 4: Same as 1 and 2, plus RAD trace.

```
D Calling RAD trigger trigger.operator.check for file operator
```

```
D RADTRACE     15 [ 0] trigger.operator.check    start          process
CPU(   46     46 )
D RADTRACE     15 [ 0] trigger.operator.check    init.company     rinit
CPU(    0     46 )
D RADTRACE     15 [ 0] trigger.operator.check    select.company   select
CPU(    0     46 )
D RADTRACE     15 [ 0] trigger.operator.check    process.rules    process
CPU(    0     46 )
D RADTRACE     15 [ 0] trigger.operator.check    do.password.check
decision     CPU(    0     46 )
D RADTRACE     15 [ 0] trigger.operator.check    RADReturn        Unknown
CPU(    0     46 )
D Finished RAD  trigger trigger.operator.check for file operator
```

debugjavascript

This debug parameter writes detailed information on calls to JavaScript code in the Service Manager applications. It can be used to find out which fields were used by the JavaScript, or to find in which function within the application an error occurred. Sample output (excerpt):

```
D Created new js context d67690
D Pinning Global object d8af48 using address 1fab257c
D CJsCode: storing JS context 1fab2570
D Compiling javascript function 1FB49650 <internal>
D Pinning Function object d8bf58 using address 1fb76320
D CJsCode: storing JS context 1fab2570
D Calling JS_GetProperty with context d67690 and system
D Calling JS_GetProperty with context d67690 and library
D Calling JS_GetProperty with context d67690 and
triggersContactsOperators
D Created package class version 0 for triggersContactsOperators
D Compiling package triggersContactsOperators...
D CJsCode: storing JS context 1fab2570
D Calling JS_GetProperty with context d67690 and syncContactToOperator
D Executing package triggersContactsOperators to access functions...
D Executing javascript using context d67690
D   -> added function syncOperatorToContact
D Pinning Function object d8c1a0 using address 1fb76d68
D   -> added function syncContactToOperator
D Pinning Function object d8c1d0 using address 1fb76da0
D   -> added function printConRec
D this: 1faddca8d, field:, datum: 130d890d, own: false
D Cleaning - this: 1faae7d8d, field:, datum: 130d890d, own: false
D Executing javascript 'triggersContactsOperators' using context d67690
D this: 1faae3c0d, field:full.name, datum: 130d890d, own: false
D Cleaning - this: 1faae3c0d, field:full.name, datum: 130d890d, own:
false
D this: 1faae3c0d, field:full.name, datum: 130d8b0d, own: false
D Cleaning - this: 1faae3c0d, field:full.name, datum: 130d8b0d, own:
false
D this: 1faae3c0d, field:email, datum: 130d890d, own: false
D Cleaning - this: 1faae3c0d, field:email, datum: 130d890d, own: false
D this: 1faae3c0d, field:contact.name, datum: 130d8b0d, own: false
D Cleaning - this: 1faae3c0d, field:contact.name, datum: 130d8b0d, own:
false
D this: 1faae3c0d, field:fax, datum: 130d890d, own: false
D Cleaning - this: 1faae3c0d, field:fax, datum: 130d890d, own: false
D this: 1faae3c0d, field:fax, datum: 130d8b0d, own: false
D Cleaning - this: 1faae3c0d, field:fax, datum: 130d8b0d, own: false
D Cleaning - this: 1faae868d, field:, datum: 130d850d, own: false
D Called to finalize object d8bec0, part of class system
```

```
D Cleaning - this: 1faddca8d, field:, datum: 130d890d, own: false
D Called to finalize object d8c4d0, part of class library
```

Other debugging parameters

debugevaluator:n

The debugevaluator parameter accepts numbers 1 through 3 as parameters. The higher the number, the more detailed the information.

debugevaluator:1

This trace provides messages detailing the creation, deletion and use of RAD threads by the applications. Sample output (excerpt):

```
  RTE D Callback for evusr
  RTE D ThreadFindByID   0
  RTE D ThreadFindByID - return  0
  RTE D ThreadFindByID   0
  RTE D ThreadFindByID - return  0
  RTE D Switching    0 -> 0
  RTE D ThreadFindByID   0
  RTE D ThreadFindByID - return  0
  RTE D Switching    0 -> 0
  RTE D ThreadFindByID   0
  RTE D ThreadFindByID - return  0
  RTE D Switching    0 ->
```

debugevaluator:2

This trace shows when different RAD threads get suspended, awakened, and switched. An AGstate gets created for every RAD panel, whenever a subroutine is called, or an expression gets evaluated. Sample output (excerpt):

```
RTE D Suspending the RAD 'call' panel
RTE D Suspend = 0x0e3ba600  Next = 0x00000000  AGState = 0x0e31bbc0
RTE D evsuspend for evusr
RTE D Suspend = 0x0e3bb140  Next = 0x0e3ba600  AGState = 0x0e31bbc0
RTE D Create = 0x0e2c0c60  Prev = 0x0e31bbc0  Thread =  0  RAD =
apm.build.inbox.list,shell
RTE D evsuspend for evusr
RTE D Suspend = 0x0e374200  Next = 0x00000000  AGState = 0x0e2c0c60
RTE D Create = 0x0e382270  Prev = 0x0e2c0c60  Thread =  0  RAD =
fillcombo,fillcombo.problem
RTE D Create = 0x0e381a90  Prev = 0x0e382270  Thread =  0  RAD =
fillcombo,select
RTE D Delete = 0x0e381a90  Prev = 0x0e382270  Thread =  0  RAD =
fillcombo
RTE D Create = 0x0e381a90  Prev = 0x0e382270  Thread =  0  RAD =
fillcombo,select
RTE D Delete = 0x0e381a90  Prev = 0x0e382270  Thread =  0  RAD =
fillcombo
RTE D Create = 0x0e381a90  Prev = 0x0e382270  Thread =  0  RAD =
fillcombo,select
RTE D Delete = 0x0e381a90  Prev = 0x0e382270  Thread =  0  RAD =
fillcombo
```

debugsqlmapping:n

This debugsqlmapping:n parameter prints out the mapping information for the file to convert from SQL to SQL into the `sm.log` file. It takes parameters from 1 to 4.

debugsqlmapping:1:

- Prints the dbdict passed into and returned by the sqlmap rtecall and the dbdict representation in the database itself.
- Prints the request and response between the System Definition utility and the server.

debugsqlmapping:2: In addition to the information provided by debugsqlmapping:1 this option also:

- Provides information on why a table is treated as a system table even if there is no sqlsystemtable record. For example, because it was already mapped as a system table before adding the new fields.
- Issues a message why the mapping was internally treated as a REPLACE MAPPING. For example because the dbdict passed into the rtecall was not yet mapped at all.
- Prints progress concerning generation of ALTER TABLE and CREATE TABLE statements, dumps the actual statements, and dumps the dbdict used to generate CREATE INDEX statement
- Issues a message for every hint used, when a sqlhint overrode the default mapping
- Issue debug information concerning table management such as, where will a new field be created, is it necessary to create a new table for the new fields (such as a1, m2, etc.).

debugsqlmapping:3: In addition to the information provided above, this option also:

- Issues messages concerning field and key sizes, and sizes generated as a result of a sqlhint

debugsqlmapping:4: In addition to all the information provided above, this option also

- Prints progress messages on a field by field basis by printing a list of all fields and tables mapped

Sample output (excerpts):

```
Debug level  Debug Message
1            RTE D **** Dbdict record passed into SchemaSQLMap: (replace is
             set to:false)
1            RTE D ======================================================
1            RTE D Field Name       L   Idx  T   RC Alias      SQL Fieldname Type
             DBMERGE       Options
1            RTE D descriptor   0  1   9   F **NULL** **NULL**  **NULL**
             **NULL**    **NULL**    [ 0/ 3]
1            RTE D id           1  1   2   F m1       ID        VARCHAR(250)
             **NULL**    **NULL**    [ 1/ 0]
1            RTE D field01      1  2   2   F **NULL** **NULL**  **NULL**
             **NULL**    **NULL**    [ 1/ 0]
1            RTE D field02      1  3   1   F **NULL** **NULL**  **NULL**
             **NULL**    **NULL**    [ 0/ 0]
1            RTE D ======================================================
1            RTE D DB Type        Alias Table name
1            RTE D ======================================================
1            RTE D Keytype Key field                                  Keyindex
1            RTE D 12      id
1            RTE D **** Mapping file: TEST
2            RTE D Size of unique key is: 252 bytes
2            RTE D adding 0x179530F0 using key M1 to table dictionary
2            RTE D NewTablesInfo called for table TESTM1 and alias=M1
4            RTE D Processing field: descriptor Type: 9
4            RTE D Mapping field: descriptor Type: 9
4            RTE D Skipping field. It is already mapped and we're not
             replacing mapped fields
4            RTE D Processing field: id Type: 2
4            RTE D Mapping field: id Type: 2
4            RTE D Skipping field. It is already mapped and we're not
             replacing mapped fields
```

```
4       RTE D Processing field: field01 Type: 2
4       RTE D Mapping field: field01 Type: 2
2       RTE D FieldMap-4: got 0x179530F0 using key M1 to table
dictionary
4       RTE D Processing field: field02 Type: 1
4       RTE D Mapping field: field02 Type: 1
2       RTE D FieldMap-4: got 0x179530F0 using key M1 to table
dictionary
3       RTE D ** Size of the TESTM1 table is 322 bytes in 3 columns
4       RTE D Table: TESTM1                 Alias: M1  New: Yes
4       RTE D  FIELD01                       VARCHAR(60)
4       RTE D  FIELD02                       FLOAT
2       RTE D GenerateDDL: CREATE for 0x179530F0 from table dictionary
2       RTE D GenerateDDL: dbdict passed into sqmkExportDDL
2       RTE D ============================================================
2       RTE D Field Name       L   Idx  T   RC Alias      SQL Fieldname Type
DBMERGE       Options
2       RTE D descriptor   0  1   9  F  **NULL** **NULL**   **NULL**
**NULL**    **NULL**               [-1/-1]
2       RTE D id           1  1   2  F  M1       ID         VARCHAR(250)
**NULL**    **NULL**               [-1/-1]
2       RTE D field01      1  2   2  F  M1       FIELD01    VARCHAR(60)
**NULL**    **NULL**               [-1/-1]
2       RTE D field02      1  3   1  F  M1       FIELD02    FLOAT
**NULL**    **NULL**               [-1/-1]
2       RTE D============================================================
2       RTE D DB Type        Alias Table name
2       RTE D sqlserver      M1    TESTM1
2       RTE D ============================================================
2       RTE I SQLMap: Alter table statement:
2       CREATE TABLE /*P4[TEST; M1; sqlserver;  ; Tot bytes: 0]*/ TESTM1
        /* Tconstraints */ (
2       /*P4[id; 1; M1; 0]*/ "ID" VARCHAR(250) NULL,
2          /*P4[field01; 2; M1; 0]*/ "FIELD01" VARCHAR(60) NULL,
2          /*P4[field02; 3; M1; 0]*/ "FIELD02" FLOAT NULL
2       )
2       COMMIT
2       CREATE UNIQUE INDEX TESTM14FA3E08B ON TESTM1 ("ID")
2       COMMIT
1       RTE D **** Finished mapping file: TEST
1       RTE D **** Dbdict record returned by SchemaSQLMap:
1       RTE D ============================================================
        [...]
2       RTE D Size of unique key is: 8 bytes
2       RTE D Setting bReplace to TRUE, this table hasn't been mapped yet
2       RTE D NewTable called for file Test with MakeMain=TRUE and
        alias=NULL
2       RTE D adding 0x0C521110 using key m1 to table dictionary
4       RTE D Processing field: descriptor Type: 9
4       RTE D Mapping field: descriptor Type: 9
4       RTE D Processing field: test1 Type: 1
4       RTE D Mapping field: test1 Type: 1
2       RTE D FieldMap-2: got 0x0C521110 using key m1 to table dictionary
4       RTE D Processing field: test2 Type: 2
4       RTE D Mapping field: test2 Type: 2
2       RTE D FieldMap-4: got 0x0C521110 using key m1 to table dictionary
4       RTE D Processing field: test3 Type: 8
4       RTE D Mapping field: test3 Type: 8
2       RTE D FieldMap-4: got 0x0C521110 using key m1 to table dictionary
4       RTE D Mapping field: test3 Type: 9
```

```
4       RTE D Mapping field: test3,test4 Type: 4
4       RTE D Mapping field: test3,test5 Type: 3
3       RTE D ** Size of the TESTM1 table is 88 bytes in 3 columns
1       RTE D **** Finished mapping file: Test
1       RTE D **** Dbdict record returned by SchemaSQLMap:
1       RTE D ========================================================
        [...]
1       RTE D ========================================================
1       RTE D DB Type        Alias Table name
1       RTE D oracle10       m1    TESTM1
1       RTE D oracle10       a1    TESTA1
1       RTE D ========================================================
1       RTE D Keytype Key field                                 Keyindex
1       RTE D 12      test1                                     1
        TESTM1               TESTM1FCEDF55D
1       RTE D --------------------------------------------------------
1       RTE D 0       test2
1       RTE D **** Dbdict record currently in system:
1       RTE D ========================================================
        [...]
1       RTE D **** Mapping file: Test
2       RTE D Size of unique key is: 8 bytes
2       RTE D adding 0x0C5214D0 using key m1 to table dictionary
2       RTE D adding 0x08C73A40 using key a1 to table dictionary
2       RTE D NewTablesInfo called for table TESTM1 and alias=m1
        [...]
4       RTE D Processing field: test6 Type: 8
2       RTE D NewTablesInfo called for table TESTA1 and alias=a1
2       RTE D adding 0x08C73800 using key a1 to table dictionary
4       RTE D Mapping field: test6 Type: 8
3       RTE D FieldMap-1: field TEST6 - key a1 got 0x08C73A40 to table
        dictionary
4       RTE D Skipping field. It is already mapped and we're not
        replacing mapped fields
4       RTE D Processing field: test6 Type: 2
4       RTE D Mapping field: test6 Type: 2
2       RTE D FieldMap-4: got 0x08C73A40 using key a1 to table dictionary
3       RTE D FieldMap-some array table: releasing table TESTA1 alias a1,
        restoring TESTM1-m1
3       RTE D ** Size of the TESTM1 table is 88 bytes in 3 columns
4       RTE D Table: TESTA1              Alias: a1  New: Yes
4       RTE D  TEST6                          VARCHAR2(20)
4       RTE D  TEST61                         VARCHAR2(60)
2       RTE D GenerateDDL: CREATE for 0x08C73A40 from table dictionary
2       RTE D GenerateDDL: ALTER for 0x0C5214D0 from table dictionary
2       RTE D GenerateDDL: dbdict passed into sqmkExportDDL
2       RTE D ========================================================
```

Server debugging

*debugca:*n

This cache trace has two output modes: 1 = minimal, 2 = verbose. It is used to trace what is put into and removed from cache.

debugca:1

Sample output:

```
D Cache add, key(dbdict), type(0)
D Cache add, key(syslanguagedbdict), type(1)
D Cache obs, key(ldaphostscldapconfig), type(1)
```

```
D obsolete(0) type(2) use count(0) interval reference count(1) key(*en-
macro-scmailmanyscmessage)
D ---------------------- Cache Dump End   ----------------------
```

debugca:2

Sample output:

```
D Cache get, key(sctemporalinfo), type(1)
D Cache add, key(sctemporalinfo), type(1)
D Cache put, key(sctemporalinfo), type(1)
D Cache get, key(ldapssldbpathscldapconfig), type(1)
D Cache put, key(ldapssldbpathscldapconfig), type(1)
D Cache obs, key(ldapssldbpathscldapconfig), type(1)
D obsolete(0) type(4) use count(1) interval reference count(2)
key(erdcontract,contractwarranty)
D ---------------------- Cache Dump End   ----------------------
```

*Memdebug:*nnn

The memdebug parameter can be used to trace memory allocation and report memory leaks.

- memdebug:0: Turns memory tracking off (default).
- memdebug:1: Track all memory allocated and freed with minimal information as to who allocated/freed it. If, during thread termination a leak is detected, the immediate caller is dumped out plus a dump of the leaked memory piece.
- memdebug:*n* [where n>1]: Track all memory allocated and freed with a stack trace attached to it and document who allocated it. The value of memdebug determines how much memory will be used for the stack trace; for example memdebug:400 uses 400 bytes. How many stack frames this includes depends on the operating system. On systems running the Windows operating system, that stack trace contains the program counter plus four parameters, each of which is 4 bytes in size. So, 1 stack frame on Windows equals 20 bytes, and memdebug:400 reserves enough space for 20 stack frames. On systems running the AIX® operating system, there are three parameters per stack frame; on systems running the Linux® operating system there are four, and on systems running the Solaris® operating system there are six. Service Manager does not perform stack analysis at all on systems running HP-UX®, so only memdebug:1 should be used on that operating system. For example, when choosing memdebug:400, every allocation is tagged with an extra 400 bytes, meaning when Service Manager allocates 2 bytes, it allocates 402+ bytes with the additional overhead and the stack trace, immensely increasing memory consumption and negatively impacting performance. Do not use this parameter value without being asked to do so explicitly by HP Software Customer Support.
- memdebug:42: Does the same as memdebug:*n* and also turns on tracing of SQL buffers.

debugattachments

This parameter enters detailed attachment-related debugging messages into the `sm.log` file. Client restart is required for this parameter to take effect. Sample output (excerpt):

```
D Created attachment collection 1287da0
D Clearing attachment collection 1287da0
D The current thread has no current record
D Clearing attachment collection 1287da0
D getPendingAttachments returning 0
D After running RAD, the current record is IM10001 and has pending
attachments collection 0 with 0 entries
D Record IM10001 of file probsummary currently has 0 attachments
D Calling buildXMLForAttachments with attachment collection 1f9f0620
D buildXMLForAttachments(): 0 attachments to be processed
D Destroying attachment collection 1f9f0620
RTE D +++++ AttachmentObject: 1fe566a8 11097
90dfe996117a87bb01117a87cb9e0001 application/octet-stream dbdict2.unl
```

```
D Created attachment object 1feff1f0
D Adding attachment 1feff1f0 with href 90dfe996117a87bb01117a87cb9e0001
to collection 1287da0
D Attachment collection size is now 1
D Incoming XML attachment element: name=dbdict2.unl  action=add
href=90dfe996117a87bb01117a87cb9e0001  type=  len=11097
D Execute request received together with 1 attachments
D getPendingAttachments returning 0
D Attachments collection successfully loaded for record IM10001 of file
probsummary
D Record IM10001 of file probsummary currently has 1 attachments
```

debugfileio

This parameter enables Service Manager to write detailed debugging messages on file input/output to the Service Manager log file. File input/output is generated on unloads, loads, exports, and imports. Sample output:

```
D agfile connect
D agfile attempting local connect (file:C:\WINDOWS\TEMP\SCAGW2, mode:
w/0x1/0x309)
D agfile connect successful
D agfile write
D agfile attempting to write 68 bytes
D agfile attempting local write
D agfile attempting to write 1 bytes
D agfile attempting local write
D agfile write
D agfile attempting to write 77 bytes
D agfile attempting local write
D agfile attempting to write 1 bytes
D agfile attempting local write
D agfile disconnect
D agfile sending disconnect to SOAP client (file:c:\temp\test.txt)
D agfile created SOAP document for request filePut
D agfile processed SOAP filePut response
```

debughttp

This debugging parameter logs detailed information about HTTP requests and responses and any SOAP operations. Sample HTTP Log output (excerpt):

```
POST /sc62server/ui  HTTP/1.1
accept: application/fastinfoset, text/html, image/gif, image/jpeg, *;
q=.2, */*; q=.2
authorization: Basic ZmFsY29uOkJDNzBFOURDNEE1MzIzQTk=
soapaction: "getList"
accept-encoding: gzip
content-encoding: gzip
pragma: requestnum="230"
cookie: JSESSIONID=A7617A3943CA9654322878B9D1BA2931; Path=/sc62server;
content-type: application/fastinfoset
content-length: 187
cache-control: no-cache
user-agent: Java/1.4.2 09
host: localhost:13080
connection: keep-alive
à   8ÏSOAP-ENV(http://schemas.xmlsoap.org/soap/envelope/ >Envelope>Header
>————————————————Body<getList<thread3 <formname,contacts.search.g
<————————————————type,listdetail <start‘32 <count¡
HTTP/1.1 200 OK
Keep-Alive: timeout=1200000, max=1000
Connection: Keep-Alive
Pragma: requestnum="230"
```

```
Content-Encoding: gzip
Content-Type: application/fastinfoset;charset=utf-8
Transfer-Encoding: chunked
Date: Fri, 23 Mar 2007 17:07:55 GMT
à   8ÏSOAP-ENV(http://schemas.xmlsoap.org/soap/envelope/Ïxsd-
http://www.w3.org/2001/XMLSchemaÏxsi(http://www.w3.org/2001/XMLSchema-
instance
?Envelope?————————————————Body<getListResponse|modelxcountA32x
————————————————more@1x————————————————namecontactsxquery
Ctruexrecord@0xstart€ <————————————————keys|
contact.namexsctypeEstring ,CM TEST M/F SUPPORT 1  |instancexrecordidCM
TEST M/F SUPPORT 1xuniquequerycontact.name="CM TEST M/F SUPPORT (888)
555-1212 |extension 'x257 |company 'PRGN |operator.id ,M/F SUPPORT 1
|dept.name 'Sales |full.name ,CM TEST M/F SUPPORT 1  FCM TEST M/F SUPPORT
2contact.name="CM TEST M/F SUPPORT 2" ,CM TEST M/F SUPPORT 2,(888) 555-
1212,M/F SUPPORT 2 ,CM TEST M/F SUPPORT 2 FCM TEST M/F SUPPORT
3contact.name="CM TEST M/F SUPPORT 3" E   , ,CM TEST M/F SUPPORT 3 G
,(888) 555-1212 H   I ¡ J ,M/F SUPPORT 3 K ¢ L ,CM TEST M/F SUPPORT 3
FCM TEST PROCUREMENT (888) 555-1212 H   I„  J ,REPLACEMENT 3 K ¢ L ,CM
TEST REPLACEMENT 3  FCM TEST SDUcontact.name="CM TEST SDU" E      , ,CM
TEST SDU G ,(888) 555-1212 H   I 'ACME J ' SDU K ¢ L ,CM TEST SDU  FCM  ,
,CM TEST WAN SUPPORT 1 G ,„  H„  J ,eeditor K„  L ,EDITOR, ED  FEMPLOYEE,
JOEcontact.name="EMPLOYEE, JOE" E   , ,
EMPLOYEE, JOE G ,(317) 455-5476 H ' 505 I ¡ J , Joe Employee K ,Marketing
L ,   Joe Employee  FEMPLOYEE,
descriptionª  N
Btok@2 'OK «  N
tcancel@3 '————————————————Cancel ,
Done Searching  N
Dtsave@4 'Save -  N
Dtfind@8 'Find ®  N
Dtfill@9 'Fill ¯  N
tmodifyC5130 ,
Modify Columns ,
Modify Columns  N
texportC5140 ,Export to Excel ,Export to Excel  N
texportC5150 ,Export to Text File ,Export to Text File  N
tcustomC5170 ,Custom Sort ,Custom Sort  ND32103   <title,"Contact
Information:  BROWN, NICHOLAS <messages
```

Sample sm.log output (excerpt):

```
D Parsing request document: <?xml version="1.0" encoding="utf-
  8"?><execute><thread>1</thread><formname>system.status.list.g</formname
  ><type>detail</type><event>400</event><modelChanges><focus
  cursorLine="1" cursorLineAbs="1"
  readonly="1">var/ss.status.stats</focus></modelChanges></execute>
D Done parsing request document
D Parsing request document: <?xml version="1.0" encoding="utf-
  8"?><execute><thread>1</thread><formname>system.status.list.g</formname
  ><type>detail</type><event>400</event><modelChanges><focus
  cursorLine="1" cursorLineAbs="1"
  readonly="1">var/ss.status.stats</focus></modelChanges></execute>
D Done parsing request document
D Parsing request document: <?xml version="1.0" encoding="utf-
  8"?><execute><thread>1</thread><formname>system.status.list.g</formname
  ><type>detail</type><event>3</event><modelChanges><focus cursorLine="1"
  cursorLineAbs="1"
  readonly="1">var/ss.status.stats</focus></modelChanges></execute>
D Done parsing request document
D Parsing request document: <?xml version="1.0" encoding="utf-
  8"?><getMessages/>
D Done parsing request document
```

```
D Parsing request document: <?xml version="1.0" encoding="utf-
  8"?><getMessages/>
D Done parsing request document
```

debugjni

This parameter provides detailed debugging in the Java® Native Interface® implementation. The Java Native Interface is used to interface the native Service Manager C/C++ code with the Java servlets in Service Manager 6.2 and higher. Sample output:

```
RTE D _____ 0122d2c0 - JNIMemoryAllocator::allocate mem at 01212370, size
= 1977
RTE D _____ 0122d2c0 - JNIMemoryAllocator::findByteBufferObject - at
01212370
RTE D JavaProxy instantiated for class name
com.hp.ov.sm.common.core.XMLAction
RTE D JavaProxy instantiated for class name
com.hp.ov.sm.common.core.AttachmentCollection
RTE D JavaProxy instantiated for class name java.util.LinkedList$ListItr
RTE D _____ JNIMemoryAllocator::getDirectByteBufferAddress - returning
address 01212370
RTE D _____ 0122d2c0 - JNIMemoryAllocator::allocate mem at 0e28b088, size
= 32000
RTE D JavaProxy instantiated for class name
com.hp.ov.sm.common.core.AttachmentCollection
RTE D _____ 0122d2c0 - JNIMemoryAllocator::findByteBufferObject - at
0e28b088
RTE D _____ 0122d2c0 - JNIMemoryAllocator::release - releasing mem at
0e28b088
RTE D _____ 0122d2c0 - JNIMemoryAllocator::releaseAll - releasing mem at
01212370
RTE D _____ 0122d2c0 - JNIMemoryAllocator::allocate mem at 0e2602a8, size
= 397
RTE D _____ 0122d2c0 - JNIMemoryAllocator::findByteBufferObject - at
0e2602a8
RTE D JavaProxy instantiated for class name
com.hp.ov.sm.common.core.XMLAction
RTE D JavaProxy instantiated for class name
com.hp.ov.sm.common.core.AttachmentCollection
RTE D JavaProxy instantiated for class name java.util.LinkedList$ListItr
RTE D _____ JNIMemoryAllocator::getDirectByteBufferAddress - returning
address 0e2602a8
RTE D <?xml version="1.0" encoding="utf-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-
ENV:Header/><SOAP-ENV:Body><start/></SOAP-ENV:Body></SOAP-ENV:Envelope>
RTE D _____ 0122d2c0 - JNIMemoryAllocator::allocate mem at 0e296138, size
= 32000
RTE D JavaProxy instantiated for class name
com.hp.ov.sm.common.core.AttachmentCollection
RTE D _____ 0122d2c0 - JNIMemoryAllocator::findByteBufferObject - at
0e296138
RTE D _____ 0122d2c0 - JNIMemoryAllocator::release - releasing mem at
0e296138
RTE D _____ 0122d2c0 - JNIMemoryAllocator::releaseAll - releasing mem at
0e2602a8
```

logdebuglevel:n

This parameter writes all messages from the servlet into the `sm.log` file. This information should be used to determine if the nodes communicate successfully in a vertical or horizontal scaling implementation.

The parameter accepts the following values:

- 0 = DEBUG: Displays all messages including debug information
- 1 = INFO (default): Displays informational messages, warnings, errors and fatal errors
- 2 = WARNINGS: Displays warnings, errors and fatal errors
- 3 = ERRORS: Displays all errors
- 4 = FATAL ERRORS: Displays fatal errors only

Sample output:

```
16.95.106.191:1993: sending msg #8
sending msg to null (src=16.95.106.191:1993), headers are {NAKACK=[MSG,
seqno=8], UDP=[channel_name=geist8440:62670], VIEW_SYNC=[VIEW_SYNC],
view= [16.95.106.191:1987|2] [16.95.106.191:1987, 16.95.106.191:1990,
16.95.106.191:1993]}
received (mcast) 191 bytes from 16.95.106.191:1994
loadClass(org.jgroups.protocols.VIEW_SYNC$ViewSyncHeader, false)
loadClass(org.jgroups.protocols.VIEW_SYNC$ViewSyncHeader, false)
message is [dst: 228.3.11.191:12346, src: 16.95.106.191:1993 (3 headers),
size = 0 bytes], headers are {NAKACK=[MSG, seqno=8],
VIEW_SYNC=[VIEW_SYNC], view= [16.95.106.191:1987|2] [16.95.106.191:1987,
16.95.106.191:1990, 16.95.106.191:1993],
UDP=[channel_name=geist8440:62670]}
16.95.106.191:1993: received 16.95.106.191:1993#8
.191:1993 (3 headers), size = 0 bytes], headers are {NAKACK=[MSG,
seqno=8], VIEW_SYNC=[VIEW_SYNC], view= [16.95.106.191:1987|2]
[16.95.106.191:1987, 16.95.106.191:1990, 16.95.106.191:1993],
UDP=[channel_name=geist8440:62670]}
received msg from 16.95.106.191:1993 (counts as ack)
16.95.106.191:1990: received 16.95.106.191:1993#8
received (mcast) 191 bytes from 16.95.106.191:1994
```

debuglk:n

This debug parameter writes information on hold times for semaphore locks into the `sm.log` file. It is called with debuglk:*n* for tracing all locks that were held longer than *n* milliseconds. Sample output:

```
RTE W WARNING: lock [8]Cache manager        held for 16 milliseconds by
routine \sc\sc6.2.0.0\src\ca.cpp line 2026
```

debugprocesses

This parameter writes detailed process creation and termination debugging messages into the `sm.log` file. This includes which procedures were called, return codes, records retrieved, encryption messages, and so forth, as shown below. Sample output:

```
RTE D After encryption using SHA512:
RTE D 0000: 24DB92E5 5E5BD443 7C0DCD6F 73BF4FF5 [$...^[.C|..os¿O.]
RTE D 0010: 34A1F230 6B0CAF09 8FF13B14 0D8E202E [4¡.0k.¯    ..;... .]
RTE D 0020: B10C5A07 F84230CB 21256410 9BD6AD99 [±.Z..B0.!%d...-.]
RTE D 0030: 566E0C33 26638EE1 70C7B637 FB3F31E1 [Vn.3&c..p.¶7.?1.]
RTE D Final encryption:
=SH524DB92E55E5BD4437C0DCD6F73BF4FF534A1F2306B0CAF098FF13B140D8E202EB10C5
A07F84230CB212564109BD6AD99566E0C3326638EE170C7B637FB3F31E1512=
RTE D evjscall: Created new js context 122b148
RTE D tzstart: Opened the "info" file successfully
RTE D tzstart: Retrieved the company record successfully
RTE D _getTZRecord: Retrieved timezone record called 'US/Mountain'
RTE D evjscall: Using existing js context 122b148
RTE D Request for work buffer (86564 bytes) exceeded work buffer size
RTE D evjscall: Using existing js context 122b148
```

debugnode

This parameter does not produce any additional output. When the parameter is added to a servlet container / node in a scaled environment, it prevents the Load Balancer from sending or redirecting requests to that node. It allows that node to be "part of the cluster," but will only receive direct connections. This will help with debugging only a subset of users that work in the load balanced environment, by putting debug parameters only onto the node that is receiving a controlled set of direct connections.

*debugrs:*n

This parameter traces how long resource locks are being held. It is called as debugrs:*n* where *n* is the number of milliseconds a resource lock can be held until a message is written to the `sm.log` file. Sample output:

```
RTE D Held resource (loginLock-falcon) for 16 milliseconds
RTE D Held resource (scirexpert) for 16 milliseconds
RTE D Held resource (scirexpert) for 16 milliseconds
RTE D Held resource (ir.probsummary) for 141 milliseconds
RTE D Held resource (techterms) for 16 milliseconds
RTE D Held resource (scirexpert) for 16 milliseconds
RTE D Held resource (scirexpert) for 15 milliseconds
RTE D Held resource (ir.probsummary) for 94 milliseconds
```

debugscauto

This parameter provides detailed debugging messages for SCAuto connections in the `sm.log` file.

debugstartup

This parameter writes detailed startup debugging messages into the `sm.log` file, such as information about getting system information, creating and attaching shared memory, and so forth. Sample output:

```
RTE D sysinfo_set: entered with type 3
RTE D GetMyHostByName: using host name hostname
RTE D GetMyHostByName: returning ip address 127.0.0.1
RTE D inhook: entered
RTE D insyslock: entered
RTE D insyslock: lock string is hostname.62670
RTE D systart entered
RTE D systart: calling inhook
RTE D sysinfo_set: entered with type 3
RTE D GetMyHostByName: using host name hostname
RTE D GetMyHostByName: returning ip address 127.0.0.1
RTE D inhook: entered
RTE D insyslock: entered
RTE D insyslock: lock string is hostname.62670
RTE D sm_hook: About to call shmget to attach to shared memory
RTE D sm_hook: Successfully attach to shared memory
RTE D ---------- shmat call MapViewOfFileEx() with shmaddr = 3020000
RTE D systart: inhook returned 0
RTE D systart returning 0
RTE D inunhook: entered
```

debugshutdown

This parameter gives detailed shutdown debugging messages. Sample output:

```
RTE D inshut() - Calling usshut..
RTE I User  has requested a system shutdown
RTE D User block found for process 5516, thread -1
RTE D pid 5516 tid -1 is prevented, stop scheduled
RTE D User block found for process 4644, thread -1
RTE D kill function entered with signal 15 for pid 4644
```

```
RTE D stop succeeded for pid 4644 tid -1
RTE D User block found for process 4392, thread -1
RTE D OpenProcess for pid 4392 failed. Last error = 87
RTE D User block found for process 4392, but process no longer exists
RTE D inunhook: entered
RTE D User block found for process 5516, thread -1
RTE D pid 5516 tid -1 is prevented, stop scheduled
RTE D User block found for process 4644, thread -1
RTE D kill function entered with signal 9 for pid 4644
RTE D Calling TerminateProcess for process 4644
RTE D stop succeeded for pid 4644 tid -1
RTE I Waiting for 5 users to terminate
JRTE I TermnateStatus is set to true, cleaning up all servlet threads.
Stopping Coyote HTTP/1.1 on http-13080
RTE I Process termination in progress
RTE D inunhook: entered
RTE D User block found for process 4644, thread -1
RTE D OpenProcess for pid 4644 failed. Last error = 87
RTE D User block found for process 4644, but process no longer exists
RTE D inshut returned: 0
RTE D Starting interm()...
RTE D Starting inRemoveScdbSystem()...
RTE D Finished inRemoveScdbSystem()...
RTE D Calling error_exit..
```

msglog

This parameter places all messages that were sent to the client into the `sm.log` as well.

Note: The msglog parameter works only with the RPCReadOnly listener.

Tuning Service Manager

Service Manager is an enterprise-level application that is used by very busy helpdesks where response time is very important. This section provides recommendation on how to best tune Service Manager.

Tuning Service Manager Server

Service Manager is designed to run 24 x 7 x 365 supporting a few hundred to thousands of concurrent users. There are two ways in which to scale a system: vertical scaling, a huge system capable of supporting all users, or horizontal scaling, using several smaller systems to each support a subset of users. There are advantages and disadvantages to each of the methods.

In order for a group of Service Manager Server processes to work together, they need to communicate information on:

1. Resource Locking
2. License management
3. Cache invalidation
4. Session status
5. Load balancing
6. IR processing

Terminology:

- Group: a group of Service Manager server processes running on one or more hosts and connecting to one database to serve Service Manager client sessions or Web Services sessions.
- Node: a node is a Service Manager server process within a group.

- Service Manager servlet process: Service Manager Server process that serves Service Manager clients (Windows or Web) as well as Web Services requests.
- Service Manager background process: Background processes that wakes up periodically to execute a particular RAD application or RTE routine such as `sm -que:ir`.
- Service Manager transient process: Service Manager Server process that executes a RAD application or RTE routine only once, not periodically such as `sm -reportlbstatus`.

Vertical Scaling

With vertical scaling, multiple Service Manager Server processes run on a single machine. They support a number of concurrent users while one Service Manager Load Balancer process runs to redirect a user to a particular Service Manager Server process. Clients first make a connection request to the Service Manager Load Balancer who then forwards the request to the next available Service Manager Server process to establish a user session.

Communications regarding load balancing happen through JGroups. JGroups uses multicasting to create and maintain consistent information among all the nodes within the group.

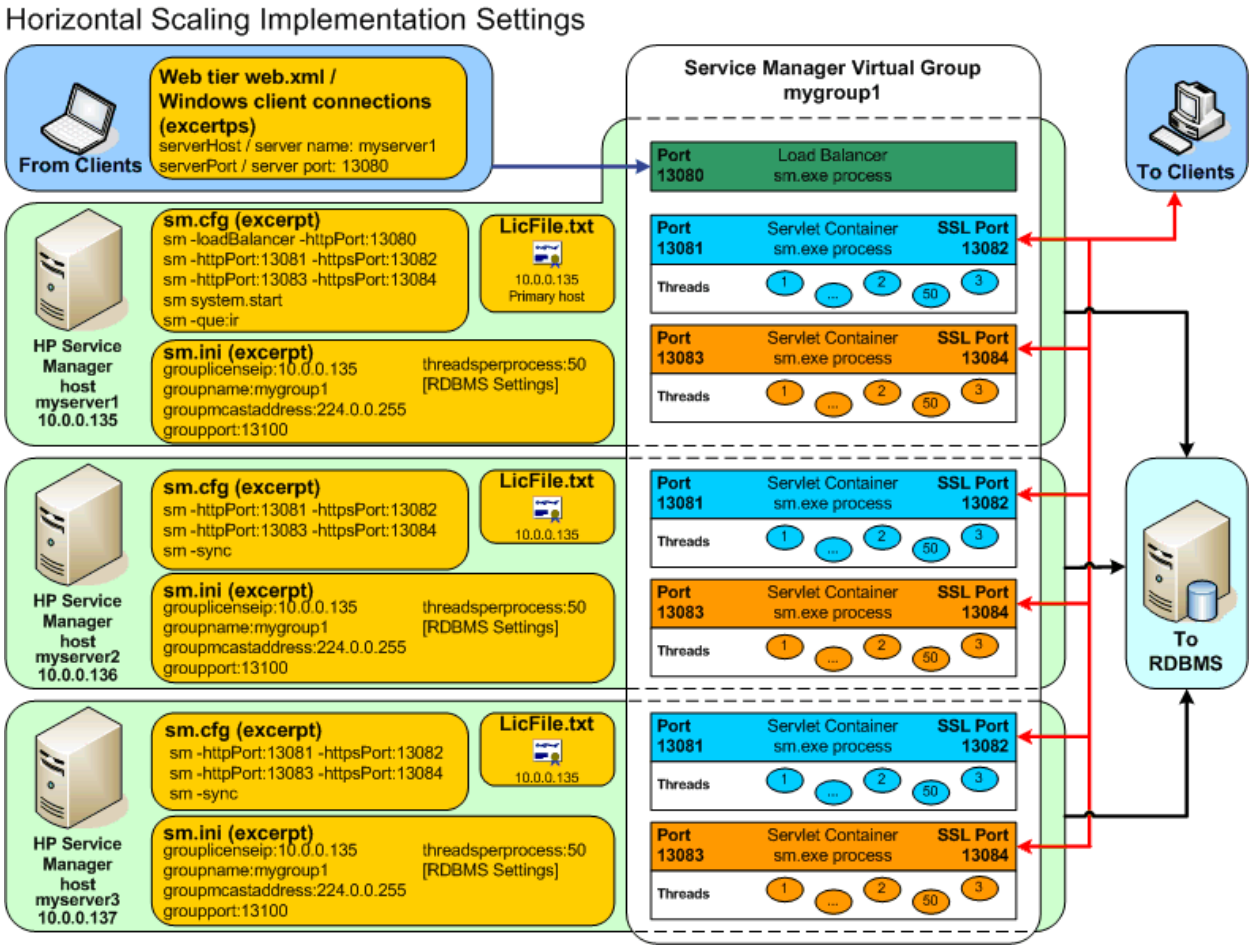
To reduce communication overhead, the Session status, Resource Locking, License management and cache invalidation information are communicated through shared memory, since all Service Manager processes are located on the same host.

Horizontal Scaling

The following diagram illustrates a typical horizontal scaling environment. The `groupname` parameter in the sm.ini indicates that it is a horizontal scaling configuration. Without `groupname`, Service Manager Server will start a vertical scaling environment and will not communicate with other Service Manager services on other hosts.

In the horizontal scaling setup, the host myserver1 runs the Service Manager Load Balancer, several Service Manager Server processes, the Service Manager background processes and `sm -que:ir`. The `sm -que:ir` process is required since asynchronous IR is forced in the horizontal environment. The hosts myserver2 and myserver3 each contain several Service Manager Server processes and the `sm -sync` process. When a client connection is made to the Service Manager Load Balancer on myserver1, the Load Balancer sends a redirect message containing an available server back to the client. The client then makes another request to the server specified in the redirect message.

Communications between all processes on the same host go through the host's shared memory. The communications between processes on different hosts go through JGroups.



Mechanisms behind the scaling environment

- Shared Memory

  Shared memory is created as soon as the first Service Manager Server process on the host starts. Each host in the group has its own shared memory.

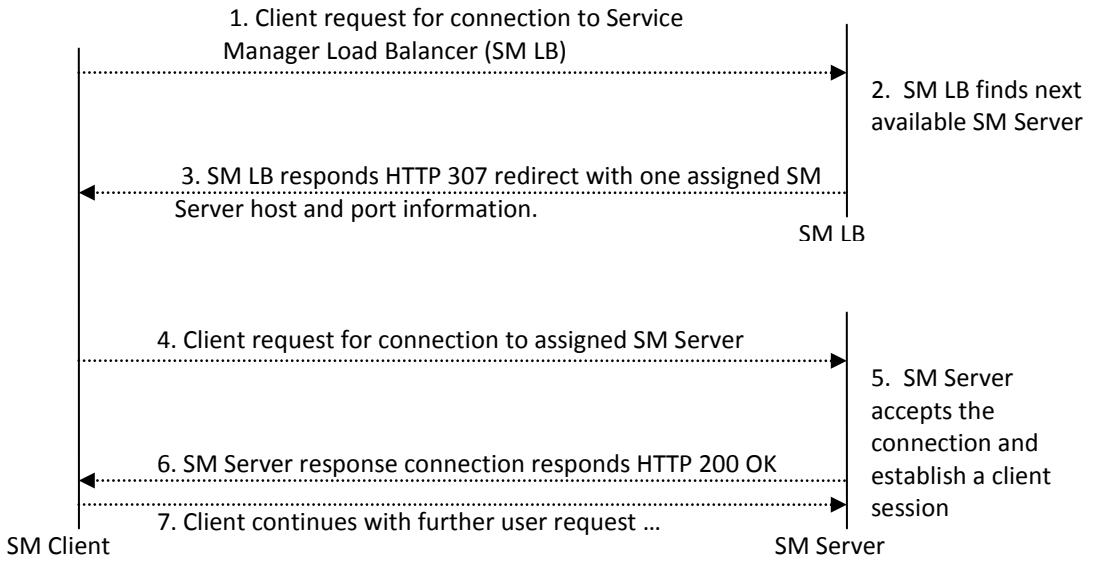
  In a vertical scaling environment, shared memory contains information on resource locking, license information, cache, session status, and IR processing that needs to be shared between Service Manager Server processes on the same host. In a horizontal scaling environment, resource locking, license information, and session information are stored in the process memory and are communicated through JGroups. Only data cache and IR cache are stored in the shared memory in this environment.

- Load Balancer

  The Service Manager Load Balancer is a special instance of the Service Manager server whose sole purpose is to provide a single entry point for all client connections to a farm of Service Manager servers. The Service Manager Load Balancer running in a vertical or horizontal scaling group automatically detects all Service Manager Server processes that are currently in the group. It is able to categorize all detected Service Manager Server processes as servlet process, background process, or transient process. It also maintains information of the maximun and current capacity of each servlet process.

  – Implementation

    The Service Manager Load Balancer is implemented as a servlet filter. This servlet filter upon receiving a HTTP/1.1 client request will return a HTTP 307 redirect with the HTTP Location header set to a URL of one of Service Manager Application servers back to the client that made the request. The client in turn uses the URL set in the Location header and connected directly to the Application server, the handling of redirect is usually transparent. The following diagram illustrates the process of how the Service Manager Load Balancer handles a client request.



- Algorithm

  As described in the previous section, SM uses JGroups multicast technology to communicate among a group of servers. Load Balancer maintains a list of available server nodes based on the group view (list of nodes identified by JGroups). The view changes when a node joins or leaves the group. The load balancer then either adds or removes the respective node from its list. When the load balancer sees a new node it queries the node to find out its capacity.

  Load balancer uses a simple round-robin algorithm when choosing a server to handle a user session request. After forwarding a request the load balancer decrements the available capacity of the node by one. When a node reaches its capacity, load balancer removes it from its pool of nodes with capacity. Nodes in the group notify the load balancer of their current availability when a session ends. The load balancer then increments the available capacity of that node if it is in the pool of nodes with some capacity otherwise the node gets added to the pool.

  When total available capacity within the group drops below 25, load balancer requests each node in the group to reply with the current status and the list of available nodes is updated. The position of each node on the list may change whenever the list is updated. Since the list is changing quite often, the node that the load balancer directs the next Service Manager Client session to may seem unpredictable.

- Current limitations
  - There can only be one Service Manager Load Balancer in a group. This can be considered as a single point of failure, affecting only new connections. Once the load balancer is restarted the system can continue functioning. Later in this document we will show how to overcome this single point of failure with a high availability configuration.
  - The HTTP 307 redirect is not fully compliant with the spec which can affect Web Services integrations through SM Load Balancer (e.g. MS BizTalk). The workaround is to connect directly to one of the Service Manager Application servers.
  - Since the server nodes notify SM Load Balancer of its capacity only when a session ends, it will not be aware of any connections made directly to a server.
  - Web Services through SM Load Balancer is not possible when SSL mode is enabled on the server.

– Web Service through SM Load Balancer is not possible for Web Services clients that can't handle redirect.

Session status

In a vertical scaling environment, the user session information is stored in shared memory. Information about all currently running processes and users can be directly retrieved from shared memory. In a horizontal scaling environment, the user session information is kept in memory on each of the Service Manager Server processes. Using JGroups communication, the session information on each of the hosts is gathered and made available to the system status application.

Resource locking

Locking provides a tool for protecting data integrity. In Service Manager, locks are logical locks, meaning they do not actually lock physical database records, even though the lock name such as "probsummary:IM1001" hint to a physical record lock. A lock prevents concurrent Service Manager applications or RAD threads to manipulate the same record which is protected by a lock.

The lock information containing which thread is currently holding the lock is stored in shared memory so that all Service Manager Server processes on the same host can get consistent information within the vertical scaling configuration. In a horizontally scaled environment, this information is stored in the Service Manager Server process memory and needs to be synchronized across all SM Server processes running on all Service Manager hosts at any given point in time, which imposes an overhead. Therefore, it is best practice to only use locking when it is necessary and to hold the lock for as little time as possible.

License management

The Service Manager License is locked to an IP address. License information is stored in shared memory in a vertically scaled system and stored in the Service Manager Server process memory in a horizontally scaled system.

In a horizontally scaled environment, the host whose IP address the Service Manager License is bound to is referred to as the primary host. Others in a group are referred to as secondary hosts. The `grouplicenseip:<primary host IP>` parameter should be included in the sm.ini file on all secondary hosts to validate the license with the primary host and the LicFile.txt file, which is retrieved from HP Webware, should be copied on all of the secondary hosts.

In order to establish the horizontal scaling group, the first node of the group needs to be started on the primary host. Once the group is established on the primary host, the Service Manager Server processes on the secondary hosts can be started to join the group. Once the whole group is established, the primary host can be brought down for maintenance and rejoin the rest of the running group when it is restarted. As long as a member of the SM Server process is running in the group, processes on a secondary host with the same `grouplicenseip:<primary host IP>` can join the group and it is not required for the primary host to be running. Another secondary host can be started even if the primary host is down for any reason. However, when the group is down, i.e. there is no node running in the group, the group has to be re-established from the primary host first.

Cache invalidation

SM server creates one shared memory per host to store cache data for faster access to frequently used records. When a record is updated, cache data in the shared memory needs to be refreshed. In the case of vertical scaling, since all nodes reside on one host with only one shared memory, all SM Server processes will be reading the up to date data as soon as the cached record is refreshed by any SM Server process. In a horizontal scaling environment, each host in a group has its own shared memory and any one record could be cached on more than one host. When this cached record is updated on any host in the group, the SM Server process has to notify all other hosts in the group to invalidate that cached record and refresh the record from the database.

IR processing in scaled environments

Service Manager IR is a free text search engine with its own index to do fast retrieval. IR index information is stored in shared memory. Whenever a record containing an IR field is updated, IR index information in shared memory also gets updated. This counts toward user response time when updating a record. We can have better response time by using Asynchronous IR, with the tradeoff that the search might not reflect the new changes for a short period of time. Asynchronous IR inserts a record in the `irqueue` table whenever a record with an IR field has been modified. Then the `sm –que:ir` background process processes the records in the `irqueue` table to update the IR index with the modified information.

In a horizontally scaled environment, updating the IR index means updating all shared memory on all hosts in the group. This causes a performance impact when using synchronous IR. Therefore, asynchronous IR is forced in horizontal environment. For IR to run successfully, start the `sm –que:ir` background process once on one host within the group.

Web Services in scaled environments

In a horizontal scaling or vertical scaling environment, it is recommended that the incoming Web Services requests to the Service Manager server do not connect through the SM Load Balancer. Instead, dedicate one or more SM Server processes to serve Web Services requests by adding the `debugnode` parameter to the process you wish to be dedicated to serve Web Services requests. The `debugnode` parameter tells the SM Load Balancer in the group not to forward any request to this node. For example: in vertical scaling, you can add "`–debugnode`" to one SM Server process in the sm.cfg file as follows. Then, direct all Web Services requests to port 13083 for http connection and port 13445 for https connections.

```
---sm.cfg---
sm -loadBalancer -httpPort:13080
sm -httpPort:13081 -httpsPort:13443
sm -httpPort:13082 -httpsPort:13444
sm -httpPort:13083 -httpsPort:13445 -debugnode
sm system.start
---------------
```

Application tailoring considerations in the clustering environment

use `batch size` in sequential number and counter file

The counter and number tables are generally used for filling a unique number into a key field when creating a new record. For instance, the incident management number record holds the last number for Incident Management tickets. These numbers have the prefix "IM" and the next Incident ticket number can use the currently stored number, incrementing it by 1. During incident ticket creation, the system has to read the number record and update it with the incremented last number which poses a bottleneck when trying to create a large volume of incident tickets.

If in your environment it is only required to have a unique number instead of a unique and continuous number, you can – in both vertical and horizontal scaling environments - relieve this bottleneck by setting the batch size field in the associated number or counter records. This allows the SM Server process to pre-fetch a specified amount of numbers or counters for later use. These pre-fetched numbers or counters are stored in shared memory. The SM Server processes on the same host can then assign these pre-fetched numbers to the newly created record until they are all used and then fetch the next set of numbers or counters.

Using the incident management number record as an example, if the batch size field is set to 25, and the last number is stored as 1000, the SM Server process on a host in the group will pre-fetch 25 numbers, and set last number field to be 1025. The SM Server process on another host in the group will then pre-fetch another 25 numbers and set the last number field to be 1050. This way, when users logged into the first SM host create Incident Management tickets, they use numbers IM1001 through IM1025. When the numbers in the shared memory on a host have been used up, the SM Server process then fetches another batch of numbers.

This improves system performance dramatically, but when the system shuts down, numbers that have been fetched but not used will not be restored back to the number or counter table. Due to this, when the batch size is set the number value in the associated field will not be continuous. In a horizontally scaled environment, this also means that lower ticket numbers do not imply that the tickets are created earlier than the higher ticket numbers. In the example above, IM1026 ticket might be created before IM1020.

The batch size should be set close to the value that is going to be used in a day, i.e. if batch size is set to 1000 and only 5 records are created for the table before the systems get restarted, 995 numbers are going to be lost. Because there is an upper limit to each number, which is a long data type, you do not want to waste numbers.

background processes (Anubis)

Anubis is a RAD application that queries for currently running processes and restarts the background processes that are not in the running processes list, but are in the system startup record. Typically you run this application via the anubis background scheduler periodically to automatically start background processes. However, in the horizontal or vertical scaling mode, this application requests every SM process in the group to provide its currently running sessions and threads, which creates a large communication overhead if anubis is running too often. It is a good practice to set the anubis background scheduler to run less often than every 30 min.

global lists

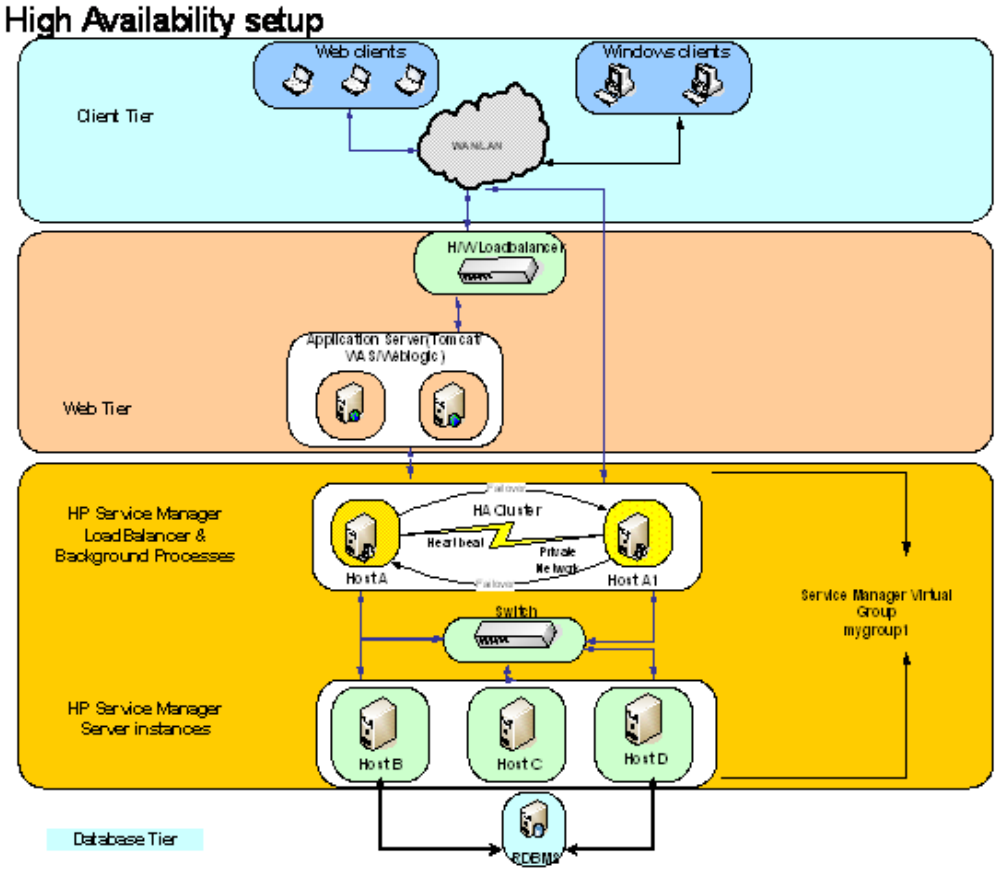
A global list that contains a huge number of items will have performance impact for the following reasons. Since global lists are often used in drop downs, big global lists mean it takes a long time for the SM Server process to build the form, a long time for the SM Clients to process the form, and SOAP messages between the SM server and client are huge.

Service Manager has a mechanism to automatically rebuild global lists by setting the global list to expire whenever a record is modified that contains a field which is part of the global list. When the lister background process wakes up every minute to rebuild expired global lists, it locks up the global list to avoid data corruption while rebuilding. In a horizontal or vertical scaling environment, if rebuilding the global list takes a long time, no other threads or processes can update the global list for that time. This means if a user modifies a record that affects a global list, the user thread has to wait for that global list to be unlocked to set the global list as expired. This causes a long response time when modifying those records.

Review the global lists on your system. If a global list gets too big, remove that global list and replace the drop down by a fill field on the form. Fill goes through a QBE list while a drop down list uses a global list.

High availability setup for horizontal scaling

The following diagram shows a high availability setup.



In this setup, the Service Manger Windows client connects through a LAN and directly talks to the Service Manager Load Balancer. Service Manager Web Client users use browsers to connect to a Hardware Load Balancer, which then redirects the user to one of the application servers that host the Service Manager Web Tier. The Web Tier then connects to one of Service Manager Server processes through the Service Manager Load Balancer. The following section takes a closer look at the Service Manger Server configuration on each of the systems in the above setup.

Host A & Host A1

Host A and Host A1 are part of a High Availability cluster. Host A and A1 will have an identical configuration, with normally only Host A being active. The load balancer and all Service Manager background processes including the `sm -que:ir` process are setup on this system. If Host A dies for any reason, Host A1 becomes active and in the process of becoming active starts the load balancer and all background processes. This setup eliminates the load balancer being the single point of failure. Also, having all of the background process in the HA systems solves the problem of losing the background processes when the host running the background process dies. The individual background processes are guaranteed to get restarted on failure by the Service Manager anubis background process.

Switch

One of the requirements for horizontal scaling is that all servers are connected to the same multicast-enabled switch for a minimum number of hops. The minimum number of hops is of utmost importance for the maintenance and performance of the group.

Server A, B, C, D

Servers A through D will all run the Service Manager servers. These servers need not be of the same capacity. At this time, we only support servers running the same Operating System. Each of the servers needs to run the sync background process to maintain the integrity of shared memory.

Scaling Environment Administration/Troubleshooting

It is not trivial to setup a horizontally scaled system. You need to understand the impact of all the parameters needed to successfully run in this mode. Here are some common issues in setting up a horizontal scaling environment and how to troubleshoot them.

Servers from different hosts do not communicate with each other

Run the command below from the <sminstall>\RUN folder

```
jre\bin\java -classpath lib\jgroups-all-2.5.0.jar;\lib\commons-logging-
1.1.jar;lib\log4j-1.2.14.jar org.jgroups.tests.Probe -timeout 30 -query
jmx
```

This will help find all the cluster members and their configurations (groupaddress, groupport).

Server not starting

Run the command below from one of the host from the <sminstall>\RUN folder

```
jre\bin\java -classpath lib\jgroups-all-2.5.0.jar;\lib\commons-logging-
1.1.jar;lib\log4j-1.2.14.jar org.jgroups.tests.McastReceiverTest -
mcast_addr 224.10.10.10 -port 5555
```

The above command will start and wait for JGroups messages. Now from another host run the command below from the <sminstall>\RUN folder

```
jre\bin\java -classpath lib\jgroups-all-2.5.0.jar;\lib\commons-logging-
1.1.jar;lib\log4j-1.2.14.jar org.jgroups.tests.McastSenderTest -
mcast_addr 224.10.10.10 -port 5555
```

After running the command, type in a message to be sent and press enter. If the message is received on the first host running the recv test, you have a valid configuration.

If the message was not received and you are running on windows, try the following command before repeating the above steps again.

```
route add -net 224.0.0.0 netmask 224.0.0.0 dev lo
```

Having multiple Network Interface Cards (NICs)

If you have multiple network cards on the server, follow the required steps outlined below:

1. Specify the groupbindaddress parameter in the sm.ini. This will be the IP address of the NIC on which all of the cluster communication will occur. This value will be different on each host with multiple NICs. This helps to improve performance and the cluster communication will occur only on the bound address. If you see this error message: Multiple ipaddress found, Failed to lock the database with the clustername and subnet address. Provide an ipaddress to use in groupbindaddress parameter, or Provide subnetaddress to use in groupsubnetaddress parameter, you will need to specify the groupbindaddress.
2. Generally, you will only need the groupbindaddress parameter if the host has multiple NICs and the groupsubnetaddress parameter is not needed. However, if you have put the groupbindaddress parameter into the sm.ini and still see the error message Failed opening a temporary socket, to retrieve subnet mask or Error occurred while retrievieng subnet mask, you will need to specify the groupsubnetaddress parameter in the sm.ini file. This will help lock the database of this cluster. The value for this parameter needs to be the same on all the hosts.

SM Administration Commands

The commands below should be executed from the <sminstall>\RUN folder.

To report the current state of the SM Load Balancer:

```
sm -reportlbstatus
```

To report locks owned by each process in the group and user sessions which are waiting for locks:

```
sm -reportlocks
```

To report the state of the group (n) number of times repeating every (m) seconds:

(Note that if (n) is set to 0, it reports the state of the group every (m) seconds until the Service Manager server is shutdown.)

```
sm -reportgroup:m,n
sm -reportgroup:m,0
```

How Do I quiesce servlets?

To quiesce the entire group, run

```
sm -quiesce:2 -group
```

To quiesce all servlets on a host, run

```
sm -quiesce:2 -host:<hostname>
```

If you have specified the `groupbindaddress` parameter in the sm.ini, you should use the bind address for the hostname in the above command.

How do I shutdown servlets?

To shutdown the entire group, run:

```
sm -shutdown -group
```

To shutdown all servers on a host, run:

```
sm -shutdown -host:<hostname or IP>
```

Note: The host name and the IP value should be the one reported through reportlbstaus.

To shutdown one SM Server process in a group, run:

```
sm -shutdown -host:<hostname or IP> -pid:<process id>
```

Additional logging for debug in Horizontal environment:

To report all communication from one SM Server process to all others in a group:

```
sm -httpPort:13080 -
log4jDebug:com.hp.ov.sm.common.resource.TaskDispatcher
```

Report all SM Load Balancer info:

```
sm -loadbalancer -httpPort:13080 -
log4jDebug:com.hp.ov.sm.common.cluster.LoadBalancer
```

Reports all lock requests and the amount of time each lock is held by a SM Server process or SM user session. This parameter will log large amounts of information, so make sure only a single SM Server process has this parameter enabled. You can place this parameter in the SM Server command line such as:

```
sm -httpPort:13080 -httpsPort:13443 -debugrs:1
```

Web Services - Service Manager as Web Services provider

For each http post request, Service Manager Server creates or finds an existing http session to serve the request. A session in Service Manager is identified by userID and JSESSIONID, which are stored in the http Authentication header and Cookie header respectively.

For the GUI clients (Service Manager web and windows client), a session is created when the user logs in and removed after the user logs out or the login fails.

For Web Services clients, a session is created when the Web Services SOAP request is received. It is removed after the corresponding SOAP response has been sent or login fails.

Here are some tips to help Service Manager performance in serving Web Services:

1. Do not use same user name for all Web Services sessions

   Service Manager must go through the login process for each session creation. When a number of requests try to log in with the same user name at the same time, the operator record becomes the bottleneck. This is due to the fact that Service Manager records the login history and other information in the operator record for each login process. If all requests use the same operator, modifying this operator record has to be synchronized.
2. For batch Web Services requests, use session persistence

   In Service Manager, a session is identified by user id and JSESSIONID in the http cookie. Traditionally, Service Manager creates one session to serve each Web Services request and destroys the session when the associated response has been sent. Because session creation is an expansive operation, it is possible to have Service Manager persist sessions for Web Services requests to avoid frequent session creation and destroy overhead.

   One use case of Web Services session persistence is when a large amount of Incident Management tickets in a remote system need to be integrated into Service Manager. Instead of doing one CreateIncidentRequest/CreateIncidentResponse per session, Service Manager can keep the session for a predefined amount of time to serve multiple sequential Web Services requests. The Service Manager parameter to specify the amount of time to keep a Web Services session alive is `webservices_sessiontimeout`. Follow the steps below to implement this parameter.
3. Cache queries against the extaccess table

   Add the following parameter to the sm.ini: dbcachequery:extaccess. This parameter caches query results and thus improves query performance.

How to setup session persistence

1. In theService Manager server sm.ini, specify webservices_sessiontimeout in seconds for the amount of idle time before the Service Manager server cleans up the Web Services session. This value should be set according to system resources available to the web service to avoid exhausting Service Manager available sessions. For example, if 50 sessions are available for the web service port, and 10 requests come in per minute, set the sessiontimeout parameter to no more than 5 minutes.
2. On the Web Services client side, specify the following http header in the Web Services requests:

```
Connection: Keep-Alive
```

3. In the response header from the Service Manager server, capture the cookie information.

```
HTTP/1.1 200
Set-Cookie: JSESSIONID=3ECB02AF2AB26987EAF75FC07DCDFF6A; Path=/SM
Keep-Alive: timeout=1200000, max=1000
Connection: Keep-Alive
Content-Type: text/xml;charset=utf-8
Content-Length: 1133
Date: Fri, 07 Nov 2008 19:34:35 GMT
```

4. In the subsequent requests, add the cookie information that is captured in step 3 to the request http header and use the same user information as in step 2.

```
cookie: JSESSIONID=3ECB02AF2AB26987EAF75FC07DCDFF6A; Path=/SM
```

   In the sm.log, you should see `4356(  2396) 11/07/2008 11:54:18  RTE D User falcon is already logged in for this process - skipping login processing.` This means the Service Manager Server has found the session and the user authentication information identified by the cookie and is using it to serve the request.
5. If the Web Services client realizes the request is going to be the last request of the session and wishes to close the session to avoid session lingering, send a Web Services request with the following header:

```
Connection: closed
```

SM Server will clean up the session after serving this request. You should see "`User falcon has logged out and freed a Named license (0 out of a maximum 25)`" messages in the sm.log.

IR Tuning for Performance

IR index is a very complex data structure. To improve performance on IR regen, index updates, or IR query the following steps can be taken:

- Purge/Archive

  Archive old records in tables that have an IR key and purge them afterwards. Archiving can mean to move them into a separate SM system where they are still accessible. Complexity is reduced by indexing fewer records.

- Add stop words

  Adding new stop words reduces the number of terms that need to be indexed. First run vrir against the file in question to determine which terms are used most often. Very often year and month numbers are found at the top of the list (e.g. 01, 02, 03, 2007, 2008). It is recommended to put all terms into the stop words file that occur in more than 10,000 documents.

- Review the key definition

  Review the IR key defined for each file. The more fields are indexed the more terms will be indexed, adding to the complexity. The original purpose of IR was to be able to quickly find a solution for a certain issue. To find the solution for an issue, only the description should be searched, not the solution or updates, since the description should match between two similar issues.

- Allow IR to use more shared memory

  More space in shared memory dedicated to IR can improve IR performance. The `ir_max_shared` parameter is used for allocating dedicated space in shared memory for IR and the `shared_memory` parameter specifies the size all SM Server processes should allocate for total shared memory. Ideally, the size of `ir_max_shared` should be big enough to contain all IR indexes. Most Service Manager systems have several large IR index files such as `ir.probsummary`, `ir.incidents` or `ir.cm3r`. The `ir_max_shared` parameter can be set to the sum of the size of these large indexes. You can find out the size of each index by running `vrir` under `sm -util`. The `shared_memory` parameter value should be set to the sum of `ir_max_shared` and recommended shared memory base size.

  For example, if you have two hosts running horizontally scaled and `ir.probsummary` is 90% of all IR indexes and you have two SM Server processes running on each host to serve 100 users per host. First find out the size of the `probsummary` IR index by running `vrir` in `sm -util`:

```
----vrir report in sm.log(pid, thread id and time stamp are omitted)---
RTE I ***       512 bytes allocated for IR header
RTE I ***    320072 bytes allocated for hashtables
RTE I ***      5764/     5633 bytes allocated/used by DOCs
RTE I ***     10708/    10360 bytes allocated/used by TERMs
RTE I ***     16694/    12529 bytes allocated/used by DOCLISTs
RTE I ***         0/        0 bytes allocated/used by ADL DOCLISTs
RTE I ***     17648/    15884 bytes allocated/used by TERMLISTSs
RTE I ***         0/        0 bytes allocated/used by ADL TERMLISTSs
RTE I ***         0 bytes free total according to IR header
RTE I ***    320584 bytes used by physical file according to IR header
```

The `ir_max_shared` parameter should be set to 371398, which is the sum of IR header, hashtables, DOCs, TERMs, DOCLISTs, ADL DOCLISTs, TERMLISTSs, and ADL TERMLISTSs + 10% for

the other IR files adding up to a total of 408537.8. The shared memory base size, in this example, is 32MB so shared memory should be set to 32,000,000 + 408537.8 = 32408537.8. Therefore, the sm.ini file in each host in the horizontal group contains the following parameters and values (rounded):

```
--- sm.ini ---
shared_memory:32408538
ir_max_shared:408538
(other SM parameters for horizontal scaling)
------
```

Tuning Database Queries

Queries that are frequently used should always be fully indexed. It is not practicable – nor does it improve performance – to index every possible query with every possible combination of fields. A badly keyed query increases CPU usage and takes a long time to return results.

General Database Tuning hints

- When writing a query, it is more efficient to use "=" (equal) rather than "#" (starts with). This includes stored queries, views, globallist queries, link queries, and so forth. When Service Manager automatically generates a query, it defaults to "#". In cases where an "=" (equal) query meets the requirements, change all "#" operators to "=".
- When a search is performed from a search screen, the order of the fields in the query is based on the order of the fields in the dbdict.
- Perform checks of the `sm.alert.log` regularly to find frequently used inefficient queries and add indexes to your tables / dbdicts accordingly.
- Perform Database Maintenance on a regular basis. On an RDBMS that means run analyze regularly and use monitoring tools to check for performance.

Tuning queries by Background Processes

Most background processes issue pre-defined queries that are keyed out-of-box. The following background processes issue custom queries regularly:

- Agent
- Marquee
- Lister

To minimize CPU use make sure that these background processes execute only when needed and not too frequently.

First, determine which queries that are frequently issued by background processes are really needed. Since the Agent and Marquee processes were used to create charts and marquee messages that can no longer be viewed using the windows or web clients, it is recommended to not run their schedule records by setting the expiration to a date far in the future. Only the Count USER Connections and Count SYSTEM Connections agent records need to be running in Service Manager.

Next check which globallists use queries other than true queries.

Click **Utilities** –> **Tools** –> **Global Lists** and perform the following advanced search:

sql~="true" and build.startup=true

This search returns all globallist records that are in use that use a query other than "True". Go through this list and ensure that custom-made globallists are still used (otherwise change build.startup to "false" to disable them). Next check the database table on the RDBMS for the Filename stated in the globallist to determine whether the fields used in the globallist query are indexed, as shown above. The keys in the dbdict should always mirror the indexes on the related tables on the RDBMS.

Tuning frequently used foreground queries

The most frequently used foreground queries are user views and favorites, including charts. When creating a view, favorite, or chart as an administrator, always ensure that the query is well formed and keyed. If users have the rights to create their own views, favorites, and charts, analyze the inbox table regularly to either modify or remove queries that negatively impact search performance. Make sure to educate users on how to create a well formed and keyed query. If you identify identical commonly used personal views, change them to a single public view.

Guide users through free-form searches, such as color-coding keyed fields that will help with query performance or removing fields from the search form that are not included in a key. Advise users to use at least one of the keyed fields in their searches. When IR searches are available, remove the deep search option for performance reasons.

Additionally, queries in links and formatcontrol are executed on a regular basis and should always be well-keyed, as discussed in the *Format Control* and *Links* sections of this document.

Tuning database behavior

It is very important to index all frequently used queries that are issued against the database, regardless of which database is used. Out of the box, unique keys in Service Manager are often mapped as unique keys in the RDBMS. In that case, change all unique keys on the RDBMS to primary keys, so that they do not allow NULLS (where the unique key in Service Manager = no NULLS, no duplicates).

Whenever possible, put the indexes and data into different tablespaces and, if possible, on different disks for faster data access.

Important: When creating queries in links, views and favorites, or stored queries, minimize the use of Service Manager functions such as `index()`. Service Manager functions that do not translate to SQL will cause a full table scan whenever they are part of a query. Operators that are safe to use are:
= , #, LIKE, ISIN

In addition, use the tuning tips below to get better response times from your database:

Detailed information about setting up the databases that Service Manager supports (DB2®, Oracle®,and Microsoft® SQLServer®) can be found in *Appendix A of this document*

General RDBMS tuning tips:

- Indexes should be considered on all columns that are frequently used by the WHERE or the ORDER BY clauses. Consider carefully which indexes to add on a table since too *many* indexes can be as detrimental as having too *few*.
- An index is best if the WHERE clause of the query matches the column(s) that are leftmost in the index. When you create an index with a composite key, the order of the columns in the key is important. Try to order the columns in the key to enhance selectivity, with the most selective columns to the leftmost of the key.
- Do not accidentally add the same index twice on a table.
- Drop indexes that are not used.
- Optimize the server kernel. Always tune your disk and network I/O subsystem (RAID, DASD bandwidth, network) to optimize the I/O time, network packet size, and dispatching frequency.
- Adjust your optimizer statistics. Always collect and store optimizer statistics to allow the optimizer to learn more about the distribution of your data and to make more intelligent execution plans.
- Remove large-table full-table scans. Unnecessary full-table scans cause a huge amount of I/O and can drag-down an entire database. The tuning expert first evaluates the SQL based on the number of rows returned by the query. If the query returns fewer than 40 percent of the table rows, it needs tuning. The most common tuning remedy for full-table scans is to add indexes.

- Cache small-table full-table scans. In cases where a full-table scan is the fastest access method, the administrator should ensure that a dedicated data buffer is available for the rows.
- Verify optimal index usage. This is especially important when using the optimizer.
- Use the latest out-of-box `sqldbinfo` file that matches your database type. Do not modify the `sqldbinfo` file unless you are told to do so by HP Software Customer Support. Additionally, always use the same database type on all converted tables. Using different database types (such as DB2 and DB2Universal) within one system will cause more database connections on the RDBMS.

The Oracle® database

The Oracle database gives the database administrator several ways to optimize performance. Some parameters are listed below:

- Adjust optimizer parameters. These include optimizer_mode, optimizer_index_caching, optimizer_index_cost_adj.
    - Beginning with Oracle 10g, the default optimizer mode is all_rows, favoring full-table scans over index access. The all_rows optimizer mode is designed to minimize computing resources and it favors full-table scans. Index access (first_rows) adds additional I/O overhead but returns rows faster back to the originating query. Service Manager's sqloptimizerrows parameter can force the optimizer mode for versions up to Oracle 10: 0 (Use Oracle default), 1 (Optimizer goal is FIRST_ROWS), >1 (Optimizer goal is ALL_ROWS)
- Optimize your instance. Your choice of db_block_size, db_cache_size, and operating system parameters (db_file_multiblock_read_count, cpu_count, and so forth), can influence SQL performance.
- Cache small-table full-table scans. In cases where a full-table scan is the fastest access method, the administrator should ensure that a dedicated data buffer is available for the rows. In Oracle Database 8 and greater, the small table can be cached by forcing it into the KEEP pool.
- Identify high-impact SQL by checking the executions column of the v$sqlarea view. The stats$sql_summary or the dba_hist_sql_summary table can be used to locate the most frequently used SQL

The SQL Server® database

Indexing

Without proper indexes, SQL Server performance will suffer. It is important to use the correct index for each query, since badly selected indexes can slow down SQL Server performance. The following general tips will help:

- Drop indexes that are never used by the Query Optimizer. To provide the up-to-date statistics the query optimizer needs to make smart query optimization decisions, you will generally want to leave the "Auto Update Statistics" database option on.
- Keep the "width" of your indexes as narrow as possible, especially when creating composite (multi-column) indexes.
- The Query Optimizer will always perform a table scan or a clustered index scan on a table if the WHERE clause in the query contains an OR operator, and if any of the referenced columns in the OR clause is not indexed (or does not have a useful index). Because of this, if you use many queries with OR clauses, ensure that each referenced column has an index.
- Periodically (weekly or monthly) perform a database reorganization on all the indexes on all the tables in your database. A reorganization rebuilds the indexes so that the data is no longer fragmented. Fragmented data can cause SQL Server to perform unnecessary data reads, slowing down its performance.

The DB2® database

The default page size in DB2 UDB is 4096 bytes (4K). To attain best performance the DB2 table space for Service Manager should be set to use 32768 byte (32K) pages. For this page size, a value

of 32000 for Maximum Row Size is best. This leaves room for DB2 overhead in each row while still optimizing the way Service Manager uses available space.

Tuning tailoring

Tailoring can greatly influence the system's performance, both in memory and CPU usage.

To minimize memory usage, make sure to use local variables whenever possible and use the `cleanup()` function on thread and global variables when they are no longer needed. Local variables are indicated by $L.<*name*>; global variables start with $G.<*name*> or $lo.<*name*>; all other variables are thread variables. Local variables are available only in the RAD application where they were declared; global variables are available to the user throughout the system; and thread variables are available until you leave the thread (or in simpler terms, leave the notebook tab in the client).

Tuning forms

It is important to tune forms that are displayed to the end user, since large forms use more memory and take longer to display to the user. Use of DVD (Dynamic View Dependency) functionality can slow down form display as well, so it is very important to use DVD efficiently.

There are 2 different kinds of DVD statements:

- Select Statements used in dropdown lists. Select statements are used to limit the values in a dropdown list based on information that was previously entered in the record. These select statements are executed whenever the form is brought up or refreshed, such as when returning from a link. Since these select statements are executed very frequently, it is of utmost importance that they are keyed and efficient. Do not put too many DVD select statements on any one form, since they add load to the database and to the Service Manager server. There are two possible alternatives to select DVD statements:
    - Globallists: If the select statement is not directly dependent on another field that was entered, a global variable defined in the `globallists` file is most efficient. If the value is dependent on the user's Mandanten restrictions, a global variable built via Format Control can replace the DVD statement.
    - Links: If the value of a drop down list is dependent on the value of a previously entered field, a recursive link is the most efficient choice. The link query can then determine the subset of values from which to select. Again, make sure that the link query is efficient and keyed.

Of course, select DVD statements can be used in moderation and are best for queries that return a small subset of records.

- Conditional statements used in Visible, ReadOnly and other conditional properties. Conditional DVD statements are executed whenever the form is displayed or refreshed. It is more efficient to have customer-specific subforms than to use excessive DVD statements. For example, users with a certain capability word can change a set of fields; and other users can only view these values. If you create two subforms with these fields on it, one for the first group, the other for the second group, you can dynamically change the subform name based on the user's capability word and do not need to use any conditional DVD statements. More information about setting up dynamic subforms can be found in the white paper *Best Practices for Multi-Tenant Environments using Service Manager* .

Note: For best client performance, minimize the number of drop-down fields and content of drop-down lists on the form. The form is translated into xml and then sent to the client, with all drop-down values and other available elements on it. If the form is too large, this will cause slow performance. The following items contribute to the size of the form:

- Number of fields on the form
- Number of DVD statements on the form

- Size of Globallists used in dropdowns. For example, if a dropdown uses a globallist with 1000 elements, all these elements are included in the xml that is sent to the client.

Format Control

A lot of tailoring in Service Manager is done in Format Control. Since Format Control statements are evaluated on each execution of Format Control and are not compiled objects, it is very important to set the conditions in Format Control (execution on add, update, delete, display, initial) only for the times that the FormatControl statement needs to be executed. For easiest maintenance, HP Software recommends that you put the tailoring statements that are true for all accesses in that module in the master format control (such as cm3r, probsummary, and so forth); and put only the form-specific tailoring into the detail format control (such as cm3r.hardware, IM.template.close, and so forth). Although moving tailoring to the master records does not improve performance per se, it does minimize both the size of the detail FormatControl records and data redundancy.

Format Control is executed at the following times:

Add	Before adding the record to the database
Update	Before updating the record in the database
Delete	In Change and Request Management: On close processing of the record, others: when deleting the record from the database
Display	Every time the record is displayed: on initial display, on screen refreshs, and upon coming back from fills
Initial	Before bringing up the record for display for the first time

Tuning Queries

Since Format Control queries are executed very frequently, it is important to verify that these queries are indexed correctly. See the section *Tuning Database Queries* on page Tuning Database Queries46 for more information about indexing queries correctly.

Ensure that queries are executed only when needed. If the resulting file variable from a query is used in add and update statements, execute the query only on add and update operations. If the results are used only on the initial display to fill a field with default values, execute the query only on initial. By doing so, calls to the database are minimized.

Periodically check if the queries are still needed. If a statement is removed from calculations, make sure to remove all related statements as well.

Tuning Calculations

Whenever you perform tailoring and during upgrades, verify that the calculation statements are still being used. If not, remove the Format Control line completely. CPU usage increases and decreases depending on how many statements are executed.

Ensure calculations are only executed when needed. To fill initial values, the Format Control should only be executed on initial. To set mandatory values or to overwrite fields, execute on add and update only.

Tuning Subroutines

Check periodically if the subroutine is still needed. If it is not needed, remove it from the Format Control record. As with all Format Control tailoring, only call the subroutines when needed, typically on add and update.

Data Validation – which one when

We have three different ways to do data validation:

- Format Control
- Data Policy

- Data Validation (called from Format Control typically)

Data Policy can be used to set a default value, set a field to be mandatory or to validate a field value against values in an existing file. Data Policy is evaluated for every record in the table, independent on which form the record was displayed in.

Format Control validations can be used both for expression based validations (validations section) and verification that the entered value matches a value in an existing table (query section). Format Control is available for all access to the table (master Format Control) or on a form-by-form basis.

Data Validation is typically called from Format Control subroutines. It is the most versatile of all validation methods. It can present users with a list of valid entries on failed validation, it can verify that the entry matches a value in an existing table, it can call a RAD application for validation, and it can determine how to proceed if no valid record was found.

JavaScript® / ScriptLibrary

JavaScript code is integrated within Service Manager and can be used for tailoring in Format Control, Triggers, via the ScriptLibrary, Links, and so forth. To best use JavaScript in Service Manager, make the programs as simple as possible, use existing variables and functions rather than re-creating functionality. Plan the variables and the code in detail to avoid overhead. Sometimes, programming functionality takes less code in the Service Manager language than in JavaScript, sometimes JavaScript is more efficient. Always use the language that needs the least number of statements to perform the job. It is possible in most cases to mix and match both, such as in Format Control, calculate variables in JavaScript and then utilize them in Calculations.

Scripts

Scripts often use variables that are passed through the different steps. It is more efficient to use the file variable and fill fields in the file variable directly, rather than use variables first that then are written into the file variable. If you use thread or global variables specifically for the script, use the cleanup() function in the last step to ensure that the memory is freed again. Try to keep the amount of script records in the script process to a minimum and plan the statements and RAD applications for best efficiency.

Wizards

Wizards are a very powerful tailoring tool, providing the functionality of a script by stepping through different panels based on conditions; and in being able to call Processes, FormatControl, and execute expressions. Wizards have a tab called “cancel expressions”. Always use this tab to clean up variables, since most variables used in wizards are thread or global variables. This is needed because wizards execute many applications, and local variables are not available throughout. In wizards as in scripts, plan the workflow in detail to prevent infinite loops.

Links

Queries in links are executed frequently and may use elaborate concatenated statements, so it is very important to have them indexed correctly. When concatenating query statements, make sure not to duplicate a query against a single field such as: company=”Test” and company=”Test”. Try to concatenate the query so that the most limiting field is the leftmost field in the query (such as company=”Test” and contact#”A” – where fewer people work for company “Test” than have a name starting with “A”).

Minimize the statements in the link pre- and post expressions and minimize the use of elaborate JavaScript. Enter only statements that directly affect which records the link will return.

Display

The display application is responsible for displaying a record to the user, including all options available to the user on that record. For best performance and usability, disable display option records that users do not need by giving them a user condition of *false*. Minimize statements on the display screen. Display Events can be used to trigger an event when the user clicks or modifies a

value on a form. Use Display Events sparingly, since they will be checked whenever the user enters information into the form.

Document Engine

The Document Engine is the central tool that handles all Service Manager transactions. Tailoring is mostly done in the Process records calling an initial Process to set information about the format to display, or adding and modifying Processes to execute statements and RAD applications. When tailoring the Document Engine, make sure to use unique names for the Process records for later upgrades. Use existing variables whenever possible and clean up thread and global variables that are no longer used.

Tuning integrations

Service Manager has three methods of exchanging data with third-party applications:

- SCAuto
- Connect-It
- Web Services.

All three are used for the bi-directional exchange of data. SCAuto and Connect-It use a pre-defined set of integrators. Web Services can integrate with any Web Services application. Connect-It is best used for scheduled data transfer, whereas SCAuto and Web Services are typically interaction-based.

Connect-It and SCAuto use Event Services with eventmaps for moving data into and out of Service Manager. For best performance, put into the event maps only those fields that need to be moved to and from the third party product. Use only keyed queries in the event registration and try to transport only records that need to be moved.

Web Services uses the SOAP API to get data into and out of Service Manager. Service Manager exposes fields via the External Access information. Consuming Web Services are executed within Service Manager via JavaScript. Exposing a Service Manager Web Service uses Document Engine States and Processes to define the actions. For consuming Web Services, make sure to write the JavaScript code as efficiently as possible, minimizing the number of statements used, and making sure that all possible error situations are covered. For exposing Web Services, limit the numbers exposed in the extaccess table to only the required fields and follow the steps discussed in the *Document Engine* tuning section in this document. For more information on tuning web services, refer to the Web Services - Service Manager as Web Services provider section of this document.

Regular Maintenance

Backups

Back up your data at least daily, using RDBMS tools when the data is converted to an RDBMS. Test your backups at least monthly to ensure data integrity.

Purging and Archiving

The following files should be purged (and archived if required by your business process) on a regular basis:

- mail
- msglog
- syslog
- spool
- eventin

- eventout
- devaudit (on a production system, development auditing should be turned off)

The following files should be archived and purged on an as-required basis. If information in the files needs to be available to users after they were purged from the primary dbdict, you can create an archive<*name*> dbdict for the data as a copy of the original dbdict. Use the `copy.file` application to copy (project) the information from the primary dbdict to the archive<*name*> dbdict and make the archive dbdict available to a subset of users for searching.

- probsummary
- problem
- screlation
- work
- activity
- SYSATTACHMENTS
- cm3r
- cm3rpage
- cm3t
- cm3tpage
- cmcalendar
- ApprovalLog
- AlertLog
- incidents
- activityservicemgmt
- ocml
- ocmq
- ocmo
- clocks
- cmlabor
- cmparts
- activityknownerror
- activityproblem
- activityproblemtask
- knownerror
- rootcause
- rootcausetask
- audit
- contract
- curconvert
- kmsearchhistory
- kmusagehistory
- kmstats

The following files should be monitored when in use:

- sqlqueue
- irqueue

Log file maintenance

Both the `sm.log` and the `sm.alert.log` files may grow very large if not monitored and cleaned up occasionally. Parameters in the `sm.ini` file allow for size-based rolling of the `sm.log` file, which should be set to about 10 MB, and a history of 2 - 3 log files to keep. The `sm.alert.log` file cannot be rolled automatically and should be cleaned up manually about once a month. Alert filtering, which is discussed in the section *sm -alertfilters* on page12, can be used to minimize output to the `sm.alert.log` file as well.

Appendix A

Optimizing Service Manager performance on the RDBMS

Service Manager provides a variety of options for mapping your data to an RDBMS. This section discusses how to optimize your mapping for pure performance and reporting simplicity. Some tables may be optimized for speed, some may be optimized for reporting, depending on their primary use. The out-of-box mapping is optimized for speed only.

For RDBMS specific tuning procedures, please refer to your RDBMS vendor documentation.

Optimizing for speed

Minimize tables and rows in a mapped record

To optimize your Service Manager RDBMS implementation for speed means reducing physical reads on the RDBMS server. Generally speaking, Service Manager will fetch a single complete Service Manager record from your RDBMS database. To optimize the speed of this process, it is important to place a Service Manager record in as few rows in as few tables as possible.

If speed is an important issue, map arrays of characters either as CLOB fields or BLOB fields in the main RDBMS table.

Index efficiently

Service Manager does not usually place an extremely high update/insert transaction load on an RDBMS server. Most of the interactions with the RDBMS database involve simple single-row selects. You can improve the speed of these selects by indexing your RDBMS tables efficiently. Avoid the temptation to under-index; this may speed updates or inserts, but slows retrieval.

Optimizing for reporting

Since the out-of-box system is optimized for speed, you may need to remap tables for reporting, or for best results create a reporting data store (RDS) that meets the reporting needs.

Avoid binary data

Store arrays upon which you want to report as either long text fields or as separate array tables. If you choose to use CLOB fields, check to ensure that your reporting package can handle the data in question.

Know how arrays are used

Service Manager will map an array of numbers as a long text field, but will not perform row aggregate functions against the contents of that array. For example, if you map an array containing 35 price quotes for a PC into a long text field, you are unable to select out the maximum, the minimum, or the mean price quote using the standard SQL syntax.

Generally speaking, you should map arrays on which you will need to perform row–level functions as array tables.

Service Manager modifications

Multiple tables

Service Manager supports vertical field splitting within a file. In other words, one Service Manager file, for example, the example file can map to multiple tables: fields 1, 3, and 7 in file examplem1, fields 2, 4, and 5 in file examplem2, and field 6 in file examplea1. Having too many different tables will negatively impact performance due to the additional queries and joins required to retrieve the complete record.

Tuning indexes

To tune indices:

1. Determine which indexes are not being used, and drop them.

2. Look at long-running queries and add indexes where needed

Note: You can also locate long-running queries by looking for sqllimit exceeded messages in the Service Manager sm.log file.

For more information

Please visit the HP Software support Web site at:

www.hp.com/go/hpsoftwaresupport

This Web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track progress on support cases
- Submit enhancement requests online
- Download software patches
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Note: Most of the support areas require that you register as an HP Passport user and sign in. Many also require an active support contract.

To find more information about support access levels, go to the following URL:

www.hp.com/go/hpsoftwaresupport/new_access_levels

To register for an HP Passport ID, go to the following URL:

www.hp.com/go/hpsoftwaresupport/passport-registration

Technology for better business outcomes