

IBM Cúram Social Program Management
Version 7.0.10

Performance tuning



Note

Before using this information and the product it supports, read the information in [“Notices” on page 24](#)

Edition

This edition applies to IBM® Cúram Social Program Management v7.0.10 and to all subsequent releases unless otherwise indicated in new editions.

Licensed Materials - Property of IBM.

© **Copyright International Business Machines Corporation 2014, 2020.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

- Tables..... iv**
- Chapter 1. Performance tuning..... 1**
- Chapter 2. Prerequisites for performance tuning..... 2**
- Chapter 3. System configuration assumptions..... 3**
- Chapter 4. Database configuration..... 4**
 - Db2..... 4
 - Oracle..... 5
- Chapter 5. Social Program Management application..... 9**
 - Static content..... 9
 - Configuring the HTTP key server..... 9
 - ID generation..... 9
 - Caches..... 9
 - Login..... 10
 - Batch processes..... 10
- Chapter 6. Application server..... 11**
 - Number of application servers..... 11
 - JVM settings..... 11
 - Thread pools..... 12
 - JDBC connection pool settings..... 13
 - JMS settings..... 14
 - WAS Java 2 security..... 15
 - Monitoring the application server..... 16
 - WAS tunable parameters summary..... 17
- Chapter 7. Tuning the HTTP server..... 21**
 - Configuring the HTTP server for static content..... 21
 - Compressing content from HTTP server..... 21
 - Monitoring the HTTP server..... 21
 - Configuring persistent connections..... 22
 - Adjusting thread tuning..... 22
- Notices..... 24**
 - Privacy Policy considerations..... 25
 - Trademarks..... 25

Tables

1. Oracle tuning parameters.....	6
2. Recommended cache sizes.....	10
3. Summary of tunable parameters for WAS.....	17

Chapter 1. Performance tuning

This guide provides a quick-start and guidelines to tuning IBM Cúram Social Program Management (SPM) in a production or production-like environment. The guide covers the key tuning parameters for a system deployment of SPM, and provides formulas to use for some parameters and starter values for others.

Scope

The following turning tasks are out of scope:

- Database tuning
- Operating system tuning
- Network tuning
- Disk subsystem tuning

While the starter values are based on experience and are sensible for production, it is unlikely that they are the optimal values for your specific system. You must further tune the starter values during your load testing and production monitoring. Tuning testing might be helpful for evaluating the initial starter values. Tuning tests consist of running short simulations of a workload representative of production volume and hardware to find the optimum tuning values for the system. The production volume might be simulated by using your load test workload. The tests do not normally use think times and need to last only half an hour. During these tests, perform full system monitoring. This type of testing allows a short cycle of test, feedback, and update to tuning parameters. Tuning tests are specialized load tests and are not a direct replacement for load tests. The goal of tuning tests is only to improve the system performance through tuning.

Audience

This guide is intended for a technical audience who is deploying, testing, or running IBM Cúram Social Program Management in a production or production-like environment.

Chapter 2. Prerequisites for performance tuning

Before you undertake IBM Cúram Social Program Management tuning activities, for example, configuring SPM for a specific deployment, obtain an understanding of SPM and the requisite middleware such as IBM WebSphere® Application Server and IBM Db2®.

To ensure you are using supported software with the correct versions, refer to the [IBM Cúram Social Program Management Supported Prerequisites](#). In addition to the product prerequisites, consult the system requirement documents for each of the required/optional software components.

For more information about IBM Cúram Social Program Management, see https://www.ibm.com/support/knowledgecenter/SS8S5A_7.0.10/com.ibm.curam.nav.doc/spm_welcome.html.

Chapter 3. System configuration assumptions

Specific values are provided as recommended initial values for the performance tuning process, and the performance tuning examples that are provided assume certain configuration settings.

The values you identify from your observation and tuning activities might vary, based on a number of factors:

- Distribution of workload
- Number of processor cores
- Speed of the processor cores
- Performance of the I/O system

The provided examples assume the following configuration settings:

- Each application server runs its own messaging engine, as for a high-availability environment.
- Each server uses a dedicated thread pool that is configured for online processing, and another pool for asynchronous processing.

For more information about configuration details, see the [Cúram Social Program Management Clustered IBM WebSphere Application Server Deployment Guide](#), or the [Cúram Social Program Management Clustered Oracle WebLogic Server Deployment Guide](#), which is available on request.

Chapter 4. Database configuration

Database tuning is out of scope. However, the following suggestions for how to configure a database for high performance with IBM Cúram Social Program Management are based on prior experience. The suggestions are not exhaustive, and tuning before production is strongly recommended.

Db2

The following maintenance and database design guidelines apply to tuning Db2 databases.

Maintenance

Regular database maintenance is essential to ensure optimal performance and reliability of your databases. Running the REORGs and RUNSTATS utilities is critically important for optimal performance with Db2 databases. After the database is populated, do the maintenance on a regularly scheduled basis, such as weekly. A regularly scheduled maintenance plan is essential to maintain peak performance of your system.

Physical database design

In addition to physical database design done as part of the project, the following tips apply specifically to performance and scalability:

- Create two 4 K page buffer pools, one for data and one for indexes, and a 4 K page temporary table space.
- Create a 4 K page large table space for data, with no file system caching.
- Create a 4 K page large table space for indexes, with no file system caching.
- Create a 4 K page large table space for LOBs, with file system caching (to enable buffering of Large Object types (LOBs)).
- Move all indexes to the 4 K page indexes table space. A common page size makes database administration easier and improves performance. For example, indexes of tables in the 32 K page table space that use 4 K pages require less I/O.
- Move all LOBs to the 4 K page LOBs table space.
- Move all tables that do not need 32 K page to the 4 K page data table space. It can improve buffer pool utilization and decrease database I/O.
- Enable compression on tables if it is wanted. Compression was tested by IBM with no issues.
- Review the use of dedicated table spaces. Consider moving the largest tables (for example, DYNAMIC EVIDENCEDATAATTRIBUTE and DEPENDENCY) to dedicated table spaces. However, from experience, with good I/O layouts, the only reason for dedicated table spaces is maintenance.
- Review the use of partitioning. From experience, with good I/O layouts, the only reason for partitioning is maintenance (similar to dedicated table spaces).

Registry variables

To improve the buffer pool management and increase performance in Db2, you can set the following Db2 registry variables:

```
DB2_USE_ALTERNATE_PAGE_CLEANING=ON
```

Setting `DB2_PARALLEL_IO=* : n` enables Db2 to parallelize IO to that logical volume. The number of disks in the RAID array that backs a logical volume is represented by “n”.

The following maintenance and database design guidelines apply to tuning Oracle databases.

Initialization and tuning parameters for Oracle databases

The application server for IBM Cúram Social Program Management uses JDBC connection pooling and prepared statement caching. At bare minimum, review and adjust the **PROCESSES** and **OPEN_CURSORS** parameters from the following list according to your system requirements. You can use the following guidance when you are deciding which values to apply for **PROCESSES** and **OPEN_CURSORS** on your system. In addition to these two basic parameters, check the parameters that are referenced in the following table, and adjust the parameters if necessary.

PROCESSES

PROCESSES specifies the maximum number of operating system user processes that can simultaneously connect to Oracle. Its value must allow for all background processes such as locks, job queue processes, and parallel execution processes. This parameter is operating system dependent. The default values of the **SESSIONS** and **TRANSACTIONS** parameters are derived from this parameter. If you change the value of **PROCESSES**, evaluate whether to adjust the values of those derived parameters. You can use the following guidance to set a value for **PROCESSES**:

```
PROCESSES > MAX
```

where

```
MAX=NUMBER OF APPLICATION SERVERS * NUMBER OF POOLED JDBC CONNECTIONS
```

or

```
MAX=NUMBER OF BATCH STREAMS * 3
```

Note: Increasing the number of processes leads to greater memory consumption. Consult with your database administrator before you make any changes.

OPEN_CURSORS

Oracle creates a memory area (context area) for processing SQL statements. The value of **OPEN_CURSORS** must be set high enough to prevent your application from running out of open cursors. Assuming that a session does not open the number of cursors that are specified by **OPEN_CURSORS**, there is no added overhead to setting this value higher than needed. The following guidance can be used to set a value for **OPEN_CURSORS**:

```
OPEN_CURSORS >= NUMBER OF OVERALL EXPECTED DBCONNECTIONS
```

or

```
OPEN_CURSORS > JDBC/CURAMDB PREPARED STATEMENT CACHE SIZE
```

SESSIONS

This parameter modifies the number of sessions that are allowed by Oracle at database level.

Note: Setting the number of allowed processes automatically sets the appropriate number of sessions, which is the recommended way of tuning this parameter. Oracle does not recommend setting this parameter explicitly.

EVENTS

This parameter is used to reduce contention that is caused by LOB management. To identify the number of chunks to be cleaned up each time a reclamation operation is performed, set EVENT 44951 to a value in the range 1 - 1024. In turn, this parameter reduces the number of requests against the high watermark enqueue.

DB_BLOCK_SIZE

This parameter controls the size of the default data block size at the time of database creation. If the database exists, this value cannot be changed: a new database needs to be created with a different block size. Since Oracle 10g Release 2, each table space can have a different block size. However, the block size still needs to be chosen wisely. Performance tests showed that decreasing the block size decreases threads contention. The **DB_BLOCK_SIZE** default value is 8192. The value of this parameter must be a multiple of the physical block size at the device level. Our suggestion is to set the parameter to the lowest possible value.

DISK_ASYNC_IO

Much CPU time can be wasted on asynchronous I/O on systems that support it. We suggest setting the **DISK_ASYNC_IO** parameter to FALSE.

Note: With **DISK_ASYNC_IO** off, you should set the **DBWR_IO_SLAVES** parameter to a value other than its default of 0, to simulate asynchronous I/O.

DBWR_IO_SLAVES

The **DBWR_IO_SLAVES** parameter is relevant only on systems with only one database writer process (DBW0). It specifies the number of I/O server processes that are used by the DBW0 process. The DBW0 process and its server processes always write to disk. By default, the value is 0 and I/O server processes are not used. **DBWR_IO_SLAVES** should be set and tuned when simulating asynchronous I/O, for example, with **DISK_ASYNC_IO=FALSE**. Performance tests have shown that setting **DBWR_IO_SLAVES** to 20 yields good performance.

In addition to the parameters mentioned previously, a number of other important Oracle tuning parameters are available that you can review and adjust, if necessary. The following table summarizes the tuneable parameters.

Tuning parameter	Description and use	Recommended value
PROCESSES	Specifies the maximum number of operating system user processes that can simultaneously connect to Oracle	PROCESSES > MAX where MAX=NUMBER OF APPLICATION SERVERS * NUMBER OF POOLED JDBC CONNECTIONS or MAX=NUMBER OF BATCH STREAMS * 3
SESSIONS	Modifies the number of sessions that are allowed by Oracle at database level.	Note: Setting this parameter explicitly is not recommended.
OPEN_CURSORS	Controls the size of the memory area (context area) for processing SQL statements.	open_cursors >= number of overall expected dbconnections or open_cursors > jdbc/curamdb prepared statement cache size
EVENTS	Set to reduce contention caused by LOB management.	SET EVENTS = '44951 TRACE NAME CONTEXT FOREVER, LEVEL 1024'

Table 1. Oracle tuning parameters (continued)

Tuning parameter	Description and use	Recommended value
DB_BLOCK_SIZE	Controls the size of the default data block size at the time of database creation.	Set to the lowest possible value (2048): DB_BLOCK_SIZE = 2048
FAST_START_MTTR_TARGET	Specifies the number of seconds that the database takes to perform crash recovery of a single instance. FAST_START_MTTR_TARGET=60 for approximately 100 MB files. FAST_START_MTTR_TARGET=300 for approximately 500 MB files. Check the size of log files with: select * from V\$LOG	FAST_START_MTTR_TARGET=300
DISK_ASYNC_IO	Controls whether I/O to data files, control files, and log files is asynchronous.	DISK_ASYNC_IO=false
DBWR_IO_SLAVES	Specifies the number of I/O server processes that are used by the DBW0 process.	DBWR_IO_SLAVES = 20
DB_WRITER_PROCESSES	DB_WRITER_PROCESSES (DBWR) manages the "dirty block" cleanouts from the data buffer. Very few tuning options exist except adjusting the number of DBWR processes. DB_WRITER_PROCESSES cannot exceed a value of 20.	The default value is the larger value of either: DB_WRITER_PROCESSES = 1 or CPU_COUNT/8
IDLE_TIME	Set to greater or equal to the timeout that is set on WebSphere Application Server.	ALTER PROFILE DEFAULT LIMIT IDLE_TIME n Set the value to greater than or equal to the WebSphere Application Server timeout.
LOG_BUFFER	Specifies the amount of memory in bytes that Oracle uses when buffering redo entries to a redo log file. The database chooses an appropriate default based on the server specification. In our tests, we set this parameter with positive results for performance. Consult your database administrator to determine whether setting this parameter is appropriate in your environment.	Note: In newer releases of Oracle, the advice is do not set the LOG_BUFFER parameter.

Table 1. Oracle tuning parameters (continued)

Tuning parameter	Description and use	Recommended value
DB_CACHE_SIZE	Use the DB cache advisor (V \$DB_CACHE_ADVICE view) to see whether any benefit can be gained from increasing the size of the buffer cache. Consult your database administrator to determine whether setting this parameter is appropriate in your environment.	No recommendation because the parameter is system-dependent.
TABLESPACE DATAFILES	More data files results in less write contention. Consult your database administrator to determine whether setting this parameter is appropriate in your environment.	No recommendation because the parameter is system-dependent.

Physical database design

In addition to physical database design that is done as part of the project, experience shows that the database design has benefits in reducing contention on hot index blocks (cache buffer chain). To achieve this benefit, create a 2 K block table space for indexes, and move indexes to that 2 K block index table space.

Statistics

The database optimizer relies on database statistics to determine which indexes are used for data access. If the database statistics are not up-to-date, the correct indexes might not be used. In this case, the performance of the system degrades, and at worst the system becomes unstable. It is important to ensure that database statistics are run and gathered frequently. The general convention is that database statistics are gathered on a table when 10% or more of the data on that table changes. Typical examples in SPM are to run statistic in the early life of the system and after some batch job executions.

More specifically, initially many tables in the SPM database are empty or have a low number of rows. Therefore, before you turn on a production or test system, you must run and gather database statistics. During the first hours and days that the system is used, run and gather database statistics often, at least daily.

Chapter 5. Social Program Management application

Use the following guidelines for basic tuning of the Social Program Management application.

Static content

An application server is optimized to serve dynamic content, while an HTTP server is optimized to serve static content. Package the IBM Cúram Social Program Management (SPM) static content and copy it to the HTTP Server.

To enable static content in SPM, set the `STATIC_CONTENT_SERVER` element in the `curam-config.xml` file and perform a full SPM build.

For information about configuring the web server for static content, see the section [“Configuring the HTTP server for static content”](#) on page 21.

Configuring the HTTP key server

For higher performance, consider setting `humanReadable=1` for all entries in the `KeyServer.dmx` file.

Note: If you set `humanReadable=1`, key allocation becomes sequential and therefore predictable. You must consider the use of this parameter in conjunction with your security requirements.

Note: Some customers have reported that when they use `humanReadable=1` and Oracle RAC, concurrent inserts into the same table can cause heavy contention on index blocks. A solution to this problem is to re-create the primary key index of the affected tables to be REVERSE KEY. However, using a REVERSE KEY index means that Oracle cannot perform an index range scan on that index. This should not be a problem for primary key indexes as we mostly use surrogate keys, which have no business meaning.

ID generation

If the default ID generation algorithms are used, change them to use the key server to prevent contention.

Set the value of the following application properties to YES:

```
curam.referencenumber.generateproviderreferencenumberfromkeyset  
curam.referencenumber.generateprovidergroupreferencenumberfromkeyset  
curam.referencenumber.generateexternalpartyidfromkeyset  
curam.referencenumber.generateemployeridfromkeyset  
curam.referencenumber.generatepersonidfromkeyset  
curam.referencenumber.generateinformationproviderridfromkeyset  
curam.referencenumber.generateutilityidfromkeyset  
curam.referencenumber.generateservicesupplieridfromkeyset  
curam.referencenumber.generateproductprovideridfromkeyset  
curam.referencenumber.generaterepresentativeidfromkeyset
```

Caches

For better performance, you can resize caches.

To improve performance, resize the following caches to the values that are indicated in the table. Monitor the caches and increase their size if evictions occur.

Cache	Recommended value
<code>curam.cache.curam-group.componentModelCache.size</code>	100000
<code>curam.cache.curam-group.CREOLERuleSetDtIsCache.size</code>	Higher than the number of rule sets that are stored in the CREOLERuleSet table
<code>curam.cache.curam-group.RuleSetSnapshotIDCache.size</code>	Set the size to the largest of the following two values: <ul style="list-style-type: none"> • 3 times the number of rule sets that are stored in the CREOLERuleSet table • The number of snapshots that are stored in the CREOLERuleSetSnapshot table

Login

The first login on a cold application server is usually slower than subsequent logins, and can exceed the application server transaction timeout. To avoid slow first logins, you can set a custom value for the `LoginBeanTransaction.transaction.timeout` property.

Set the value of the `LoginBeanTransaction.transaction.timeout` property to greater than the application server transaction timeout. When the value is set, the value overrides the application server transaction timeout for the login transaction.

Batch processes

Use the following guidelines to tune the performance of batch processes.

Batch chunk size

For streamed batch jobs, to offer good balance between scalability and error management, tune the chunk size for a batch job so that it takes between one to two minutes to process one chunk.

Db2

When Db2 is used, set the following tables to be volatile:

```
BATCHPROCESS
BATCHPROCESSCHUNK
```

Oracle

When Oracle is used, enable Oracle connection and statement caching in the `bootstrap.properties` file:

```
curam.db.oracle.connectioncache.enabled=true
curam.db.oracle.connectioncache.minlimit=3
curam.db.oracle.connectioncache.maxlimit=3
curam.db.oracle.connectioncache.initiallimit=3
curam.db.oracle.cachesize=1000
```

Chapter 6. Application server

The following section indicates starting values for application server tuning parameters. The parameters are key to the performance of Social Program Management on applications servers such as WebSphere Application Server (WAS) and WebLogic Server (WLS). Refine the values during load testing of the system and during monitoring of production. A load test phase in the project is highly recommended.

Number of application servers

Determine the number of application servers in your application clusters that are based on availability requirements. An initial starting point is to have at least two application servers per cluster. However, it is recommended to use three or more application servers per cluster to ensure good performance and quality of service in case of server failure.

Another consideration for the number of application servers is the number of cores that are available from the operating system. In general, it is recommended to keep the number of threads in a Social Program Management application server to a low multiplier of the number of cores, as described in the “[Thread pools](#)” on [page 12](#) section. However, a high number of cores can result in a very high number of threads and, therefore, a higher probability of threads contending on Java locks. In that case, it is advisable to split the threads across multiple application servers instead of just one, which reduces the probability of contention. However, a drawback of this approach is increased memory utilization on the operating system as multiple JVMs are needed.

JVM settings

Because of the memory requirements in Social Program Management, either a 64-bit WebSphere Application Server (WAS) or WebLogic Server (WLS) application server is required for production.

For the SPM internal application and the portals that are deployed in their own cluster, start with the following settings:

```
minimum heap size = maximum heap size = 4GB
```

HotSpot JVM

- Size the Permanent Generation to at least 512 MB
- Set the Nursery size to about two-thirds of the Heap size

GSS and BirtViewer

- Start with Minimum Heap Size = Maximum Heap Size = 1GB

Then, use the following heuristic to further tune heap sizes during your load tests, with an objective of getting above 98% garbage collection efficiency. Configure the following settings, where `avg_used_after_global` is the average amount of used heap after global collections:

```
-Xmx = 8 * avg_used_after_global  
-Xms = 8 * avg_used_after_global  
-Xmn = 6 * avg_used_after_global
```

The following common recommendations outline the rationale behind the previous heuristic:

- A recommendation for sizing the tenured generation is to have a free tenured utilization after global collection of around 50%. The tenure contains global and thread-scoped SPM caches, and any long-lived objects.
- A recommendation for sizing the nursery generation is to have a nursery as large as possible, if the pause times are acceptable. The nursery contains transient objects for in-flight SPM transactions, so its sizing depends on the number of concurrent transactions and the average memory payload of a

transaction. The goal is to minimize the number of transient objects that survive collections and end up in the tenure. We have found in our load tests that a 3:1 ratio of nursery to tenure gives good results.

Thread pools

For online HTTP requests and asynchronous JMS processing, IBM Cúram Social Program Management (SPM) is configured to use two distinct thread pools. Tune both pools as explained in the following sections.

To enable the system to be driven at optimum throughput, it is generally recommended that the total number of application threads are about twice the number of CPUs. This general convention assumes that the environment has fast I/O subsystems.

Note: Threads that are used by SPM carry caches and therefore impact memory requirements. In addition, the higher the number of threads, the more likely that contention occurs on Java™ locks. Therefore, it is recommended to limit the number of threads to a low multiplier of the number of cores.

WAS: WebContainer and SIBJMSRAThreadPool

In WebSphere Application Server (WAS), the `WebContainer` thread pool is set up to process HTTP requests and the `SIBJMSRAThreadPool` is set up to process JMS messages.

Use the following specification as a starting point: the total number of threads that SPM uses in the application server can be set to twice the number of available cores. This starting point assumes that only one application server is running on the operating system or logical partition. Setting the number of threads to twice the number of cores is based on experience that processing in SPM is usually split relatively equally between I/O and CPU.

The way processing is then broken down between online and asynchronous processing depends on the characteristics of your system: how much asynchronous processing does it do? As a quick-start, a simple, equal split is suggested:

```
WebContainer_max_threads = number of cores
SIBJMSRAThreadPool_max_threads = number of cores
```

It is suggested that you set the minimum number of threads equal to the maximum number of threads. Setting the minimum threads equal to the maximum threads avoids the processing cost of pool growth and shrinkage. Use the following settings to configure the pools:

```
WebContainer_min_threads=WebContainer_max_threads
SIBJMSRAThreadPool_min_threads=SIBJMSRAThreadPool_max_threads
```

Then, monitor the thread pool usage and the number of threads that are adjusted according to CPU utilization. For example, if a thread pool is fully used and spare CPU capacity exists, you can add a thread. Spare CPU capacity, depending on the platform, might be CPU use below 80%. You need to define your CPU plateau threshold based on your environment configuration and the results of load testing.

WLS: Maximum thread constraints

In WebLogic Server (WLS) configure the two work managers that SPM uses, which are the default work manager for HTTP requests and the `MDBWorkManager` for JMS. Specify a maximum thread constraint. As a starting point, set the maximum thread constraints to the following values:

```
default_max_thread_constraint = number of cores
MDBWorkManager_max_thread_constraint = number of cores
```

Then, tune the work managers by monitoring thread usage. Monitoring indicates thread usage for a work manager. If all threads are used and CPU capacity exists, you can increase the maximum thread constraint.

JDBC connection pool settings

Set the JDBC connection pools for the Social Program Management data sources.

Data source: jdbc/curamdb

A Social Program Management (SPM) transaction can require two JDBC connections, one for the transaction itself and another one for the key server. Size the `jdbc/curamdb` data source connection pool to prevent deadlocks, with more connections available than threads that SPM uses. Therefore, size the connection pool for the `jdbc/curamdb` data source by using the following formula:

WebSphere Application Server (WAS)

```
max_connections = WebContainer_max_threads + SIBJMSRThreadPool_max_threads + 1
```

WebLogic Server (WLS)

```
max_connections = default_max_thread_constraint + MDBWorkManager_max_thread_constraint + 1
```

If a firewall exists between the application servers and the database, to prevent issues that are related to `StaleConnectionException`, we recommend setting `min_connections = 0` and `reapTime<= unused connection timeout <= firewall timeout`.

As a starting value for SPM, increase the data source prepared statement cache (`jdbc/curamdb Statement Cache Size`) to 1000. Then, monitor the cache use and increase it if discards occur. In our experience, preventing discards can increase throughput by up to 20%.

Note: While we recommend an initial value of 1000 for the prepared statement cache to prevent discards, this value can be too high for SPM-based systems that have many threads and that are memory constrained. In that case, it is recommended to review the SQLStats from the JMX Stats, from either load tests or production. Then, use a simple heuristic based on the distribution of SQL executions to find a smaller cache size that covers around 90% of SQL executions from the application and gives a better balance between system performance and Java heap utilization. However, monitor prepared statement cache discards, system performance, and heap utilization, and adjust the cache size further as needed.

Data source: jdbc/curamsibdb

As all SPM transactions can potentially create a JMS message, size the connection pool for the `jdbc/curamsibdb` data source by using the following formula:

WAS

```
max_connections = WebContainer_max_threads + SIBJMSRThreadPool_max_threads + 1
```

WLS

```
max_connections = default_max_thread_constraint + MDBWorkManager_max_thread_constraint + 1
```

To prevent the processing cost of pool growth and shrinkage, it is suggested to set `min_connections = max_connections`.

If a firewall exists between the application servers and the database, to prevent issues that are related to `StaleConnectionException`, we recommend setting `min_connections = 0` and `reapTime<= unused connection timeout <= firewall timeout`.

Data source: jdbc/curamtimerdb

The EJB timer service is used by all SPM transactions, but only once per transaction, in our application infrastructure and at the very start of an SPM transaction. Currently, no reference to or usage of this service exists after the very start of an SPM transaction.

You can tune the size of the `jdbc/curamtimerdb` data source connection pool to be the same size as the number of threads, which would ensure that no contention can occur on the pool. However, given that the time that is spent using the EJB timer service is typically short compared to the duration of the transactions, a smaller size should work well with barely any contention. So our advice is to start with the default size, monitor the system, and then increase the size if evidence exists of a significant contention under normal conditions.

We have not had to resize the `jdbc/curamtimerdb` data source connection pool in our load tests of SPM in a default installation, where our application servers are tuned for high throughput and memory protection. In such an environment, the number of threads is a low multiplier of the number of cores that are available to the application server, as documented in the [“Thread pools” on page 12](#) section.

JMS settings

The application server settings for Java Message Service (JMS), which include connection factory and activation specification settings, can affect performance. You must tune both the connection factory settings and the activation specification settings.

The client of a connection factory is the application. The application uses the connection factory to push or pull messages to or from the messaging engine through a queue. The client of an activation specification is the Enterprise JavaBeans (EJB) container. The EJB container obtains an activation specification to register a `MessageEndpointFactory` for the message driven bean (MDB) with a `ResourceAdapter`.

When a client pushes a message to the messaging engine, the messaging engine uses the registered `MessageEndpointFactory` to forward the message to the application, for example, the MDB. Then, the application asynchronously receives messages, rather than requiring the client to poll or block by trying to pull a message from the queue.

Connection factory: `jms/CuramConnectionFactory`

Similar to the `jdbc/curamsibdb` data source, size the connection pool for the `jms/CuramConnectionFactory` connection factory by using the following formula:

WebSphere Application Server (WAS)

```
max_connections = WebContainer_max_threads + SIBJMSRThreadPool_max_threads + 1
```

WebLogic Server (WLS)

```
max_connections = default_max_thread_constraint + MDBWorkManager_Max_thread_constraint + 1
```

To prevent the processing cost of pool growth and shrinkage, it is suggested to set `min_connections = max_connections`.

WAS - activation specifications

Tune the maximum concurrent end points for the JMS activation specifications for the SPM queues. The settings define how many EJB MDBs are available to process JMS messages. Asynchronous processing concurrency in the application server is limited by the lower of the number of either MDBs or `SIBJMSRThreadPool` threads.

Set the maximum concurrent end points for the SPM error queues to 1. This setting is suggested because errors are not expected at high volume.

This sizing applies to the following activation specifications:

- `DPErrors`
- `WorkflowErrors`
- `CuramDeadMessageQueue`

Then, for simplicity, use the following formulas to set the maximum concurrent end points for the three main Cúram queues:

```
max_end_points_DPEnactment = SIBJMSRThreadPool_max_threads
max_end_points_WorkflowEnactment = SIBJMSRThreadPool_max_threads
max_end_points_WorkflowActivity = SIBJMSRThreadPool_max_threads
```

The previous formulas effectively simplify tuning for asynchronous processing by creating a single point for tuning, which is the number of JMS threads. However, if more granular tuning is required, you can decrease the concurrent end points, for either SPM Deferred Processing or Workflow.

WLS - message driven beans

In WLS, the number of EJB MDBs is set in the `weblogic-ejb-jar.xml` deployment descriptor. The descriptor also associates MDBs with the `MDBWorkManager`, as shown in the following example:

```
<weblogic-enterprise-bean>
  ...
  <message-driven-descriptor>
  <pool>
    <max-beans-in-free-pool>3</max-beans-in-free-pool>
    <initial-beans-in-free-pool>3</initial-beans-in-free-pool>
  </pool>
</message-driven-descriptor>
  ...
  <dispatch-policy>MDBWorkManager</dispatch-policy>
</weblogic-enterprise-bean>
```

Set both the maximum and initial values of the beans in the free pool for the SPM error MDBs to 1. This setting is suggested because errors are not expected at high volume, and it applies to the `DPErrorsMDB` and `WorkflowErrorsMDB` beans.

Then, for simplicity, use the following formulas to set both the maximum and initial values of the beans in the free pool for the three main SPM MDBs:

```
beans_in_free_pool_DPEnactmentMDB=MDBWorkManager_max_thread_constraint
beans_in_free_pool_WorkflowEnactmentMDB=MDBWorkManager_max_thread_constraint
beans_in_free_pool_WorkflowActivityMDB=MDBWorkManager_max_thread_constraint
```

The previous formulas effectively simplify tuning for asynchronous processing by creating a single point for tuning, which is the number of JMS threads. However, if more granular tuning is required, you can decrease the beans in the free pool, for either SPM Deferred Processing or Workflow.

WAS Java 2 security

The recommendation is to turn off Java 2 security because Social Program Management (SPM) does not use it. The WebSphere Application Server (WAS) Java 2 security feature has a performance cost when it is turned on.

In WAS, when Java 2 security is turned on, every time Java code calls the classloader to load a class or a resource and so on, the call goes through a synchronized block, then a security check, and potentially an access to the file system and then another security check. Effectively, the impact is twofold:

- Performance degrades badly in low memory situations, as weak reference caches like `ResourceBundles` are constantly cleared and reloaded. Each reload means a call to the classloader, and hence going through a Java lock, security checks, and a file system access.
- Even in a normal memory situation, it is a bottleneck because Java code that is calling the classloader goes through the lock and security checks.

Monitoring the application server

Use the tasks in the following section to monitor the system after you make the tuning changes.

JVM

To confirm and fine-tune the JVM heap size, turn on garbage collection (GC) logging by adding the following JVM parameter:

```
-verbose:gc
```

On WebSphere Application Server (WAS) with the IBM JVM, you can specify the location of the GC log file by setting the following JVM parameter:

```
-Xverbosegclog:<<path to file>>
```

With a non-IBM JVM, you can specify the location by setting the following JVM parameter:

```
-Xloggc:<<path to file>>
```

You can then process the GC log file to analyze the GC efficiency and identify better GC tuning values. As a general convention, if more than 2% of the JVM time is spent doing garbage collection, adjust the heap size as described in the [“JVM settings”](#) on page 11 section.

JVM heap size for WebSphere Application Server Liberty

Cloud

To get a heap dump if the JVM crashes, set the following JVM parameter:

```
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=$WLP_HOME/usr/servers/CuramServer/
```

-XX:+HeapDumpOnOutOfMemoryError generates a heap dump when an allocation from the Java heap or the permanent generation cannot be satisfied.

Threads

Monitor thread utilization in the thread pools. For example, the WebSphere Performance Viewer - “Active Thread” counter shows the number of threads that are used in a thread pool. You can compare the counter with the defined number of threads to determine whether a thread pool is fully used.

If a thread pool shows as fully used and if spare CPU capacity exists, you can add a thread to the thread pool. Then, use the formulas that are described previously to update the connection pool sizes to reflect the new number of threads. If no spare CPU capacity exists, then you must balance the tuning to favor either online or asynchronous processing.

Consider the case where the online thread pool is fully used, which results in poor user response times from the application server. You might favor the online processing by decreasing the number of asynchronous threads by 1 and by increasing the number of online threads by 1.

JDBC

Monitor the prepared statement cache discards for the `jdbc/curamdb` data source. For example, in WAS, monitor the JDBC connection pools `PrepStmtCacheDiscardCount` in the WebSphere Performance Viewer - Extended Statistic Set.

Not reusing a prepared statement has a significant processing cost. Therefore, aim for no discards, and increase the prepared statement cache size if many discards are reported. If you are using the Oracle database, monitor the maximum number of open cursors and update the configuration as needed.

JMS

Monitor the depth of JMS queues at run time. It indicates how many messages are waiting to be processed. For example, in WAS the AvailableMessageCount counter is available from the WebSphere Performance Viewer-All Statistic-Queues-Queue Stat. The following list indicates the key queues to monitor:

- DPEnactment
- WorkflowEnactment
- WorkflowActivity

If the rate of JMS message processing is not satisfactory and if spare CPU capacity exists, you can increase the number of threads. Use either SIBJMSRThreadPool for WAS or MDBWorkManager for WLS.

However, if no spare CPU capacity exists, review either the maximum concurrent endpoints for the queues in WAS or the beans in the free pool in WLS. In this case, either constrain the queues in WAS or the WLS free pool beans that have a low depth to favor those queues with a high depth. That is, either the queues or the beans that have high queue depths or message counts need more resources.

For more information about applying the constraints, see [“WAS - activation specifications”](#) on page 14 or [“WLS - message driven beans”](#) on page 15.

WAS tunable parameters summary

The table in this section summarizes the parameters that you can tune in WebSphere Application Server (WAS).

Tuning parameter	Description and use	Recommended value
SERVLET CACHING	After a servlet is launched and completes generating the output to cache, a cache entry is created that contains the output and the side effects of the servlet. The side effects can include calls to other servlets, JavaServer Pages (JSP) files, or metadata about the entry, including timeout and entry priority information. Configure servlet caching to save the output of servlets and JavaServer Pages (JSP) files to the dynamic cache.	Disable

Table 3. Summary of tunable parameters for WAS (continued)

Tuning parameter	Description and use	Recommended value
SERVLET REQUEST AND RESPONSE POOLING	Specifies to disable the pooling of servlet request and servlet response objects that are pooled by the web container. When you disable pooling of servlet request and servlet response objects, new servlet request and servlet response objects are created for each request, that can negatively affect performance, but that provide protection from any unforeseen pooling issues.	Disable
JAVA HEAP	By default the application server JVM initial and maximum heap size is set to 1024 MB. However, you can override the default JVM initial and maximum heap size by setting the <code>curam.server.jvm.heap.size</code> property in the <code>AppServer.properties</code> file. We recommend setting minimum heap size = maximum heap size to 4 GB as a starting point. You can then use a heuristic algorithm to further tune the heap size and achieve high garbage collection efficiency.	Start with <code>curam.server.jvm.heap.size=4096</code> Then further tune the application server JVM heap size by using the following heuristic: <code>-Xmx = 8 * avg_used_after_global</code> <code>-Xms = 8 * avg_used_after_global</code> <code>-Xmn = 6 * avg_used_after_global</code>
NUMBER OF HTTP SESSIONS	Tune the number of allowed HTTP sessions to a large enough number to serve your expected number of users. Ensure that the timeout is set so as not to waste sessions and to keep memory usage under control.	See description.
CONNECTION TIMEOUT and KEEP-ALIVE	HTTP persistent connection, or HTTP keepalive, is where a single connection is used to send and receive multiple HTTP requests.	Depends on system behavior under load. Disable if the issues that are mentioned in “Configuring persistent connections” on page 22 occur.
NUMBER OF WEB THREADS	HTTP requests are processed by a pool of server threads. You can configure the minimum and maximum thread pool size for the web container for optimal performance.	<code>WebContainer_max_threads = number of cores</code> <code>WebContainer_min_threads = WebContainer_max_threads</code>

Table 3. Summary of tunable parameters for WAS (continued)

Tuning parameter	Description and use	Recommended value
NUMBER OF MDB THREADS	SIBJMSRAThreadPool is set up to process JMS messages. Always set minimum threads = maximum threads to avoid thread reconstruction.	SIBJMSRAThreadPool_max_th reads = number of cores SIBJMSRAThreadPool_min_th reads = SIBJMSRAThreadPool_max_th reads
MAX CONCURRENT MDB INVOCATIONS PER ENDPOINT (activation specifications)	A JMS activation specification is associated with one or more message-driven beans (MDBs) and provides the configuration necessary for the MDBs to receive messages.	For all queues set the value equal to SIBJMSRAThreadPool_max_th reads
QUEUES FETCH (activation specifications)	Use this setting to control the message retrieval from the queue. We recommend setting the value equal to SIBJMSRAThreadPool_max_th reads to aim for zero messages in the queue.	Set equal to SIBJMSRAThreadPool_max_th reads
MIN/MAX CONNECTIONS (connection factory)	This setting is used to size the connection pool for JMS connection factory messages. To prevent the processing cost of pool growth and shrinkage, we suggest setting min_connections = max_connections.	max_connections = WebContainer_max_threads + SIBJMSRAThreadPool_max_th reads + 1 min_connections=max_connections
NUMBER OF DATABASE CONNECTIONS (jdbc data source)	An SPM transaction needs two JDBC connections, transaction and KeyServer. jdbc/curamdb data source connection pool needs to be sized to ensure that more connections are available than threads. Use Performance Monitoring Infrastructure (PMI) to identify the average number of connections and to size the pool to three times that number to avoid deadlocks.	max_connections = WebContainer_max_threads + SIBJMSRAThreadPool_max_th reads + 1
STATEMENT CACHE SIZE (jdbc data source)	This parameter controls the size of the data source prepared statement cache size. We recommend a starting value of 1000. However, monitor the cache size and increase the value if discards occur.	Starting value 1000

Table 3. Summary of tunable parameters for WAS (continued)

Tuning parameter	Description and use	Recommended value
CONTAINER MANAGED PERSISTENCE (jdbc data source)	An entity bean that uses container-managed persistence (CMP) delegates the management of its state, or persistence, to the application server container. We observed better response times when CMP is disabled.	Disable in Curamdb and curamsibdb
JAVA 2 SECURITY	Social Program Management (SPM) does not use Java 2 Security and this WAS feature has a performance cost when it is turned on.	Turn off

Chapter 7. Tuning the HTTP server

Use the guidelines in the following sections to tune the HTTP server.

Configuring the HTTP server for static content

As described in the “Static content” on page 9 section, package the Social Program Management (SPM) static content outside the EAR files that are to be copied onto the HTTP Server. The guidelines assume that the relative URL `/CuramStatic/` is being used.

You can use the Apache module `mod_expires` to automatically configure the HTTP header of static content to be cached by the browser. To apply the configuration, configure the following settings in the `httpd.conf` file on the HTTP server:

```
<LocationMatch /(Curam|CuramStatic)>
...
ExpiresActive On
ExpiresByType text/css "access plus 1 month"
ExpiresByType text/javascript "access plus 1 month"
ExpiresByType text/plain "access plus 1 month"
ExpiresByType image/gif "access plus 1 month"
ExpiresByType image/jpg "access plus 1 month"
ExpiresByType image/png "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
ExpiresByType application/x-javascript "access plus 1 month"
Header unset Last-Modified
</LocationMatch>
```

To increase performance under heavy loads, consider hosting the static content on a separate server, which in turn reduces the load on the SPM application server.

Compressing content from HTTP server

You can compress the content that is served by HTTP Server.

You can use the Apache module `mod_deflate` to compress the content that is served through the HTTP Server. To apply the configuration, configure the following settings in the `httpd.conf` file on the HTTP server:

```
<LocationMatch /(Curam|CuramStatic)>
AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css application/x-
javascript
# Netscape 4.x has some problems...
BrowserMatch ^Mozilla/4 gzip-only-text/html
# Netscape 4.06-4.08 have some more problems
BrowserMatch ^Mozilla/4\.[0-9] no-gzip
# MSIE masquerades as Netscape, but it is fine
BrowserMatch \bMSIE[E] !no-gzip !gzip-only-text/html
# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary
...
</LocationMatch/>
```

Monitoring the HTTP server

To monitor the HTTP server, modify the `httpd.conf` file as shown in the example.

Modify the `httpd.conf` file so it matches the following example. If you are using secure access to `server-status`, see the relevant HTTP server documentation.

```
LoadModule status_module modules/mod_status.so
<Location /server-status
```

```
SetHandler server-status
  Order allow,deny
  Allow from all
</Location>
```

To gather extended status information, ensure that the following section is not commented out in the `httpd.conf` file:

```
<IfModule mod_status.c>
  ExtendedStatus On
</IfModule>
```

Restart the web server and open the following URL in your browser:

```
http://<web_server_host>/server-status
```

In the `server-status` page, you can view the number of threads and their state.

Configuring persistent connections

HTTP persistent connection or HTTP keepalive is a TCP feature that is used to reduce network congestion and CPU usage, in particular on systems that have lower hardware specifications. The feature allows multiple requests over a single connection. Reusing a connection in this way is useful with secure HTTP because the same connection does not need to be renegotiated every time.

The following parameters in the `httpd.conf` file control persistent connections:

```
KeepAlive On
MaxKeepAliveRequest 1000
KeepAliveTimeout 2
```

The `KeepAliveTimeout` parameter controls the amount of time that the server waits for the next request on a persistent connection. If the value is too small, connections might not be reused efficiently. If the value is too large, web server threads might not be used efficiently. We recommend starting by setting the value to 2 and then adjust it if necessary. It is suggested that you keep this value lower than the average think time of the users of the system.

Recent performance tests highlighted that some disadvantages occur when persistent connections are used with SPM application servers:

- Intermittent blank or frozen application
- Failure to load content after some time

In addition, we found that the WebSphere Application Server (WAS) setting `WebServer Plug-in Properties-Connection Timeout` affects the server's ability to handle heavy loads. When the feature is turned on, issues that arise include huge numbers of objects are created when the server is unable to handle the load, extreme resource utilization occurs on the server, and server failures occur.

If you experience any of the issues when you test your system with heavy loads, we recommend disabling the persistent Connections/keepalive feature.

Adjusting thread tuning

If the system throughput is known or a good estimation of the system throughput is available, you can calculate the number of threads that are in the web server. You can do the calculation after the persistent connections are configured.

The following formula indicates the total number of threads that are required in the web server:

```
Throughput * KeepAliveTimeout
```

Alternatively, you can use the following formula to obtain a rough starting value for the number of threads that you can use, where `WebContainer_max` is the number of application server threads that are available to process HTTP requests:

```
WebContainer_max * KeepAliveTimeout
```

If the HTTP server load balances against multiple application servers, multiply the number of threads as needed:

```
ratio_of_application_servers_per_HTTP_servers * WebContainer_max * KeepAliveTimeout
```

The previous thread calculation is likely to be too low a number, but you can use it as a quick start. Then, refine the number of threads during load tests or production monitoring by including the 95th percentile of response time through the following heuristic:

```
threads = ratio_of_application_servers_per_HTTP_servers * WebContainer_max *  
(KeepAliveTimeout + user_interaction_95th%ile_response_time)
```

You might need to adjust the number of threads further depending on the following factors:

- The average amount of static content that is in your web pages
- How well the static content is cached by the web browser

The dependency occurs because some web clients can use multiple connections for retrieving the static content on a web page. The behavior is a strong case for monitoring the HTTP server threads utilization.

During monitoring, as a general convention, increase the number of threads only if the following statements are true:

- All threads are active, and no idle threads exists.
- The response time is not acceptable.
- The CPU is not fully utilized.

The procedure for setting the number of threads differs according to the operating system:

Windows

```
ThreadLimit          300  
ThreadsPerChild     300  
MaxRequestsPerChild 0
```

UNIX systems

```
ServerLimit          1  
ThreadLimit          300  
StartServers         1  
MaxClients           300  
MinSpareThreads      300  
MaxSpareThreads      300  
ThreadsPerChild     300  
MaxRequestsPerChild 0
```

Notices

This information was developed for products and services offered in the United States.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Privacy Policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies or other similar technologies that collect each user's name, user name, password, and/or other personally identifiable information for purposes of session management, authentication, enhanced user usability, single sign-on configuration and/or other usage tracking and/or functional purposes. These cookies or other similar technologies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.



Part Number:

(1P) P/N: