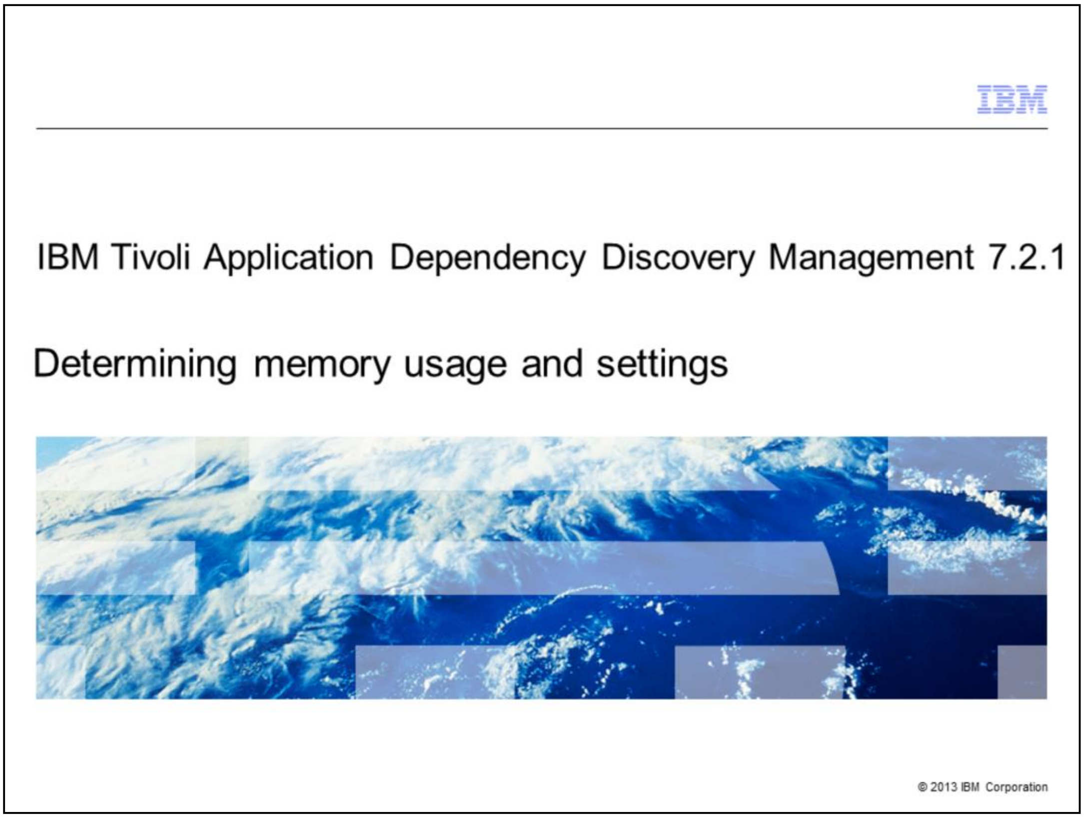
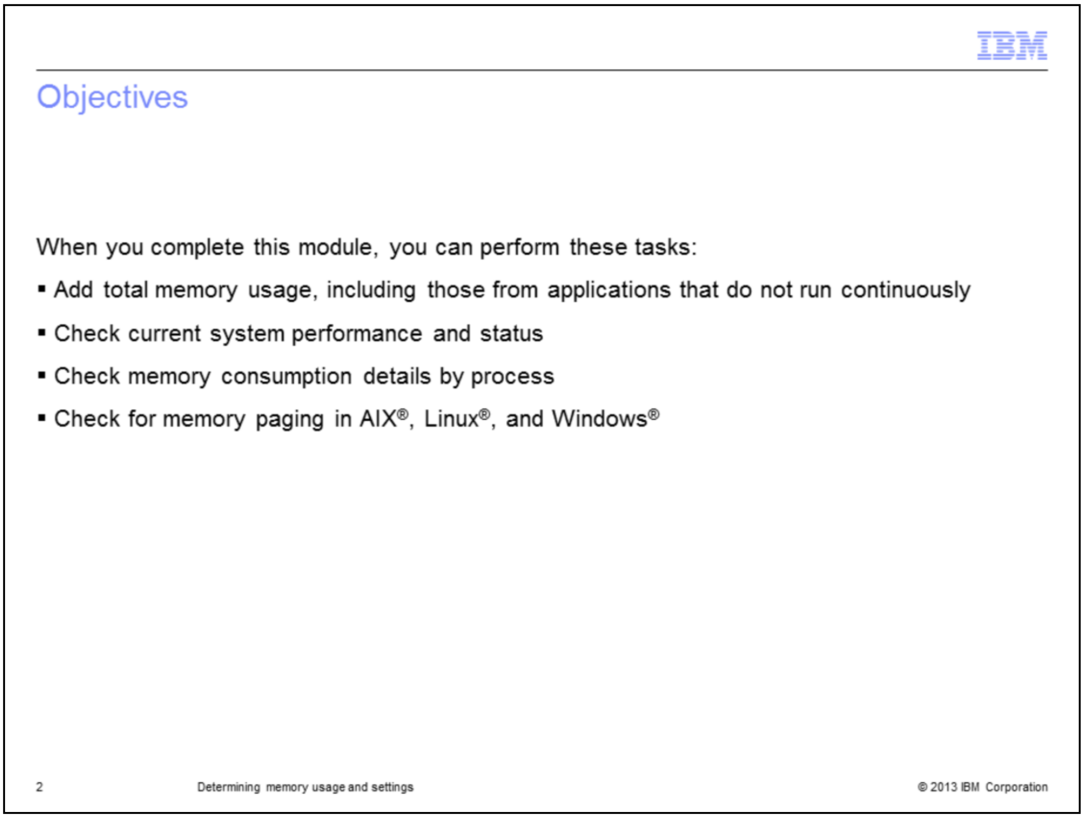


Tivoli® Application Dependency Discovery Management version 7.2.1. Determining memory usage and settings.



When you complete this module, you can perform these tasks:

- Add total memory usage, including those from applications that do not run continuously
- Check current system performance and status
- Check memory consumption details by process
- Check for memory paging in AIX, Linux, and Windows

How to determine Tivoli Application Dependency Discovery Manager potential maximum memory allocations

Memory settings are done in these files:

- collation.properties
- cmdb-context.xml
- Other .sh or .bat scripts

Determine the Tivoli Application Dependency Discovery Manager potential memory maximum allocations. Memory settings are made in files. Some are in the collation.properties, cmdb-context.xml, and other .sh and .bat scripts.

Add the memory used by Tivoli Application Dependency Discovery Manager

- $TADDM_HOME/dist/etc/collation.properties:
  grep Xmx collation.properties

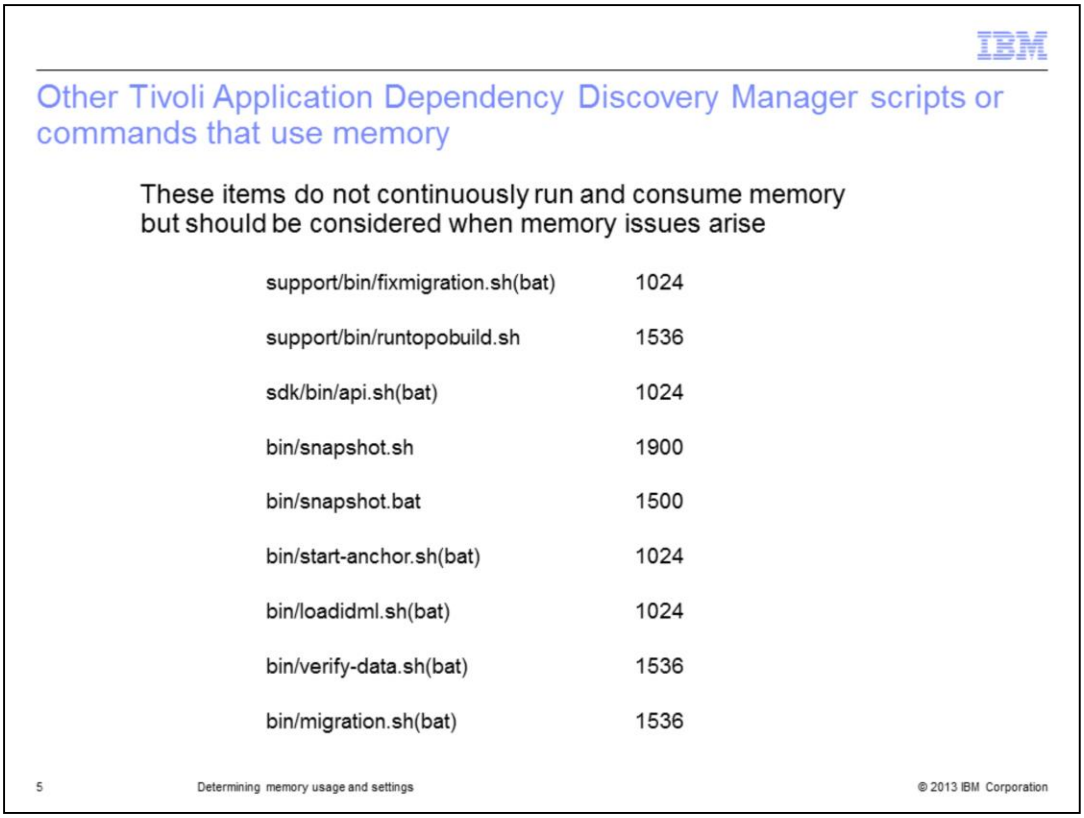
Depending on the Tivoli Application Dependency Discovery Manager Server type, determine the appropriate context.xml

- ./deploy-tomcat/ROOT/WEB-INF/discovery-server-context.xml
- ./deploy-tomcat/ROOT/WEB-INF/ecmdb-context.xml
- ./deploy-tomcat/ROOT/WEB-INF/cmdb-context.xml
- ./deploy-tomcat/ROOT/WEB-INF/storage-server-context.xml
  grep Xmx cmdb-context.xml

	Xmx		
Servicename	**Domain**	**Discovery**	**Storage**
tomcat	**1024**	1024	1024
Discover	1024	1024	
EventsCore	512		
StorageServer	1512		4096
DiscoverAdmin	512		
DiscoverService		1512	
Proxy	1024		
Gigaspaces	768	768	768
Topology	4096		
Sum	**10472**	**4328**	**5888**

4 Determining memory usage and settings

Total all of the memory maximums, Xmx, settings depending on the Tivoli Application Dependency Discovery Manager server type Domain, Discovery or Storage Server. Start by looking in the dist/etc/collation.properties. Grep for Xmx. Properties in this file will override properties in the following context.xml files. If there are no overriding Xmx arguments in the collation.properties file, the Xmx setting in the corresponding context.xml file is used. The table in this slide shows the default maximum memory allocations for each Tivoli Application Dependency Discovery Manager JVM.



Other Tivoli Application Dependency Discovery Manager scripts or commands that use memory are listed here with their default maximum memory settings. Because these scripts or commands are not continuous running programs, the memory usage is temporary, but you should consider them when you determine memory usage.

Check current system performance

`top` (Linux) `topas` (AIX)

- Check the output of these commands for memory and swap
- In this example, there is little free memory and swap is being used

```
top - 12:52:49 up 56 days, 17:02, 10 users, load average: 0.31, 0.48, 1.08
Tasks: 182 total, 1 running, 181 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.3%us, 0.0%sy, 0.0%ni, 98.0%id, 1.7%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4044648k total, 3913916k used, 130732k free,  2180k buffers
Swap: 5898232k total, 4205820k used, 1692412k free, 302496k cached
```

- Check top processes that consume the memory
  They are listed in order of highest consumers

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24401	taddmusr	18	0	5198m	1.3g	3320	S	10.3	33.7	1375:35	java
24399	taddmusr	24	0	1320m	167m	2256	S	3.7	4.2	740:13.59	java
24651	taddmusr	18	0	1718m	153m	2212	S	3.7	3.9	714:22.88	java
24400	taddmusr	19	0	2102m	744m	3468	S	3.0	18.9	218:32.38	java
23365	db2inst1	25	0	2541m	175m	165m	S	1.7	4.5	410:20.60	db2sysc
24316	taddmusr	22	0	1977m	437m	3896	S	0.3	11.1	487:26.54	java

- Consider also the amount of OS overhead required. For example, an OS running on an LPAR can consume 2 G of memory or more.

6 Determining memory usage and settings

Check the current system performance by using **top** on Linux or **topas** on AIX. Check the output of the command for memory and swap usage. In this example, there is little free memory, 130,732k free, and you can see that swap is being used. The command also shows each process listing the top consumers first. In this example, you can see that the user is taddmusr, which is a clue this is a Tivoli Application Dependency Discovery Manager process. To confirm the last column shows Java, indicating which Tivoli Application Dependency Discovery Manager processes are Java process. Further confirmation of the full command can be found by issuing the **ps –ef** command and searching for the PID, as shown in the first column.

Check current system status

Use the dist/bin/healthcheck tool

```
healthcheck -u administrator -p collation checkTaddmStatus
```

```
************************************************************************************************
**                                   Begin checkTaddmStatus                                   **
**                                   ----------------------                                   **
**                                                                                            **
**       This section displays information about the TADDM status(like bin/control status)   **
**       for the major services but also about the services they provide.  For the major      **
**       services, it shows (where available):                                                **
**                                                                                            **
**       Label           -- Service Name (+-- indicates a sub-service)                        **
**       Max memory      -- Maximum Memory available to JVM (-Xmx setting)                    **
**       Total Memory    -- Total Memory used right now                                       **
**       Free Memory     -- Of the total, how much is Free                                    **
**       Threads         -- Number of threads assigned                                        **
**       Running Threads -- Number of threads in use                                          **
**       Status          -- Status of the service                                             **
**                                                                                            **
**       See java.lang.Runtime for further information about these values                     **
**                                                                                            **
************************************************************************************************
Label                       Service Max Memory Total Memory Free Memory Threads Running Threads  Status
GigaSpaces                        -          -            -           -       -               - Started
Tomcat                            -          -            -           -       -               - Started
StorageService                    -     4096MB       4096MB        89MB     126              27 Started
+--               TopologyManager          -            -           -       -               - Started
+--               TopologyBuilder          -            -           -       -               - Started
+--                 ReportsServer          -            -           -       -               - Started
+--          AuthorizationManager          -            -           -       -               - Started
+--                     ApiServer          -            -           -       -               - Started
+--                ProfileManager          -            -           -       -               - Started
+--               SecurityManager          -            -           -       -               - Started
+--                      Template          -            -           -       -               - Started
+--                   ClientProxy          -            -           -       -               - Started
+--                     Semaphore          -            -           -       -               - Started
+--                 ChangeManager          -            -           -       -               - Started
+--                   ViewManager          -            -           -       -               - Started
DbInit                            -          -            -           -       -               - Started
```

7 Determining memory usage and settings

You can use the healthcheck tool provided with Tivoli Application Dependency Discovery Manager to help determine current memory usage. In some cases, not all JVM status is gathered, and JVM thread dumps are required.

Check memory consumption for each Tivoli Application Dependency Discovery Manager Java virtual machine (1 of 2)

- If the highest consumers of memory are Tivoli Application Dependency Discovery Manager JVMs from the **top** or **topas** commands, get thread memory dumps
  Technote link: http://www-01.ibm.com/support/docview.wss?uid=swg21598190
- Check each for their memory usage
- Find the javacore in the **dist/external/gigaspaces-4.1/bin** or **dist/bin** directory
- Find the section that shows memory information, which shows the hex values of the memory usage and allocation:

```
NULL           ------------------------------------------------------------------------
0SECTION       MEMINFO subcomponent dump routine
NULL           =================================
1STHEAPFREE    Bytes of Heap Space Free: 3643e90
1STHEAPALLOC   Bytes of Heap Space Allocated: 8000000
NULL
```

8 Determining memory usage and settings © 2013 IBM Corporation

Check the memory consumption for each Tivoli Application Dependency Discovery Manager Java Virtual Machine (JVM). Technote 1598190 describes how to force a JVM thread dump for each Tivoli Application Dependency Discovery Manager service. After you create the javacore using the steps in this technote, find the javacore files in the dist/external/gigaspaces-4.1/bin or the dist/bin directory. Open the javacore file with a text file viewer or editor, and find the section that shows memory information. MEMINFO subcomponent indicates the right section.

IBM

Check memory consumption for each Tivoli Application Dependency Discovery Manager Java virtual machine (2 of 2)

```
NULL           ------------------------------------------------------------------------
0SECTION       MEMINFO subcomponent dump routine
NULL           =================================
1STHEAPFREE    Bytes of Heap Space Free: 3643e90
1STHEAPALLOC   Bytes of Heap Space Allocated: 8000000
NULL
```

- Use the calculator in programmer mode to convert.
- Subtract to determine how much memory is in use
  3643e90 = 56901264 (~.42 GB free)
  8000000 = 134217728 (1 GB allocated for the JVM)
  134217728 – 56901264 = 77316464 (~.58 GB in use)

The result is that slightly less than half of the allocated memory is still available

- If the highest consumers of memory are not Tivoli Application Dependency Discovery Manager JVMs, investigate those PIDs listed in the **top** or **topas** output

The information is in hex; so you must convert it to decimal. Use the calculator in programmer mode to convert. Subtract to determine how much memory is in use. In this example, there is .42 gigabit free out of a maximum of 1 gigabit. Investigate the other javacore files for each of the high consumers listed in the **top** or **topas** command.

Check for memory paging

The vmstat output shows whether paging is taking place

vmstat output has this format:

- On AIX:

```
kthr     memory             page              faults        cpu        time
----- ----------- ------------------------ ------------ ----------- --------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa hr mi se
 0  0 45483   221   0   0   0   0    1   0 224  326 362 24  7 69  0 15:10:22
 0  0 45483   220   0   0   0   0    0   0 159   83  53  1  1 98  0 15:10:23
 2  0 45483   220   0   0   0   0    0   0 145  115  46  0  9 90  1 15:10:24
```

- On Linux:

```
procs -----------memory----------  ---swap--  -----io---- --system-- ----cpu----
 r  b   swpd   free    buff   cache   si   so    bi    bo   in    cs us sy  id wa
 0  0  17196 679860 1196656 2594884    0    0     1     4    0     0  0  0 100  0
 0  0  17196 679868 1196656 2594884    0    0     0    40 1012    43  0  0 100  0
 0  0  17196 679992 1196656 2594884    0    0     0     3 1004    43  0  0 100  0
```

The following columns are of interest:

- AIX: pi and po (page in and page out)
- Linux: si and so (swap in and swap out)

Nonzero values indicate that paging is taking place

Reference:
http://publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp?topic/com.ibm.java.doc.igaa/_1vg00014557b090-11cd67f58fb-7fec_1001.html

Check for memory paging. You can use the **vmstat** command for AIX and Linux. The columns of interest are pi and po (page in and page out) for AIX and the si and so (swap in and swap out) columns in Linux. Nonzero values indicate that paging is taking place.

IBM

Check for memory paging in Windows

- Looking for paging on Windows
  - The paging activity log generated using permon can be reloaded into permon to display the data. The information describes hot to start perfmon and load the data.
- Start perfmon with one of these methods:
  - Type **perfmon** at a command prompt line or use the **Start > Run** option
  - Click **Control Panel > Administrative Tools > Performance**
- Load the log file in to perfmon:
  1. From the graphical display on the right side, select the icon for **View Log Data**, which opens the System Monitor properties
  2. Select the **Log files** radio button, click **Add**, and browse to the perfmon log file
     Example: Java Paging Usage
  3. Select the **Data** tab, and click the **Add** button to select data points in the perfmon log file
     Only the counters added to the log file are available
  4. When the **Page Faults** counter for the Java instance is added, click **Close**
     You return to the System Monitor Properties window on the **Data** tab
  5. Change the scale of the graph by selecting the correct scaling for the data in the **Scale** pull-down list
- Although a few page faults are acceptable, many page faults indicate a paging problem

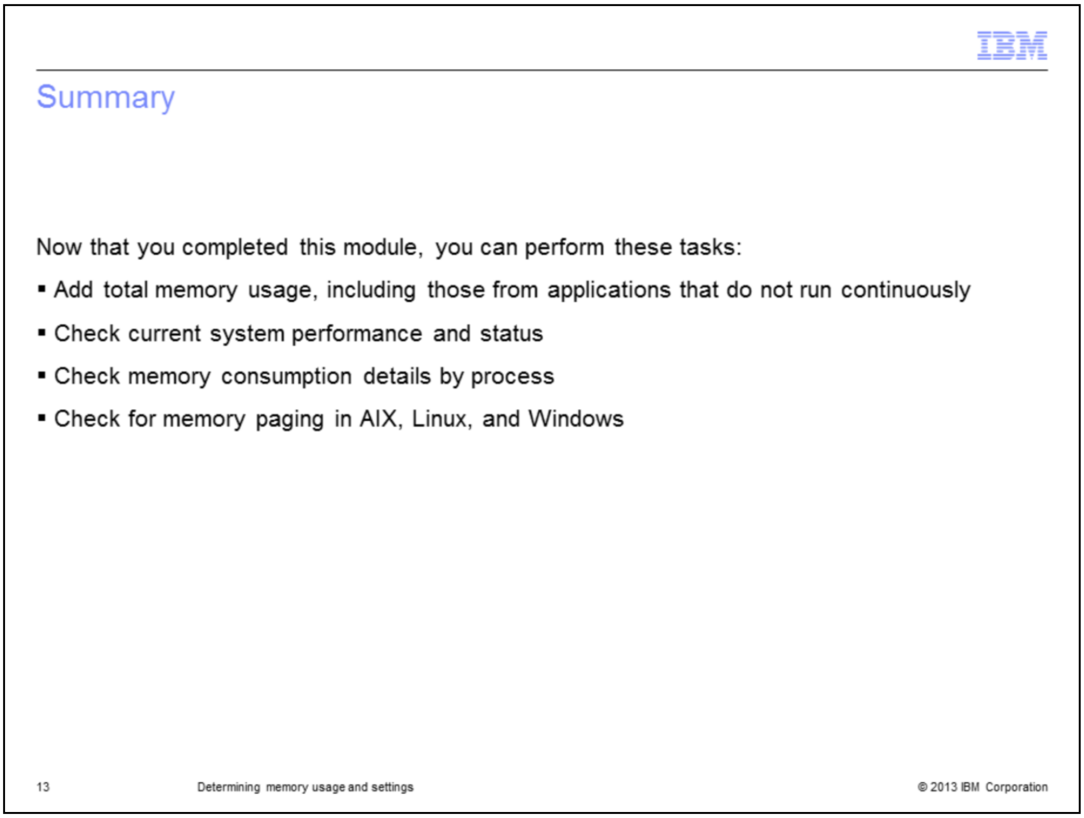
You can check for memory paging on Windows using PerfMon. Start PerfMon by using the command line **perfmon** or select **the Control Panel > Administrative Tools > Performance option**. Load the log file into PerfMon. From the graphical display on the right side, select the icon for **view Log Data**. Select the **log files** radio button and click **Add**. Now browse to the location of the PerfMon log, for example, Java Paging Usage. Select the **Data** tab and click the **Add** button to select the data points in the PerfMon log file that you want to display. After the Page Faults counter for the Java instance is added, select **Close**. This action returns you to the System Monitor Properties window on the **Data** tab. You can change the scale of the graph using the **Scale** pull-down list and selecting the correct scaling of the data. A small number of page faults is acceptable. If a larger number of page faults are occurring, there is a paging problem.

Review

- The total amount of Xmx being greater than the physical memory can be a potential problem
- Consider other applications and operating system (OS) overhead running on this server
- Is this OS running on an LPAR? How much overhead is being used by LPAR management?
- You can see memory consumption by application by issuing the **top** and **topas** commands.

12 Determining memory usage and settings

To summarize, add all of the maximum memory, Xmx. If this number is greater than the physical memory on the server, you potentially have performance problems. Do not forget to consider other applications and operating system overhead running on the server. Additionally, if this server is running on an LPAR, check how much memory is used to manage the LPAR. You can discover memory consumption by process using **top** and **topas** commands.



Now that you completed this module, you can perform these tasks:

- Add total memory usage, including those from applications that do not run continuously

- Check current system performance and status

- Check memory consumption details by process

- Check for memory paging in AIX, Linux, and Windows

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, AIX, and Tivoli 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 other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.