HP Service Manager

Software Version: 9.41

For the supported Windows® and UNIX® operating systems

Server performance tuning help topics for printing

Document Release Date: September 2015
Software Release Date: September 2015

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

For a complete list of open source and third party acknowledgements, visit the HP Software Support Online web site and search for the product manual called HP Service Manager Open Source and Third Party License Agreements.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hp.com/.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support site at: https://softwaresupport.hp.com.

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

HP Software online 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 support customer, you can benefit by using the support website to:

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

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: https://softwaresupport.hp.com/web/softwaresupport/access-levels.

HPSW Solutions Catalog accesses the HPSW Integrations and Solutions Catalog portal website. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01702710.

About this PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format. Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not

be present in this PDF version. Those topics can be successfully printed from within the online help.

Contents

Server Performance Tuning 6

Scheduled restart of Service Manager processes 7

Example: Restart all processes on a host 10

Example: Restart one process 11

Defining batch sizes in the counters file 13

Defining record identifiers in batches using the numbers file 15

Lock management 17

Scheduled processes 23

Access the schedule file 23

Add an agent to the startup agent record 23

Configure the alert periodic schedule record 24

Create an anubis agent record 25

Create an anubis schedule record 26

Restart stopped background processes automatically 27

Start a scheduled process 27

Stop a process 28

Unlock a deadlocked resource 28

Scheduled processes 29

Shared memory 31

Automatic selection of shared memory address 31

Example: response time monitoring (RTM) output 32

RTM:5 37

Example: semaphore reports 39

Debug a suspended or hung system 39

View shared memory statistics 40

Shared memory storage report 41

Example: Shared Memory report 42
Type statistics 42
Shared memory guide 43
Send Documentation Feedback 44

Server Performance Tuning

The HP Service Manager server includes a number of performance tuning options you can configure after installation. The following table lists the performance tuning options you can enable or configure from a new development environment installation.

Feature	Description	Default state in new installations
"Shared memory" on page 31	A configuration option that allows you to specify how much of the Service Manager host's system resources are available for processes and threads	**Enabled**

Scheduled restart of Service Manager processes

The restart command allows administrators to schedule the restart of one or all Service Manager processes on a host. Restarting one or more processes allows Service Manager to offer high availability without having to restart the entire Service Manager cluster. Typically, administrators want to restart Service Manager processes for one of the following reasons:

- The Service Manager system is running low on system resources while at normal load
- A particular Service Manager processes is consuming a large amount of system resources
- The administrator has some need to regularly restart Service Manager processes

These symptoms may arise from various causes such as faulty tailoring changes or unstable third-party libraries that increase memory consumption on your system. Restarting one or all Service Manager processes allows you to temporarily workaround any performance issues while you diagnosis the root cause and determine a permanent fix.

The restart command has two basic modes of operation:

- Restart a particular process
- Restart all processes on a host

Restart a particular process

If you can identify the processes comsuming system resources, you can restart them individually by process ID with the pid parameter. A process restart uses the following workflow:

1. The restart command notifies the process of a restart request
2. The process operates normally during the restart waiting period
3. The process goes into quiesce mode during the restart grace interval
4. The process restarts

You can schedule when a Service Manager process restarts with the value of the restart command. The value of the restart command determines the minimum amount of minutes a process waits before restarting. By default, there is no restart waiting period value so a process enters restart immediately.

During the restart waiting period, an administrator can cancel the restart by reissuing the restart command with a value of -1. You can specify the same process ID you originally listed with the pid parameter to cancel a process restart. You can only cancel a restart command during the restart waiting period. After a process is in the grace interval, you can no longer cancel a restart command.

After the restart waiting period has expired, the server checks to see if there is a restartGraceInterval value. This parameter determines how long the process will be quiesced and not accept any new connections (not even administrator accounts). This allows users currently connected to the process to complete their work and log off prior to the restart. Service Manager displays the following message to users during the grace interval:

```
Your Service Manager session is shutting down in %d minutes for maintenance. Please save your work and log out. You can log in again immediately.
```

Service Manager replaces the variable %d with the restartGraceInterval value. You can edit or localize the restart message from the notification engine. It is IDS_ALERT_RESTART message 126.

By default, there is no restartGraceInterval value so all processes immediately restart. If you provide a restartGraceInterval, the process will remain quiesced for a number of minutes up to the value supplied or until all users log off, whichever comes first. If either condition is met the process restarts.

Note: Processes dedicated to Web Services connections or background schedulers ignore the restartGraceInterval value because they are stateless connections that do not require a user to log off. These processes restart immediately after the restart waiting period expires. After restart, the server will only run the background scheduler processes listed in the sm.cfg file or started from the OS command prompt. Background schedulers started from the System Status form are not automatically resumed.

Restart all processes on a host

If you cannot identify particular Service Manager processes that are consuming system resources, you can restart all Service Manager processes on a host with the host parameter.

Note: A host restart command does not restart the load balancer process. The only way to restart a load balancer process is to specify it by process ID.

A host restart uses the following workflow:

1. The restart command notifies all processes on the host of the restart request
2. The restart command randomly assigns a time extenstion to the restart time of each process
3. The processes operates normally during the restart waiting period

4. The processes go into quiesce mode during the restart grace interval
5. The process restarts

When Service Manager processes receive a restart command they first calculate a future restart time based on the values provided with the restart command and the restartRandMax sm.ini parameter. The value of the restart command determines the minimum amount of time a process waits before restarting. By default, there is no restart waiting period value so the server randomly assigns a time extension. The value of the restartRandMax parameter extends the restart waiting period by a random amount of minutes from 0 to the value provided in the command.

The purpose of the random restart time extension is to minimize the chance that two or more processes restart at the same time since each process that restarts briefly reduces system capacity. If you are only restarting one process, there is no need to stagger restart times to preserve capacity and therefore the server ignores any restartRandMax value.

During the restart waiting period, an administrator can cancel the restart by reissuing the restart command with a value of -1. You can only cancel a restart command during the restart waiting period. After a process is in the grace interval, you can no longer cancel a restart command.

After the restart waiting period has expired, the server checks to see if there is a restartGraceInterval value. This parameter determines how long the process will be quiesced and not accept any new connections (not even administrator accounts). This allows users currently connected to the process to complete their work and log off prior to the restart. Service Manager displays the following message to users during the grace interval:

```
Your Service Manager session is shutting down in %d minutes for maintenance. Please
save your work and log out. You can log in again immediately.
```

Service Manager replaces the variable %d with the restartGraceInterval value. You can edit or localize the restart message from the notification engine. It is IDS_ALERT_RESTART message 126.

By default, there is no restartGraceInterval value so all processes immediately restart. If you provide a restartGraceInterval, the process will remain quiesced for a number of minutes up to the value supplied or until all users log off, whichever comes first. If either condition is met the process restarts.

Note: Processes dedicated to Web Services connections, background schedulers, or the load balancer process ignore the restartGraceInterval value because they are stateless connections that do not require a user to log off. These processes restart immediately after the restart waiting period expires. After restart, the server will only run the background scheduler processes listed in the sm.cfg file or started from the OS command prompt. Background schedulers started from the System Status form are not automatically resumed.

Recommendations

While a process is quiesced, the system loses some connection capacity because there are fewer processes to accept connection requests. Restarting one Service Manager process typically takes a short amount of time, approximately 30 seconds depending upon the system. In general, quiescing processes reduces capacity for longer than restarting them (several minutes quiesce time compared to 30 seconds restart time). For this reason, HP recommends you set a high restartRandMax value to minimize the chance of two processes restarting at the same time, and a low restartGraceInterval value to minimize the amount of time your system has reduced capacity from quiesced processes.

If you want to regularly schedule the restart of one or all of your Service Manager processes, you must use your operating system's scheduling tools to run the restart command.

Example: Restart all processes on a host

The following example illustrates how to restart all processes on a Service Manager host because the system is running out of memory. This scenario uses the following system configuration:

System property	Value
Number of hosts	3
Total Service Manager processes	21
Threads per process	40
Maximum number of concurrent users expected	700
Maximum user capacity	840

This horizontally scaled configuration can support 700 concurrent users with an extra 20% capacity to handle high usage and outages.

Problem

The system administrator recently applied some new JavaScripts and since then has noted out of memory exceptions in the sm.log file. All Service Manager processes have high memory usage even when only at half capacity (20 concurrent user connections per process).

Recommendation

While the root cause of the high memory usage is being determined, the system administrator can schedule the restart the Service Manager processes on each host to occur periodically. Restarting the processes will temporarily free up system memory until the next system maintenance down time or until the root cause of the problem is identified and fixed.

The administrator uses an operating system scheduler to run the following commands at midnight every three days:

```
sm -restart:0 -host:<host1>
sm -restart:180 -host:<host2>
sm -restart:360 -host:<host3>
```

These commands cause host1 to restart immediately, host2 to restart in three hours, and host3 to restart in six hours. To avoid having all Service Manager processes on a host quiesced at one time, the administrator adds the following parameters to the sm.ini file on each host:

```
restartRandMax:60
restartGraceInterval:15
```

The restartRandMax:60 value ensures that each host finishes restarting all Service Manager process within one hour of receiving the restart command. In this case, host1 restarts anywhere from 0 to 60 minutes after receiving the restart command; host2 restarts after 180 to 240 minutes, and host3 restarts after 360 to 420 minutes. The restartGraceInterval:15 value ensures that users on each Service Manager process have 15 minutes to save their work and re-login before the process restarts. Users who log off one process can immediately re-login and continue work on another available process.

Note: A host restart command does not restart the load balancer process. The only way to restart a load balancer process is to specify it by process ID. The system cannot accept new connection requests until after the load balancer process restarts.

Example: Restart one process

The following example illustrates how to restart one processes on a Service Manager host because it is running consuming a high amount of system resources. This scenario uses the following system configuration:

System property	Value
Number of hosts	3

System property	Value
Total Service Manager processes	21
Threads per process	40
Maximum number of concurrent users expected	700
Maximum user capacity	840

This horizontally scaled configuration can support 700 concurrent users with an extra 20% capacity to handle high usage and outages.

Problem

The system administrator notes that one Service Manager process is consuming a large amount of system resources such as CPU time or system memory.

Recommendation

While the root cause of the issue is being determined, the system administrator can schedule the restart of the Service Manager process. Restarting the process will temporarily free up system memory until the next system maintenance down time or until the root cause of the problem is identified and fixed.

The administrator uses the following command to restart the affected process:

```
sm -restart:0 -host:15.80.177.12 -pid:3433
```

This command causes process ID 3433 on the identified host to restart immediately. To provide the currently connected users time to save their work, the administrator adds the following parameters to the sm.ini file:

```
restartGraceInterval:15
```

The restartGraceInterval:15 value ensures that users on the Service Manager process have 15 minutes to save their work and re-login before the process restarts. Users who log off one process can immediately re-login and continue work on another available process.

Note: A host restart command does not restart the load balancer process. The only way to restart a load balancer process is to specify it by process ID. The system cannot accept new connection requests until after the load balancer process restarts.

Defining batch sizes in the counters file

Administrators can allocate blocks of record identifiers for each servlet in horizontal and vertical scaling implementations. By allocating record IDs in advance, administrators can reduce traffic between the Service Manager server and the RDBMS, which improves overall system performance.

Note: Setting a batch size value replaces the fastcounters parameter from previous versions.

Administrators allocate record IDs in batches from the `counters` file. When you define a batch size, Service Manager retrieves that number of ID keys from the RDBMS and stores the keys in shared memory for later use. Each time someone creates a new record, the system assigns it an ID key from shared memory until there are no more IDs available. Once the system exhausts its supply of IDs, it requests the next batch of IDs from the RDBMS. Allocating IDs in advance improves performance, because retrieving the reserved ID keys from shared memory is more efficient than connecting to the RDBMS and requesting a new ID key with every new record.

Batch file size recommendation

HP recommends you set the batch file size to the number of new records you expect users to create in an hour divided by the number of servers in the implementation. For example, if you expect to generate 200 incident records an hour and have only one server in your implementation, then set the **Batch Size** value to 200. If your system consists of multiple servers, such as in a horizontal scaling implementation, then divide the total records per hour by the number of servers. For example, a horizontal scaling implementation with four servers only needs a batch size of 50 for each server to equal 200 incidents per hour.

Note: In a horizontally scaled system, each server requests its own batch of ID keys. Since new records may be created from processes running on different servers in a horizontal scaling implementation, the ID numbers of new records may not be in sequential order. For example, process A on server 1 has reserved ID keys 1-100 and process B on server 2 has reserved ID keys 101-200. The first record a user creates on process A will have ID 1, but if the very next record created comes from process B, then the ID will be 101, not 2.

If you do not specify a batch size value, Service Manager uses the default batch size of 1 and fetches a new ID from the RDBMS for each new record you create.

Example data from a counters file

```
TABLE_NAME  COLUMN_NAME  CURRENT_VALUE  BATCH_SIZE
irqueue     counter      0              100
```

```
schedule    schedule.id  2587564        1000
scirexpert  record.id    5198034        5000
```

Defining record identifiers in batches using the numbers file

Administrators can define record identifiers and allocate those identifiers in batches from the numbers file for new interactions, incidents, and changes. By allocating record IDs in batches in advance, administrators can reduce traffic between the Service Manager server and the RDBMS, which improves overall system performance.

Note: Currently the counters file is used by administrators to define the table and field where a new record number is going to be stored. However, the numbers file is still available for those who previously chose to define record numbers using the numbers file and are upgrading. This file has been enhanced to work more like the counters file. For more information on the counters file, see the related topics.

When using the numbers file, administrators define the following:

- Batch size.
- Starting number.
- Identifying prefix for each module. For example, IM for Incident Management records.
- Whether numbers are going to increment or decrement.
- The table and field where the new numbers are to be stored.

When you define a batch size, Service Manager retrieves that number of ID keys from the RDBMS and stores the keys in shared memory for later use. Each time someone creates a new record, the system assigns it an ID key from shared memory until there are no more IDs available. Once the system exhausts its supply of IDs, it requests the next batch of IDs from the RDBMS.

HP recommends you set the batch file size to the number of new records you expect users to create in an hour, divided by the number of servers in the implementation. For example, if you expect to generate 200 incident records an hour and have only one server in your implementation, then set the Batch Size value to 200. If your system consists of multiple servers, such as in a horizontal scaling implementation, then divide the total records per hour by the number of servers. For example, a horizontal scaling implementation with four servers only needs a batch size of 50 for each server to equal 200 incidents per hour.

Note: In a horizontally scaled system, each server requests its own batch of ID keys. Since new records may be created from processes running on different servers in a horizontal scaling implementation, the ID numbers of new records may not be in sequential order. For example, process A on server 1 has

reserved ID keys 1-100 and process B on server 2 has reserved ID keys 101-200. The first record a user creates on process A will have ID 1, but if the very next record created comes from process B, then the ID will be 101, not 2.

If you do not specify a batch size value, Service Manager uses the default batch size of 1 and fetches a new ID from the RDBMS for each new record you create.

Example data from a numbers file

```
TABLE_NAME  COLUMN_NAME  CURRENT_VALUE  BATCH_SIZE
irqueue     counter      0              100
schedule    schedule.id  2587564        1000
scirexpert  record.id    5198034        5000
```

Lock management

The locking mechanism in versions of Service Manager (SM) prior to version 9.31 had a locking mechanism that used a multicasting to request and obtain a lock on a resource in Service Manager. This locking mechanism was implemented using a Peer Lock in the JGroups toolkit. However, there were several issues with this implementation, which are addressed with the new locking mechanism introduced in this document.

Issues in Service Manager Multicast Communication

Service Manager's previous multicast implementation suffers from the following limitations:

• All nodes must be contacted by the node that is requesting the resource. This leads to high overhead per request.

• All nodes must give approval to a request, regardless of whether a node is using a resource or not.

• A node that does not respond represents a single point of failure for the system, even if that node has no other connection to the resource or the requesting node.

• If one node does not respond, the request must be re-issued by the originating node, which increases overhead even further.

• Nodes that are slow to respond will eventually be removed from the node cluster.

• Scalability is very poor.

• The potential for netstorms is high (wherein every node is attempting to request permission from every other node, leading to n^2 requests).

New Locking Mechanism Overview

The new locking mechanism consists of a record entry for each locked resource in a database table. The new Lock table (for exclusive locks) and LockShared table (for shared locks) have been created to house these records. See the following tables for details on the structure and fields of the Lock and LockShared tables.

Note: The only difference between these two tables is the value in the TYPE field, and the primary key for the LockShared table is a combination of the LOCKID, pID, tID, and IP fields.

The Lock Table

Field	Type	Null	Key	Default	Extra
LOCKID	varchar (600)	N	PK		LockID is a hex value of ResourceName. This ensures that the ResourceName field can be either case-sensitive or case-insensitive.
RESOURCENAME	varchar (200)	N			Logic lock ID
TYPE	char(1)				Exclusive / Shared lock
PID	float				The Process ID that holds the lock.
TID	float				The Thread ID that holds the lock.
RADTHREADID	float				The Rad Thread ID that holds the lock.
SESSIONID	float				The session ID that holds the lock.
REASON	VARCHAR2 (60)				Application private
USER	VARCHAR2 (60)				SM login user (For example, "falcon")
HOSTNAME	VARCHAR2 (60)				The host that holds the lock.
IP	VARCHAR2 (60)				The IP address of the host that holds the lock.
DEVICENAME	VARCHAR2 (60)				The device that holds the lock.
For the bg scheduler thread, the devicename is "SYSTEM"					
LOCKAT	datetime				When the lock is obtained.
STARTAT	datetime				When the lock was requested.
RETRYCOUNT	float				Number of times the lock was requested.
HEARTBEAT	float				Updated periodically by the lock holder.
SUSPECTED	float				Indicates another node suspects the lock holder has failed.
SYSRESTRICTED	CHAR(1)				Indicates no modification allowed for

The Lock Table, continued

Field	Type	Null	Key	Default	Extra
					end-users
SYSMODCOUNT	float				
SYSMODUSER	VARCHAR (60)				
SYSMODTIME	datetime				

The LockShared Table

Field	Type	Null	Key	Default	Extra
LOCKID	varchar (600)	N	PK		LockID is a hex value of ResourceName. This ensure that the ResourceName field can be either case-sensitive or case-insensitive.
RESOURCENAME	varchar (200)	N			Logic lock ID
TYPE	char(1)	N	PK		Shared lock
PID	float		PK		The Process ID that holds the lock.
TID	float		PK		The Thread ID that holds the lock.
RADTHREADID	float				The Rad Thread ID that holds the lock.
SESSIONID	float				The session ID that holds the lock.
REASON	VARCHAR2 (60)				Application private
USER	VARCHAR2 (60)				SM login user (For example, "falcon")
HOSTNAME	VARCHAR2 (60)				The host that holds the lock.
IP	VARCHAR2 (60)				The IP address of the host that holds the lock.
DEVICENAME	VARCHAR2 (60)				The device that holds the lock.
For the bg scheduler thread, the devicename is "SYSTEM"					

The LockShared Table, continued

Field	Type	Null	Key	Default	Extra
LOCKAT	datetime				When the lock is obtained.
STARTAT	datetime				When the lock was requested.
RETRYCOUNT	float				Number of times the lock was requested.
HEARTBEAT	float				Updated periodically by the lock holder.
SUSPECTED	float				Indicates another node suspects the lock holder has failed.
SYSRESTRICTED	CHAR(1)				Indicates no modification allowed for end-users
SYSMODCOUNT	float				
SYSMODUSER	VARCHAR (60)				
SYSMODTIME	datetime				

Lock Behavior

Locks may be either shared locks or exclusive locks. A shared lock allows multiple nodes to read the data from a resource. An exclusive lock may only be obtained if any shared locks on a resource have been released by the nodes that hold them.

Exclusive Lock

The process by which a node obtains an exclusive lock is as follows:

1. A node that requests an exclusive lock tries to insert a record into the Lock database table to see whether a resource is available.
2. If there is no record for this resource in the Lock table, the resource is available and the node holder obtains the lock and inserts a record into the Lock table.
3. If there is a record in the Lock table, the node will fail to obtain a lock.

To unlock an exclusive lock, the corresponding lock record is removed from the Lock table.

Shared Lock

The process by which a node obtains a shared lock is as follows:

1. A node that requests a shared lock tries to insert a record into the Lock database table to see whether a resource is available.
2. If there is no record for this resource in the Lock table, the node holder inserts a record into the Lock table and proceeds to step 4.
3. If there is a record in the Lock table, the TYPE field of that record is checked. If the TYPE field is “Exclusive” the lock requested is rejected. If the TYPE field is “Shared,” proceed to step 4.
4. The node tries to insert a record into the LockShared table. If the insertion is successful, the shared lock request is granted. If the insertion fails, the shared lock request is rejected.

To unlock a shared lock, the corresponding lock record is removed from the LockShared table. Then, the corresponding lock record is removed from the Lock table unless other records share the same resource ID.

Note: There is no mechanism to escalate a Shared lock to an Exclusive lock. To obtain an exclusive lock on a resource, all shared locks must be released.

Lock Retry, Timeout, and Heartbeat

Each process that requires a lock has a dedicated LockHandler thread to handle all lock related operations. When a process needs to execute a lock/unlock operation, the process places a request in queue. The LockHandler reads the queue and attempts to insert the appropriate records in to the Lock or LockShared tables. Additionally, the LockHandler will also return the response to the process that has requested the lock.

The LockHandler will attempt to retry a “wait” lock request every two seconds until the lock attempt succeeds. If a “no-wait” lock request is specified, the LockHandler will reattempt lock acquisition immediately.

- It is possible that a node may obtain a lock, and then fail to release the lock for several reasons. For example, the node could fail, or a problem with the network may prevent communication between the nodes. To prevent other nodes from waiting for an unresponsive lock owner, the following heartbeat mechanism has been implemented:

- When a process requests a lock and finds the lock is already held and the heartbeat value has not been updated, it sets the “Suspected” flag of the lock record in the database table to 1.
- The lock owner updates the heartbeat and checks the “Suspected” flag of the lock record in the database table every 60 seconds.
- If the “Suspected” flag is set to 1, the lock owner sets the “Suspected” flag back to 0 and then increases the heartbeat.
- If the lock owner has failed to update the “Suspected” flag and heartbeat, the process that is waiting will periodically recheck the “Suspected” flag. After 10 minutes (the default value specified in the deadnodelocktimeout parameter), the lock is forcibly removed by deleting the lock record from the database table.

New Parameter

This new locking mechanism implements the deadnodelocktimeout parameter. This parameter specifies the amount of time that must elapse before a process forcibly removes a lock from the Lock or LockShared table. By default, this parameter is set to 10, which indicates that 10 minutes must elapse before a record is forcibly removed. 10 minutes is also the minimum value for this parameter.

This parameter is specified in the sm.ini file. Changing the value of this parameter does not require a restart of the server.

Scheduled processes

You can start RAD applications, known as scheduled processes, as unattended background processes. Scheduled processes are also known as background agents, schedulers, processors, and background tasks. Scheduled processes run at set intervals and check for specific conditions before executing a RAD application. Each scheduled process has one or more schedule records that determine the conditions that must be met for the RAD application to run. You can start scheduled processes from the Scheduler form.

Note: There can be only **one** kmupdate process running at any time. Starting more than one kmupdate process causes unpredictable behavior on the search engine server.

Access the schedule file

Applies to User Roles:

System Administrator

You can access the schedule file to see a list of outstanding scheduled tasks your HP Service Manager system will perform. Scheduled tasks are created either from agent initialization records or from triggers.

To access the schedule file, type the following command into the Service Manager command line:

```
sch
```

The schedule.looksee.g form opens.

Add an agent to the startup agent record

Applies to User Roles:

System Administrator

Adding an agent to the startup agent record allows HP Service Manager to automatically start a background process on startup.

To add an agent to the startup agent record:

1. Type the following command into the Service Manager command line:

   `info`

   The info.startup.g form opens.

2. In the Type field, type the following:

   `startup`

3. Click **Search**.
   The startup agent record opens.

4. Scroll to the first available open record in the Processor Information section.

   Type or select the following agent information.

Field	Description
Name	Type the name you want to use for the agent. For example, `message1`. This name appears in the System Status list of background processes.
RAD Application	Type `scheduler`.
Class	Type the name for the type of objects this processor will handle. For example, `message`.
Wakeup Interval (secs.)	Type the number of seconds that you want the message processor to wait before checking for messages. For example, `60`.
Priority	Type the numeric CPU priority you want the message processor to have. For example, `1`.

5.

6. Click **Add**.
   Service Manager adds the Information record.

Configure the alert periodic schedule record

Applies to User Roles:

System Administrator

To configure the alert periodic schedule record:

1. Click **Tailoring** > **Database Manager**.
   The format.prompt.db.g form opens.
2. In the Form field, type schedule.
3. Click **Search**.
   Service Manager displays of a list of schedule forms.
4. Click **schedule**. The schedule.g form opens.
5. In the Name field, type Alert Periodic.
6. Click **Search**.
   Service Manager opens the details for the Alert Periodic schedule record.
7. In the Repeat Interval field, type the time interval you want Service Manager to wait between starting the alert periodic scheduled process.
   Type the time period in the following format: *Dayshours*:*minutes*:*seconds*. For example, 4 03:02:01 starts the alert periodic process every 4 days, 3 hours, 2 minutes, and 1 second.
8. In the Application field, verify that Service Manager displays us.alert.periodic for the application name.
9. Click **Save**
   Service Manager updates the schedule record updated.

Create an anubis agent record

Applies to User Roles:

System Administrator

Important: This is the old method to restart stopped background processes automatically. For better system performance, you are recommended to use the enableAnubisMonitor and anubisPollInterval parameters instead.

You can create an anubis agent record to restart stopped background processes automatically.

To create an anubis agent record:

1. Type the following command into the HP Service Manager command line, and press **Enter**:

   ```
   info
   ```

   The info.startup.g form opens.

2. Type or select the anubis agent record information.

3. Click **Add**.
   Service Manager adds the Information record.

Create an anubis schedule record

Applies to User Roles:

System Administrator

Important: This is the old method to restart stopped background processes automatically. For better system performance, you are recommended to use the enableAnubisMonitor and anubisPollInterval parameters instead.

You can create an anubis schedule record to restart stopped background processes automatically.

To create an anubis schedule record:

1. Type the following command into the HP Service Manager command line, and press **Enter**:

   ```
   sch
   ```

   The schedule.looksee.g form opens.

2. Type or select the following information.

Field	Description
Name	Type **anubis**.
Class	Type **anubis**.
Expiration	Type or select any date and time prior to the current time.
Scheduled Class	Type **anubis**.
Repeat Interval	Type or select a repeat interval. For example, type 00:00:30 for a thirty-second repeat interval.

Field	Description
Application	Type **apm.anubis**.

3. Click **Add**.
   Service Manager adds the schedule record.

Restart stopped background processes automatically

Applies to User Roles:

System Administrator

Important: This is the old method to restart stopped background processes automatically. For better system performance, you are recommended to use the enableAnubisMonitor and anubisPollInterval parameters instead.

To restart stopped background processes automatically, you can create a special background process called anubis that checks the system status of each background process.

Note: The anubis background process can only restart processes listed in the startup info record. In order to stop any of these processes, you must first stop anubis, otherwise anubis will restart them on its next scheduled run time.

To restart stopped background processes automatically:

1. Create an anubis agent record. See "Create an anubis agent record" on page 25.
2. Create an anubis schedule record. See "Create an anubis schedule record" on the previous page.
3. Start the anubis process. See "Start a scheduled process" below.

   Note: You can also add the anubis background process to the startup record to have HP Service Manager automatically start anubis.

Start a scheduled process

Applies to User Roles:

System Administrator

You can start a scheduled process from the system status scheduler. All scheduled processes run in the background at pre-defined intervals.

To start a scheduled process:

1. Click **Miscellaneous** > **System Status**. The system.status.list.g form opens.
2. Click **Start Scheduler**. HP Service Manager displays a list of RAD applications that you can start.
3. Double-click the RAD application you want to start.

   The system.status.lit.g form reopens, and Service Manager displays the message: System background scheduler: *Process* started at: *DateTime*.

Stop a process

Applies to User Roles:

System Administrator

You can manually stop any running process from the system status form. If you want the server to automatically stop any process after a pre-defined period of inactivity, use the Start Inactivity Timer form.

To stop a process:

1. Click **System Status**. The system.status.list.g form opens.
2. To stop a process, type **k** in the Command field.
3. Click **Execute Commands**.

   Service Manager stops the selected process.

   Note: Service Manager stops the process without issuing a warning.

Unlock a deadlocked resource

Applies to User Roles:

System Administrator

You can manually stop any running process from the System Status form. If you want the server to automatically stop any process after a pre-defined period of inactivity, use the Start Inactivity Timer form.

To unlock a deadlocked resource:

1. Click **System Status** > **Show Locks**.
2. Review the list of system locks, and locate the deadlocked resource.
3. To stop a process, type **k** in the Command field.
4. Click **Execute Commands**.

   Service Manager stops the selected process.

   Note: Service Manager stops the process without issuing a warning.

Scheduled processes

By default, Service Manager provides the following scheduled processes:

Scheduler Name	Runs as Process Name or Names	Description
KMUpdate	KMUpdate	This process checks for update records and sends them to the index
SLA	sla	This process monitors service level agreements and processes alerts.
Sync	sync	This process identifies orphan processes that were shut down improperly and frees up any resources they are using before stopping them.
agent	agent	This process manages charts from stored queries.
alert.processor	alert	This process manages Incident Management log messages and alerts.
availability.startup	availability	This process manages asset availability and updates outage statistics.
change.startup	change	This process manages change events.
contract	contract	This process manages contract events.
event.startup	event	This process manages asynchronous incoming events from

Scheduler Name	Runs as Process Name or Names	Description
		the Eventin Queue.
gie.startup	gie	This is a legacy process from earlier versions of Service Manager that manages incoming Prim Events.
inactive.startup	inactive	If using the SM applications earlier than version 9.32, this process automatically logs off users who are inactive for a period of time. If using the SM applications version 9.32 or later, this process is obsolete. A new inactivity timer mechanism is used, which no longer requires this process. Note that this new mechanism requires your SM server, applications, and web client to upgrade to version 9.32 or later. As long as your applications is earlier than version 9.32, the old mechanism is used.
linker.startup	linker	This process manages related calls to closed incidents.
lister.startup	lister	This process manages global variables in the *globallist* file.
marquee	marquee	This process manages static marquee messages.
ocm.startup	ocm	This process manages request events.
printer.startup	despooler	This process manages server-side print requests.
problem	problem	This process manages IM alerts and messages.
report.startup	report	This process manages Report Writer reports and importing and exporting of reports.
scauto.startup	scautod	This process manages incoming and outgoing HP ServiceCenter Automate connections.

Shared memory

HP Service Manager shared memory is a pool of memory available to all processes on the server. You can control how the server allocates and manages shared memory using system parameters in the Service Manager initialization file. Allocating too little shared memory can degrade system performance, and allocating too much wastes system resources.

The shared memory storage report generates a snapshot of shared memory that enables you to monitor usage. If you are trying to isolate a performance bottleneck, run a report at start-up and periodically throughout the monitoring period. As a general rule, if Unused falls below 25%, consider increasing the amount of shared memory (see Startup parameters: shared_memory). If you are not sure how to interpret the shared memory report, contact HP customer support.

To run the shared memory report for a Unix or Windows system, do the following.

- Issue the following command from the shell prompt:

  ```
  sm -reportshm
  ```

Automatic selection of shared memory address

The Service Manager server can now automatically select a shared memory address based on your system's available memory. This change makes allocating shared memory easier on Windows systems because you no longer have to specify one particular address for your system's shared memory. The server automatically scans your system memory and selects an address range large enough to support your system's shared memory. If for some reason your system has insufficient free memory, the server will generate an error message telling you so.

The server will automatically select a shared memory address on Windows systems as long as you do not specify an address with the shared_memory_address parameter. On all Unix-based systems, the server uses the default values of the shared_memory_address parameter based on the operating system. If you previously specified a value for the shared_memory_address parameter, you can remove it from the sm.ini file to have your system automatically select a shared memory address.

Tip: HP recommends you remove the shared_memory_address parameter on all Windows platforms so that the system automatically selects your shared memory address. This minimizes the chance that your system will fail to start due to a memory collision with a system resource assigned by Windows ASLR (Address space layout randomization).

Example: response time monitoring (RTM) output

The following sections show example output from each Response Time Monitoring setting.

RTM:2

```
248(  6112) 10/21/2013 11:09:59  RTE D Response-Total: 0.016 -- RAD:0.016  JS:0.000
Log:0.000 Database:0.000  LoadBalancer: 0.000 (CPU 0.781) Transaction -
format:wizard-wiz.prompt.patchrel.unl application:display,show.rio option:0
248(  6112) 10/21/2013 11:09:59  RTE D -Memory : D(261304) S(4011264) O(1212028)
MAX(5387068) - MALLOC's Transaction(17425) Total(397602)
```

The debugging output includes:

- **Process ID** — lists the process identification number for the transaction
- **Thread ID** — lists the thread identification number for the transaction
- **Date** — lists the date the transaction occurred
- **Time** — lists the time the transaction occurred
- **Response -Total** — lists the number of seconds taken to complete the entire transaction
- **RAD** — lists the number of seconds taken for the RAD to complete the transaction
- **JS** — lists the number of seconds taken for the JavaScript to complete the transaction
- **Log** — lists the number of seconds taken for the log to complete the transaction
- **Database** — lists the number of seconds taken for the database to complete the transaction
- **LoadBalancer** — lists the number of seconds taken for the load balancer to complete the transaction
- **(CPU n)** — lists the cumulative number of seconds the process used on the CPU
- **Transaction - format** — lists the form displayed by the transaction
- **Application** — lists the application when the transaction is completed
- **Option ID** — lists the GUI option or button ID number used in this transaction

- **D** — lists the memory delta from the previous transaction. This value could be positive or negative.
- **S** — lists the storage (in bytes) currently allocated to this session by Service Manager. This does not include shared memory or memory that is not directly allocated by Service Manager.
- **O** — lists the overhead associated with the internal memory manager.
- **MAX** — lists the peak value of the storage (in bytes) allocated to this session by Service Manager.
- **MALLOC's Transaction** — lists the delta number of times of HP Service Manager called MALLOC to allocate memory from the previous transaction
- **Total** — lists the total number of times of HP Service Manager called MALLOC to allocate memory

RTM:3

```
248(   364) 10/21/2013 11:06:04  RTE D RADTRACE   2781 [ 1] display
           show.rio                  rio          CPU(    0    874 )
248(   364) 10/21/2013 11:06:04  RTE D Response-Total: 3.047 -- RAD:3.031  JS:0.000
Log:0.000 Database:0.016  LoadBalancer: 0.000 (CPU 0.874) Transaction -
format:wizard-wiz.prompt.patchrel.unl application:display,show.rio option:0
248(   364) 10/21/2013 11:06:04  RTE D -Memory : D(272304) S(3343528) O(945684) MAX
(4440604) - MALLOC's Transaction(18643) Total(409020)
```

The debugging output includes:

- **Process ID** — lists the process identification number for the transaction
- **Thread ID** — lists the thread identification number for the transaction
- **Date** — lists the date the transaction occurred
- **Time** — lists the time the transaction occurred
- **RADTRACE #** — lists the cumulative elapsed time for the transaction in milliseconds
- **[n]** — lists the thread number of the process
- **RAD application** — lists the RAD application used in the transaction
- **RAD panel** — lists the name of the RAD panel

- **CPU (n m)** — lists the number of milliseconds that the CPU processed the RAD panel (n) and the total number of milliseconds since the RAD trace began (m)
- **Response -Total** — lists the number of seconds taken to complete the entire transaction
- **RAD** — lists the number of seconds taken for the RAD to complete the transaction
- **JS** — lists the number of seconds taken for the JavaScript to complete the transaction
- **Log** — lists the number of seconds taken for the log to complete the transaction
- **Database** — lists the number of seconds taken for the database to complete the transaction
- **LoadBalancer** — lists the number of seconds taken for the load balancer to complete the transaction
- **(CPU n)** — lists the cumulative number of seconds the process used on the CPU
- **Transaction - format** — lists the form displayed by the transaction
- **Application** — lists the application when the transaction is completed
- **Option ID** — lists the GUI option or button ID number used in this transaction
- **D** — lists the memory delta from the previous transaction. This value could be positive or negative.
- **S** — lists the 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** — lists the overhead associated with the internal memory manager.
- **MAX** — lists the peak value of the number of bytes of storage allocated by Service Manager.
- **MALLOC's Transaction** — lists the delta number of times of HP Service Manager called MALLOC to allocate memory from the previous transaction
- **Total** — lists the total number of times of HP Service Manager called MALLOC to allocate memory

RTM:4

```
3688(  5512) 10/21/2013 17:21:25  RTE D RADTRACE    219 [ 1] display
             show.rio                  rio          CPU(    0   3108 )
3688(  5512) 10/21/2013 17:21:25  RTE D Response-Total: 0.328 -- RAD:0.187
JS:0.000 Log:0.125 Database:0.141  LoadBalancer: 0.000 (CPU 3.155) Transaction -
```

```
format:wizard-wiz.prompt.patchrel.unl application:display,show.rio option:0
3688(  5512) 10/21/2013 17:21:25  RTE D -Memory : D(272624) S(3341592) O(943524)
MAX(4440604) - MALLOC's Transaction(54547) Total(879376)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 88   calls made to ';' (1)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 252  calls made to '=' (2)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 7    calls made to ' or ' (3)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 38   calls made to ' and ' (4)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 16   calls made to 'not ' (5)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to '<=' (7)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 47   calls made to '=' (8)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 17   calls made to '~=' (9)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 10   calls made to '>' (11)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 72   calls made to '+' (12)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 61   calls made to 'if ' (19)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 2    calls made to 'if ' (20)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 8    calls made to 'for ' (22)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 834  calls made to ' in ' (23)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 32   calls made to 'decision' (28)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 29   calls made to 'process' (42)
3688  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'rio' (48)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 5    calls made to 'select' (50)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 23   calls made to 'str' (51)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 6    calls made to 'userdefn' (53)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 9    calls made to 'lng' (54)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 14   calls made to 'filename' (55)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'operator' (56)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 11   calls made to 'rinit' (58)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 17   calls made to 'currec' (59)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 11   calls made to '#' (60)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'val' (72)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'index' (78)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 29   calls made to 'null' (88)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 7    calls made to 'denull' (113)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'contents' (117)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'loop' (119)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 17   calls made to 'parse' (122)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 39   calls made to 'evaluate' (123)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 12   calls made to 'type' (124)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 67   calls made to 'nullsub' (125)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 2    calls made to 'cleanup' (127)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 8    calls made to '+=' (128)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 8    calls made to 'same' (133)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 6    calls made to 'param2' (134)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 4    calls made to 'gui' (155)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 1    calls made to 'sysinfo.get' (169)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 9    calls made to 'strrep' (188)
3688(  5512) 10/21/2013 17:21:25  RTE D ... 14   calls made to 'scmsg' (238)
```

The debugging output includes:

- **Process ID** — lists the process identification number for the transaction
- **Thread ID** — lists the thread identification number for the transaction
- **Date** — lists the date the transaction occurred
- **Time** — lists the time the transaction occurred
- **RADTRACE #** — lists the cumulative elapsed time for the transaction in milliseconds
- **[n]** — lists the thread number of the process
- **RAD application** — lists the RAD application used in the transaction
- **RAD panel** — lists the name of the RAD panel
- **CPU (n m)** — lists the number of milliseconds that the CPU processed the RAD panel (n) and the total number of milliseconds since the RAD trace began (m)
- **Response -Total** — lists the number of seconds taken to complete the entire transaction
- **RAD** — lists the number of seconds taken for the RAD to complete the transaction
- **JS** — lists the number of seconds taken for the JavaScript to complete the transaction
- **Log** — lists the number of seconds taken for the log to complete the transaction
- **Database** — lists the number of seconds taken for the database to complete the transaction
- **LoadBalancer** — lists the number of seconds taken for the load balancer to complete the transaction
- **(CPU n)** — lists the cumulative number of seconds the process used on the CPU
- **Transaction - format** — lists the form displayed by the transaction
- **Application** — lists the application when the transaction is completed
- **Option ID** — lists the GUI option or button ID number used in this transaction
- **D** — lists the memory delta from the previous transaction. This value could be positive or negative.
- **S** — lists the 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** — lists the overhead associated with the internal memory manager.
- **MAX** — lists the peak value of the S
- **MALLOC's Transaction** — lists the delta number of times of HP Service Manager called MALLOC to allocate memory from the previous transaction
- **Total** — lists the total number of times of HP Service Manager called MALLOC to allocate memory
- **# of calls made to 'x'** — lists the number of times HP Service Manager called the function, panel, or operator x from the last format response to this one
- **(#)** — lists the function identification number for the function, panel, or operator from the last format response to this one

RTM:5

```
3688(  3952) 10/21/2013 17:22:45  RTE D RADTRACE    203 [ 1] display
             show.rio                  rio          CPU(    0   2827 )
3688(  3952) 10/21/2013 17:22:45  RTE D Response-Total: 0.281 -- RAD:0.281
JS:0.000 Log:0.265 Database:0.000  LoadBalancer: 0.000 (CPU 2.843) Transaction -
format:wizard-wiz.prompt.patchrel.unl application:display,show.rio option:0
3688(  3952) 10/21/2013 17:22:45  RTE D -Memory : D(261512) S(3330480) O(950540)
MAX(4440604) - MALLOC's Transaction(54063) Total(878892)
3688(  3952) 10/21/2013 17:22:45  RTE D ... 595  calls made to ' in ' (23)
3688(  3952) 10/21/2013 17:22:45  RTE D ... 1    calls made to 'rio' (48)
3688(  3952) 10/21/2013 17:22:45  RTE D ... 1    calls made to 'evaluate' (123)
3688(  3952) 10/21/2013 17:22:45  RTE D ... 2    calls made to 'nullsub' (125)
3688(  3952) 10/21/2013 17:22:45  RTE D ... 2    calls made to 'gui' (155)
```

The debugging output includes:

- **Process ID** — lists the process identification number for the transaction
- **Thread ID** — lists the thread identification number for the transaction
- **Date** — lists the date the transaction occurred
- **Time** — lists the time the transaction occurred
- **RADTRACE #** — lists the cumulative elapsed time for the transaction in milliseconds
- **[n]** — lists the thread number of the process

- **RAD application** — lists the RAD application used in the transaction
- **RAD panel** — lists the name of the RAD panel
- **CPU (n m)** — lists the number of milliseconds that the CPU processed the RAD panel (n) and the total number of milliseconds since the RAD trace began (m)
- **Response -Total** — lists the number of seconds taken to complete the entire transaction
- **RAD** — lists the number of seconds taken for the RAD to complete the transaction
- **JS** — lists the number of seconds taken for the JavaScript to complete the transaction
- **Log** — lists the number of seconds taken for the log to complete the transaction
- **Database** — lists the number of seconds taken for the database to complete the transaction
- **LoadBalancer** — lists the number of seconds taken for the load balancer to complete the transaction
- **(CPU n)** — lists the cumulative number of seconds the process used on the CPU
- **Transaction - format** — lists the form displayed by the transaction
- **Application** — lists the application when the transaction is completed
- **Option ID** — lists the GUI option or button ID number used in this transaction
- **D** — lists the memory delta from the previous transaction. This value could be positive or negative.
- **S** — lists the 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** — lists the overhead associated with the internal memory manager.
- **MAX** — lists the peak value of the S
- **MALLOC's Transaction** — lists the delta number of times of HP Service Manager called MALLOC to allocate memory from the previous transaction
- **Total** — lists the total number of times of HP Service Manager called MALLOC to allocate memory
- **# of calls made to 'x'** — lists the number of times HP Service Manager called the function, panel, or operator x from the last panel response to this one

- **(#)** — lists the function identification number for the function, panel, or operator from the last panel response to this one

Example: semaphore reports

The following illustrates the results of semaphore commands.

sm -reportsem

```
C:\Program Files\HP\Service Manager x.xx\Server\RUN>sm -reportsem
03/09/11 10:15:11   pid (4940) HP Service Manager diagnostic report follows:
--- Reportsem ---

[0]System               Available
count(2)
[1]Application cache    Available
count(708)
[2]Shared memory        Available
count(5416)
[3]IR Expert            Available
count(2)
[4]Licensing            Available
count(2)
[5]Resource manager     Available
count(0)
[6]User chain           Available
count(211)
[7]Cache manager        Available
count(8829)
[8]Database Services    Available
count(82)
[9]Alert Services       Available
count(0)
[10]Counter Services    Available
count(0)
[11]Diagnostics Service Available
count(5861)
```

Debug a suspended or hung system

Applies to User Roles:

System Administrator

If HP Service Manager is unresponsive or appears hung, you can use the reportsem command to determine if any semaphores are locked.

To run the semaphore report type the following command from a shell prompt (Unix) or a command prompt (Windows):

```
sm -reportsem
```

View shared memory statistics

Applies to User Roles:

System Administrator

To view shared memory statistics:

1. Click **System Status** in the System Navigator. The system.status.list.g form opens.
2. From the System Status form, click **System Monitor**. The system.monitor.1.g form opens.
3. From the System Monitor - Main User Info form, click **Shared Memory Info**.

   HP Service Manager displays shared memory statistics of the server:

   - Total physical memory available
   - Segment allocation
   - Large block allocation
   - Unused disk space
   - Percentage of free space
   - Server memory caches and processes
     - Type
     - Allocations
     - Frees
     - Allocated

Shared memory storage report

In the shared memory storage report, the top “Shared memory” section summarizes shared memory allocation. The bottom “By type” section details areas of the system that use shared memory.

Shared memory

Shared Memory	Description
Current size	The current size of shared memory as specified in the sm.ini parameter file.
Occupied space	The amount of shared memory occupied by allocations less than 32K in size.
Big alloc space	The amount of shared memory occupied by allocations greater than 32K in size.
Unused	The amount of shared memory that has never been used.

By type

Allocation	Description
Not named	System overhead.
User blocks	HP Service Manager allocates a storage block for each logged on user.
Messages	Messages sent from one user to another.
Resource locks	Record locks.
Cache overhead	Overhead for the five cache categories (Application cache, DBDICT, Format, SQL descriptor, IR Expert).
Application cache	(Cache category.) RAD applications that have been read by users and that are in use.
DBDICT cache	(Cache category.) Database definitions.
SQL descriptor cache	(Cache category.) SQL definitions.
Join/ERD/Type cache	ERD and JOINDEFS definitions.
String Type Cache	Area in which formats, links and other definitions are stored.

Allocation	Description
IR Expert cache	(Cache category.) IR Expert data.

Example: Shared Memory report

The following sections illustrates the results of the **sm -reportshm** command:

```
02/10/04 16:16:01   pid (215) HP Service Manager diagnostic report follows:

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

Shared Memory Release     6.0

Current Size              32000000

Segment Allocation         4755184
Large Block Allocation     2322176

Unused Space              24922640  (77%)
Free Space                25243264  (78%)

Shared Memory Type  Allocations    Frees     Allocated
------------------  ----------- ----------- -----------
Not named                    76          55        4304
User blocks                  84          52        8448
Messages                      0           0           0
Resource locks           520654      520625        1856
Database Services          1081         830       59840
Cache overhead                3           0       10272
Application cache          2755           0     4252048
DBDICT cache               3308        2554     2271232
SQL descriptor cache          0           0           0
Join/ERD/Type cache         326           0       43200
Remote DBDICT cache           0           0           0
Remote Record cache           0           0           0
String Type cache           702         608       10208
IR Expert cache             372           0       90304
Publish/Subscribe          4469        4368        4512
24x7 cache                    1           0         512
Web cache                     0           0           0
```

Type statistics

The following are definitions for the type statistics of a shared memory report.

Type	Definition
Allocations	The number of times that shared memory for this type has been allocated.
Frees	The number of times that shared memory for this type has been freed.
Allocated	The amount in bytes that is currently allocated for this type.

Shared memory guide

The *HP Service Manager Shared Memory guide* has information about creating and using shared memory. The following information is included in this guide:

- Introduction to shared memory
- Creation and use of shared memory
- Shared memory sizing, including:
  - Cache sizing
  - Shared memory reports
  - Shared memory areas
  - IR Expert and shared memory
- Horizontal scaling - special considerations
- Troubleshooting

You can view and search this guide using Adobe® Reader, which you can download from the Adobe Web site.

The *HP Service Manage Shared Memory Guide* is available from the help.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Server performance tuning help topics for printing (Service Manager 9.41)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to ovdoc-ITSM@hp.com.

We appreciate your feedback!


Java™
COMPATIBLE