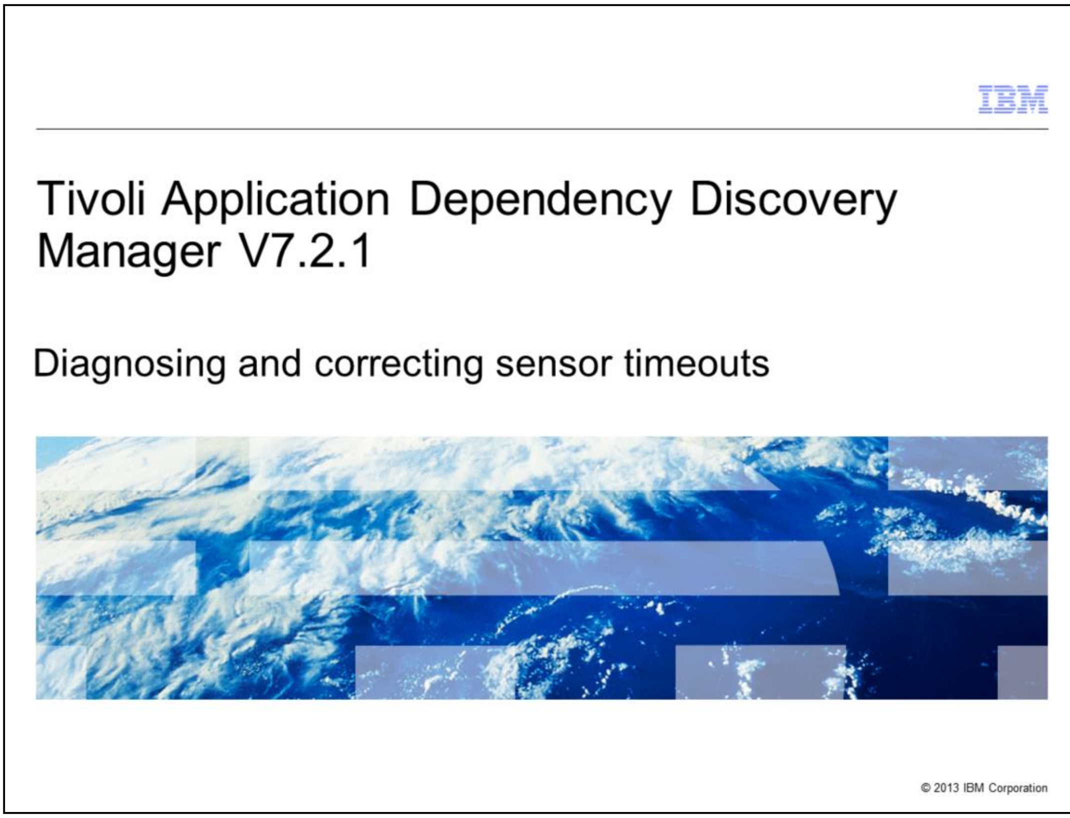


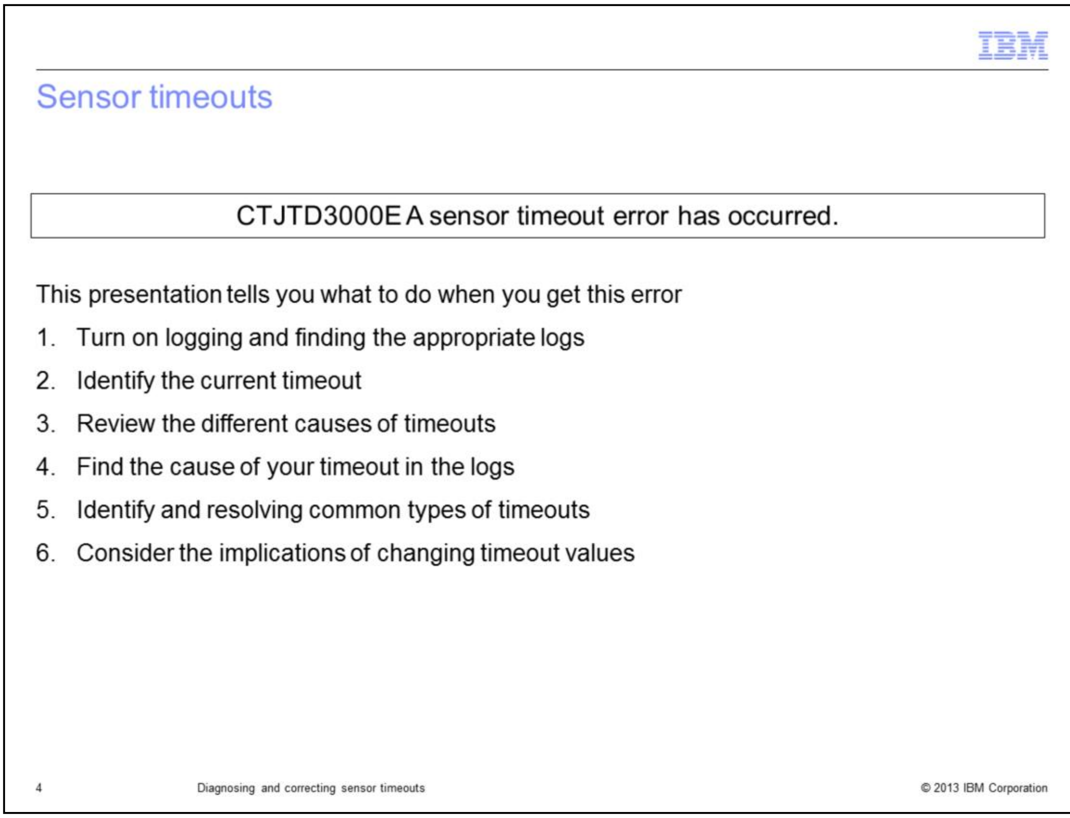
In this module, you learn how to diagnose and correct sensor timeouts in Tivoli®
Application Dependency Discovery Manager V7.2.1.

An assumption for this module is that you are familiar with Tivoli Application Dependency Discovery Manager.

Objectives

When you complete this module, you can diagnose and correct many sensor timeouts within Tivoli Application Dependency Discovery Manager

The objective for this module is to understand how to diagnose and correct sensor timeouts within Tivoli Application Dependency Discovery Manager.



You know your sensor has timed out if it fails with a “CTJTD3000E A sensor timeout error has occurred” error in the discovery history.

This presentation shows the steps to diagnose this error.

-- Turn on logging and finding the appropriate logs

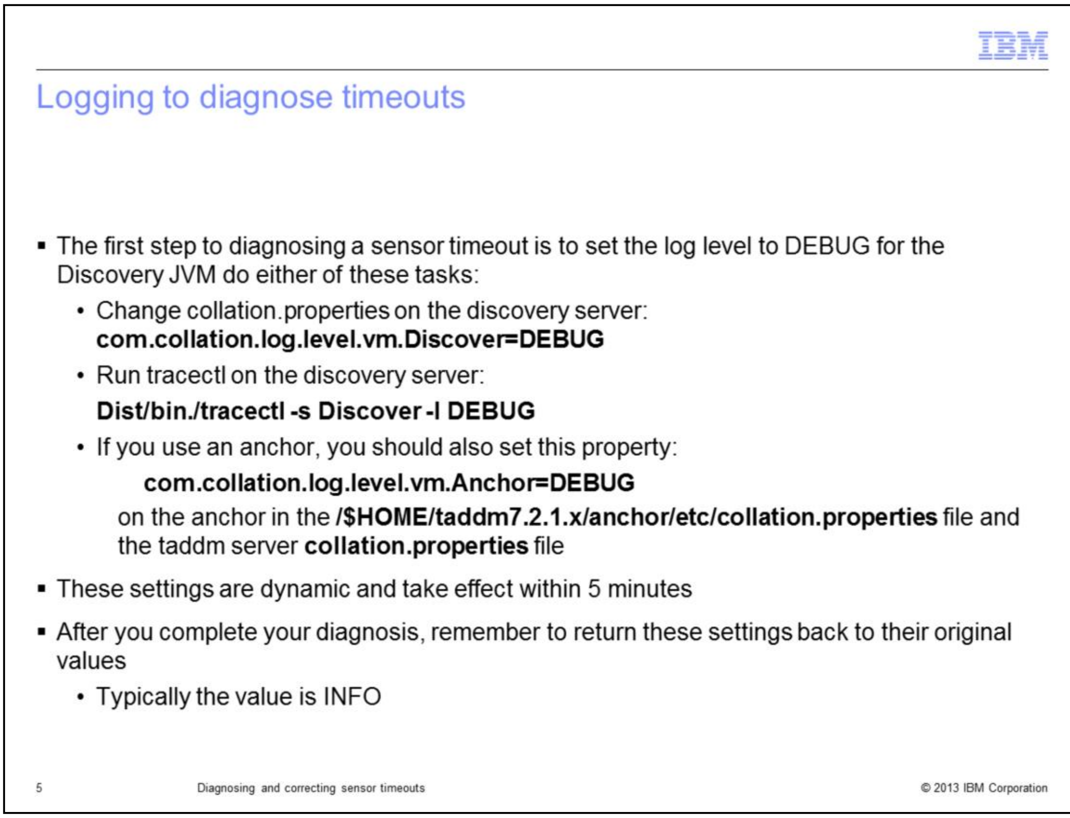
-- Identify the current timeout

-- Review different causes of timeouts

-- Find the cause of your timeout in the logs

-- Identify and resolve common timeouts

-- Consider the implications of changing timeout values



To diagnose a sensor timeout, you must first set the Discover JVM to DEBUG mode. You can set the debug mode two ways.

1. Set the **collation.properties** setting **com.collation.log.level.vm.Discover**.

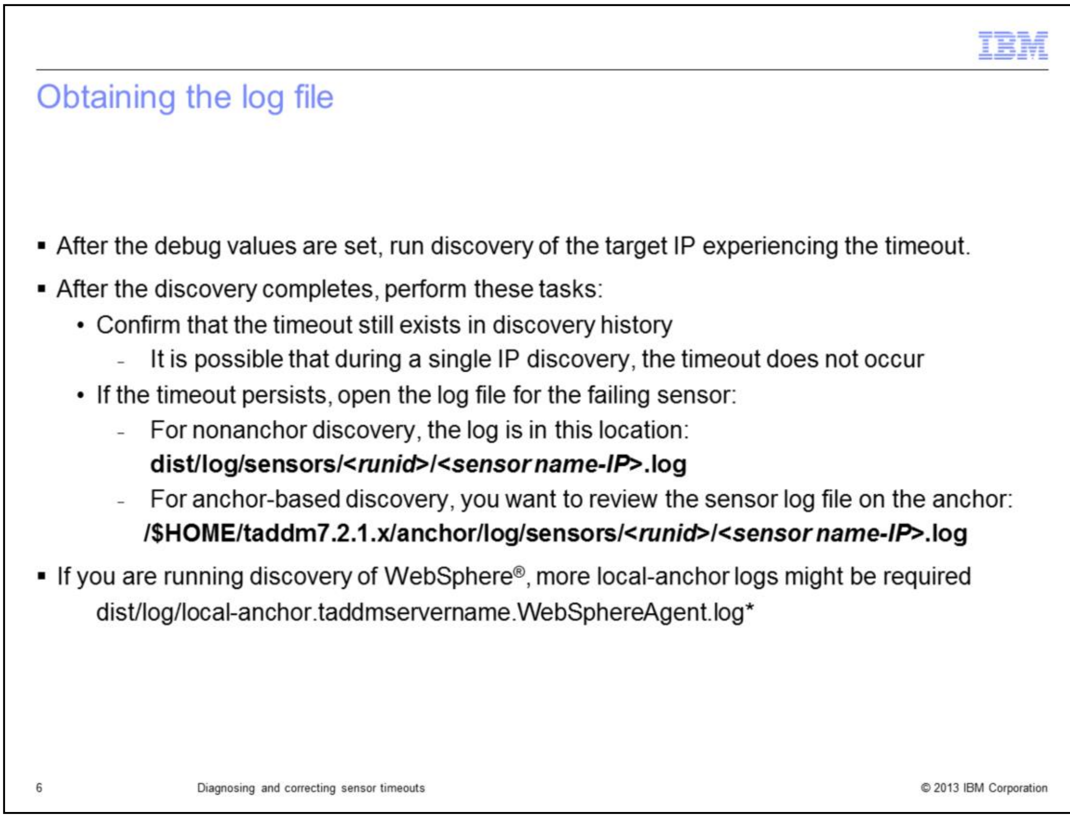
2. In the **dist/bin** directory, run the command tracectl -s Discover -l DEBUG.

The setting is dynamic; wait for 5 minutes to ensure that it was picked up.

If you are discovering with an anchor, you must also set a value on the anchor server to ensure that DEBUG logs are also collected there.

Set the **com.collation.log.level.vm.Anchor** value to **DEBUG**.

After you complete the diagnosis, be sure to set these log level settings back to their original values, typically INFO.



After debug is set, run discovery of the target IP address that is experiencing the timeout.

After it completes, confirm that the timeout error still exists in discovery history. It is possible that during a single IP discovery, the timeout will not occur. This behavior can occur for several reasons:

A) The problem is intermittent and occurs only when the target itself is overused and the commands cannot return in a reasonable time frame.

B) If the target is Windows®, the gateway might be throttling the requests. This concept is presented later.

C) Other load or resources-based factors, such as hangs in the login because of Active Directory problems, or WMI being inoperable during the prior discovery.

If the timeout persists, open the log file for the affected sensor. The log is in the **dist/log/sensors/<*runid*>** directory, where **runid** is the date stamp of the discovery you ran. If you use an anchor, the sensor log is on the anchor in the anchor users home directory under **taddm7.2.1.x/anchor/log/sensors/<*runid*>**. The name of the log includes the sensor name and IP that is failing. For WebSphere, you might also look at the local anchor log in the software discovery server **dist/log** directory.

IBM

Determining the current timeout

1. Verify that the log is in DEBUG mode
   - You see DEBUG messages at the beginning:

     ```
     2012-11-09 11:37:58,451 DiscoverManager [DiscoverWorker-82]
     SessionSensor-1.2.3.4-[23,22] DEBUG util.ProfileProps - Calling
     for property: com.ibm.cdb.discover.PreferScriptDiscovery
     ```

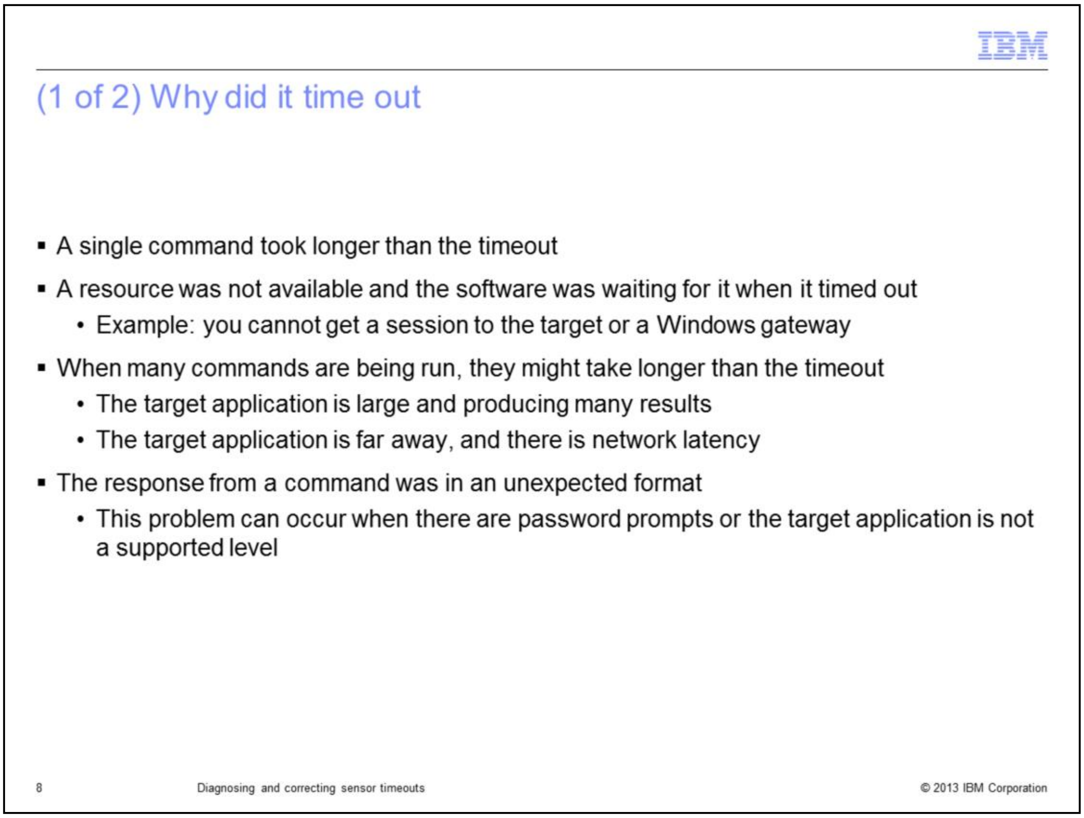
   If you do not see these entries, verify the **log.level** settings explained on the previous slide
2. Note that the time stamp of the first message in the preceding log is **2012-11-09 11:37:58**
3. Search for the first occurrence of DISCOVER_SENSOR_CLEANUP in the log:

   ```
   2012-11-09 11:47:58,728 DiscoverManager
   [DISCOVER_SENSOR_CLEANUP_DiscoverWorker-82] SessionSensor-
   1.2.3.4-[23,22] DEBUG session.Ssh2SessionClient …….
   ```
4. Note that the time stamps of the two previous messages are 10 minutes apart

The time difference between the first message and the cleanup message is the current timeout for this sensor. It might be a default setting for all sensors or specific to this sensor or a command timeout

When you review the log, the first item to look for is how long is the sensor takes to time out. This information helps later when you assess which timeout value to change, if a change is required.

To determine the time to time out, note the time stamp of the first message in the log file. Then, search for the first instance of **DISCOVER_SENSOR_CLEANUP**. This message indicates that a timeout occurred. Determine the difference between the time stamp of this message and the first message in the log. This difference is an indication of the current timeout setting, which can be a sensor timeout or a command timeout.



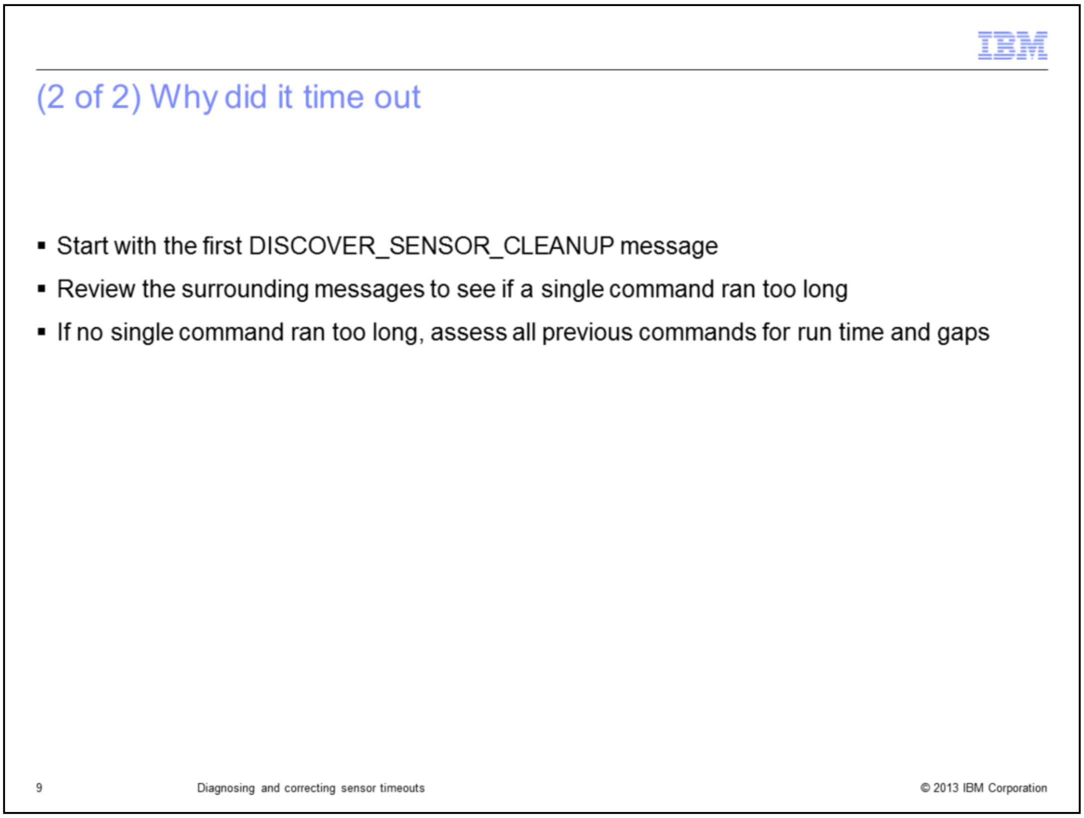
Now that you know how long it took to time out, you must understand why. There are many possible reasons:

A) A single command took longer than that timeout.

B) A resource was not available and the software was waiting for it when it timed out. For example, the Windows **gateway ssh** program was denying the request because of load and the retry time exceeded the timeout.

C) The sensor is running too many commands, and the combination of all of them and the length of their responses exceeded the timeout value.

D) The response from a command was in an unexpected format. and it caused the sensor to hang. For example, password prompts on login.



Here is a list of things you can do to determine the cause of the timeout.

Locate the first DISCOVER_SENSOR_CLEANUP message.

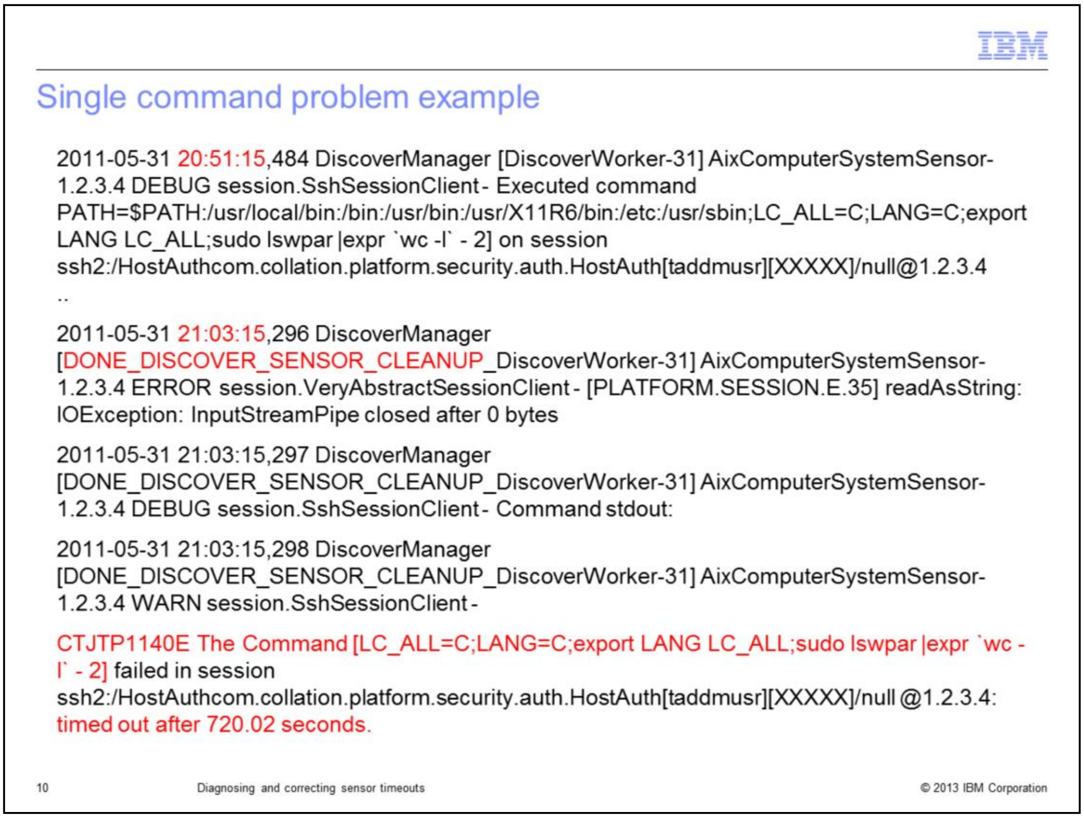
Look at the message and the messages immediately before and after it.

Assess whether the previous command occurred seconds before or several minutes before. If it was several minutes earlier, determine what command was run. If possible, run the command manually on the target to determine how long it takes. Determine if the output looks correct for the command.

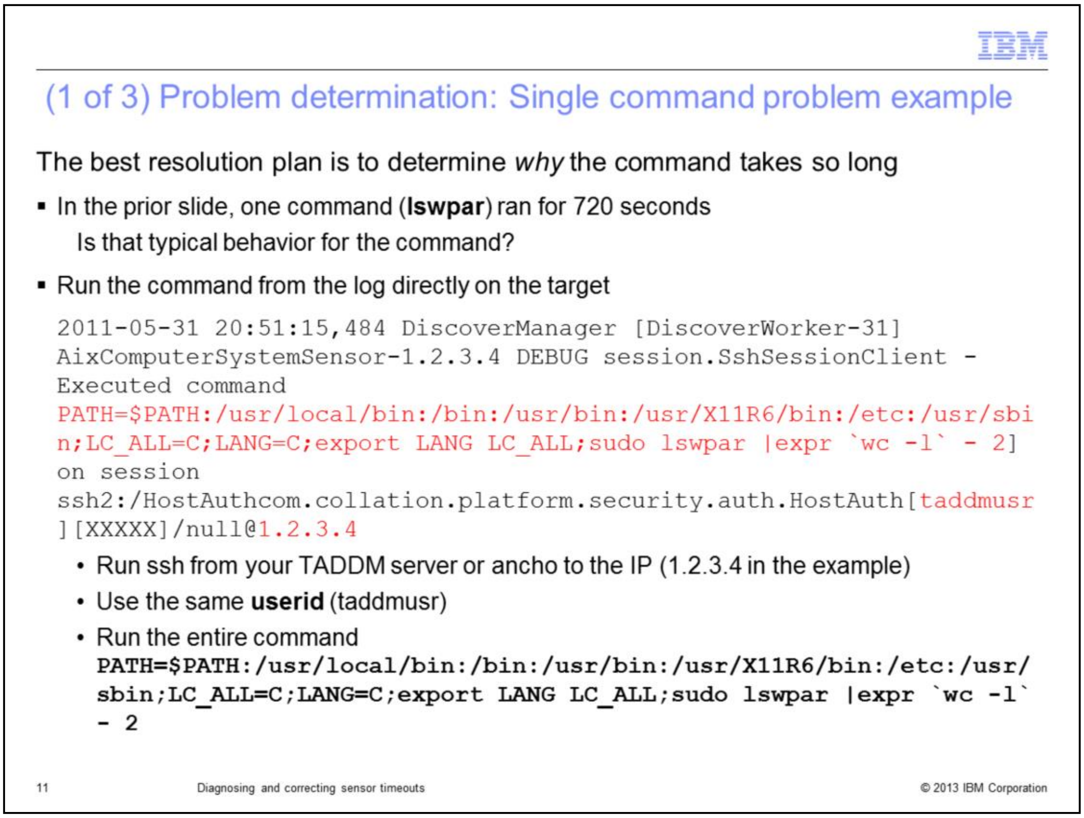
If there is no long-running command, check the duration of all commands. Determine whether the total duration exceeds the default timeout value, which is typically 10 minutes.

Determine if there are noticeable gaps in the log file where the software seems to be waiting to run a command.

Next the presentation shows some examples.



In this example, the sensor started at 20:51 and the timeout is at 21:03; it took 12 minutes to time out. When you examine the **DISCOVER_SENSOR_CLEANUP** message and surrounding ones, you see one command, **sudo lswpar**, timed out after 720 seconds or 12 minutes.



The best resolution to a single command that takes to time much to complete is to determine why it is taking so long, rather than increasing the timeout. Review the log entry that shows the command that timed out. The log entry indicates the full command and the UNIX® IP and login ID. The entries often start with PATH and include the user that ran it.

After you obtain this information, log in to the target directly with **ssh** that uses the same **user ID** as indicated in the log. Manually run the full command. Note how long it takes to run the command. On many UNIX systems, you can include **time** before the command to get accurate timing data.

(2 of 3) Problem determination: Single command problem example

- After the command completes, note how long it took and review the results.
- In this example, the actual command result when run manually was a password prompt, which the software perceived as the command not returning:

```
taddmusr@1.2.3.4$
PATH=$PATH:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/etc:/usr
/sbin;LC_ALL=C;LANG=C;export LANG LC_ALL;sudo lswpar |expr `wc
-l` - 2
```

System administrators give a typical lecture, which boils down to two points:

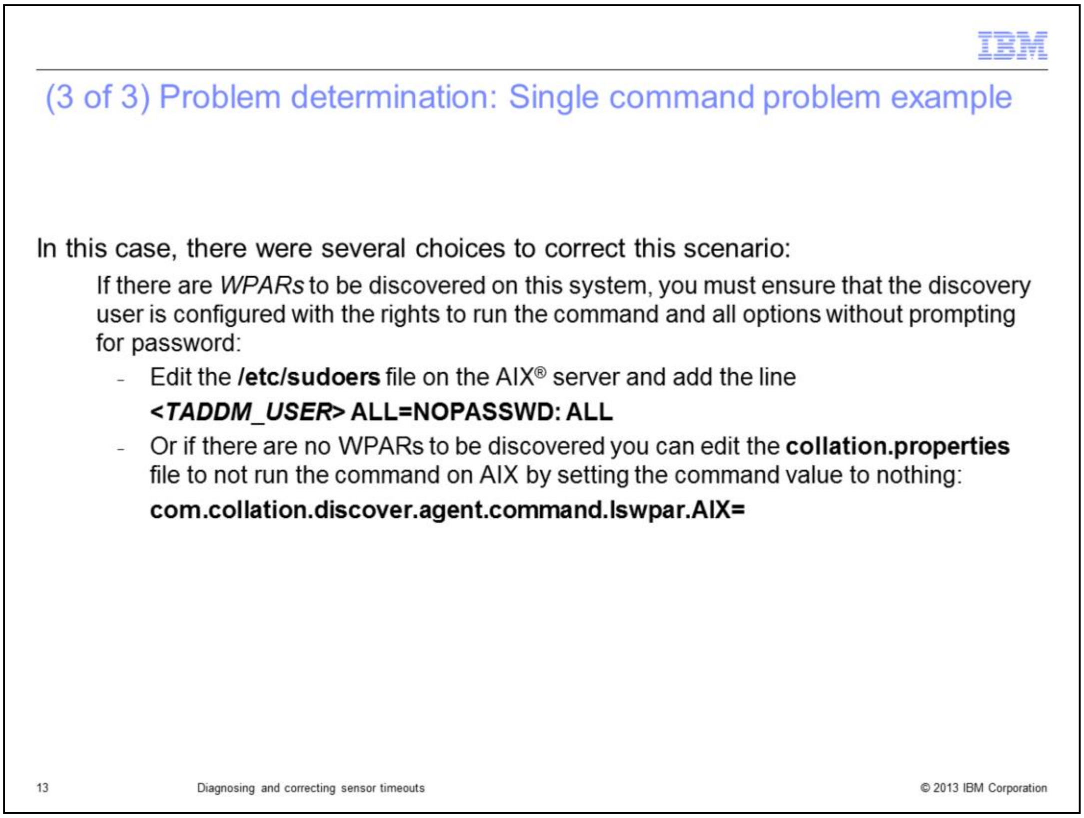
1. Respect the privacy of others
2. Think before you type

Password:

When you use **sudo**, a password prompt can be triggered

12 Diagnosing and correcting sensor timeouts

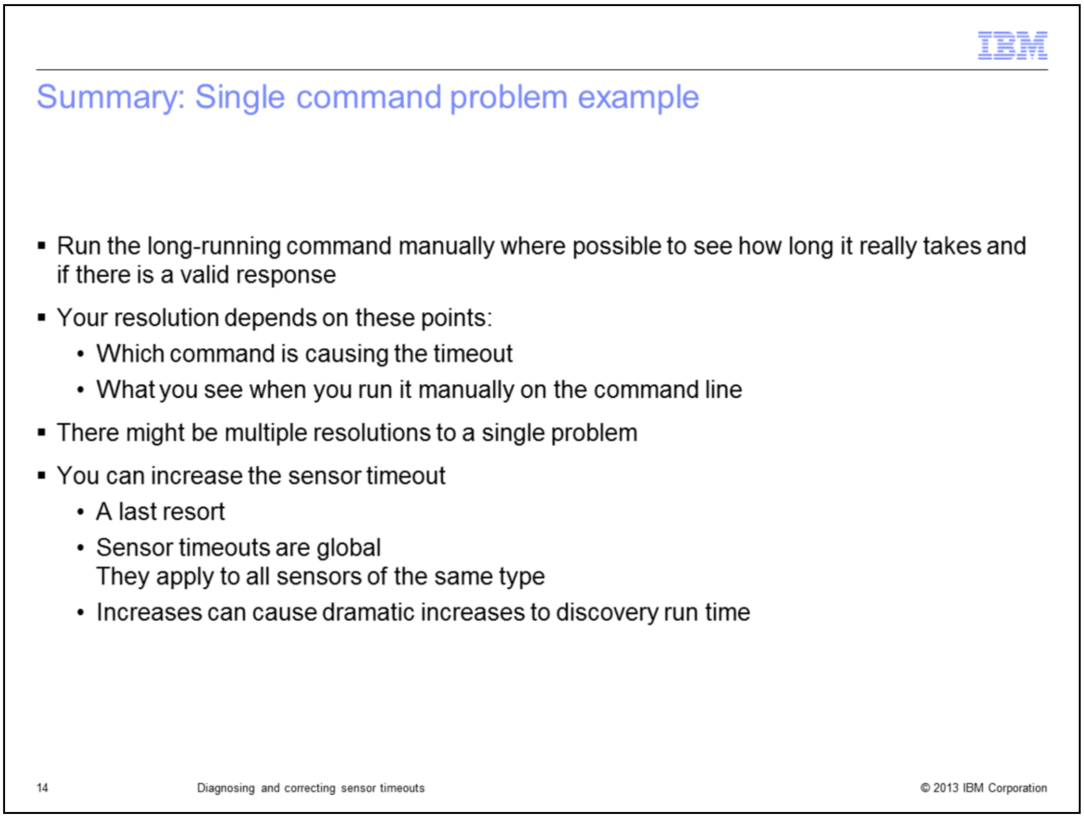
After the command completes, note how long it took and review the results. In this example, the command produced a **password prompt**, which is not an expected response but is often triggered when you use **sudo**.



In this example, there are two possible solutions to the timeout.

If there are WPARs to be discovered, then ensure that the discovery user can run the **lswpar** command without a password prompt by configuring NOPASSWD (no password) in the **/etc/sudoers** file.

If there are no WPARs to be discovered in your environment, you can set the **lswpar** command to **blank** in **collation.properties** file with the example shown so that no command is run. You can add a dotted decimal IP address to the end, to specify it for a single IP address rather than all AIX.



In summary, when you deal with a timeout caused by a single command, perform these steps:

1. Run the command manually and assess the time that it takes and the results.

2. Determine if you can correct the issue with the command.

3. The resolution can have multiple solutions; consider the best for your environment.

4. If absolutely necessary, increase the sensor timeout. Note that doing so can increase overall discovery time because sensor timeouts are global. All of the sensors of the same type are now allowed to run for a longer time and use a limited pool of discovery threads.

Consequences are covered in a few minutes.

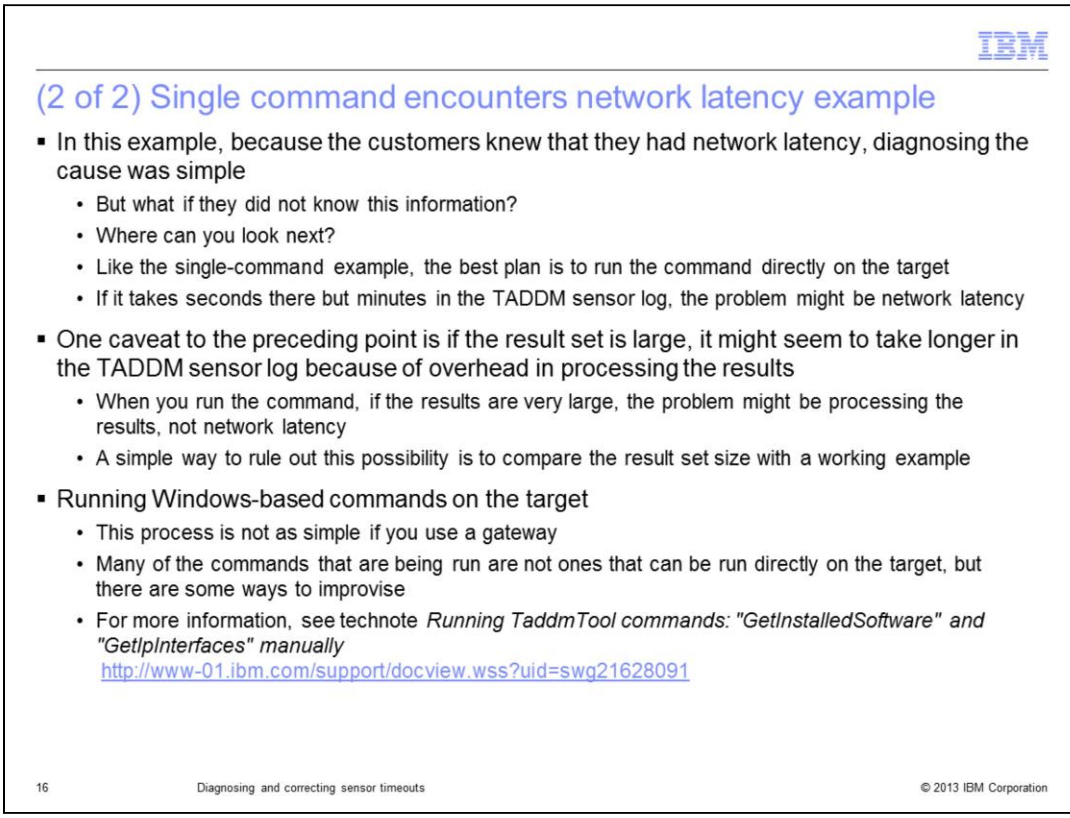
(1 of 2) Single command encounters network latency example

- Timeouts because of network latencies are similar to the prior single-command example
- You typically see commands that run in a few seconds on local servers that take minutes on servers that are geographically distant or have other latency issues such as firewalls
- Example:

```
[cmd.exe /c C:\WINDOWS/temp/taddm.buqz9z/TaddmTool.exe -
DTADDM_ID=buqz9z -DTADDM_USERNAME="taddmusr" -DTADDM_TIMEOUT=260
GetInstalledSoftware @4.5.6.7] failed in session
ssh2:/WindowsHostAuthcom.collation.platform.security.auth.WindowsHos
tAut h[taddm][XXXXX]/null@10.11.12.13:
SessionCommandTimedOutException: CTJTP1104E The command did not
complete. The command timed out after 260.935 seconds.
```

In this case, the machine 4.5.6.7 was in Asia while the discovery server and gateway server were in the United States.

Timeouts that are caused by network latency are similar to the previous single long running command example. You see commands that run in a few seconds typically on local servers take minutes to complete on servers that are geographically distant or show other latency issues such as firewalls. In this example, a WMI request from the Tivoli Application Dependency Discovery Manager gateway to the Windows host 4.5.6.7, timed out after 260 seconds. The customer stated that this server was located far away, in Asia and the Tivoli Application Dependency Discovery Manager gateway is in the United States. They also said that there were network issues at times with the connectivity.



In this example, you already know that there is a network latency issue that is based on customer description. The options to resolve this issue are presented on a later slide. But what if the latency issue is not known; where can you look next?

Like the single-command example, it is best to try to run the command directly on the target. But because this system is Windows, doing so is not as straightforward as for UNIX. See technote 21628091 for detailed instructions on running Tivoli Application Dependency Discovery Manager commands manually on Windows.

One caveat not mentioned in the previous example, is that there is more to the times you see in the Tivoli Application Dependency Discovery Manager log than the actual running of the command. The Tivoli Application Dependency Discovery Manager must receive the results and process them. If there is a large result set, the volume can add more overhead. When you evaluate the results of a command, keep the size of the results in mind.

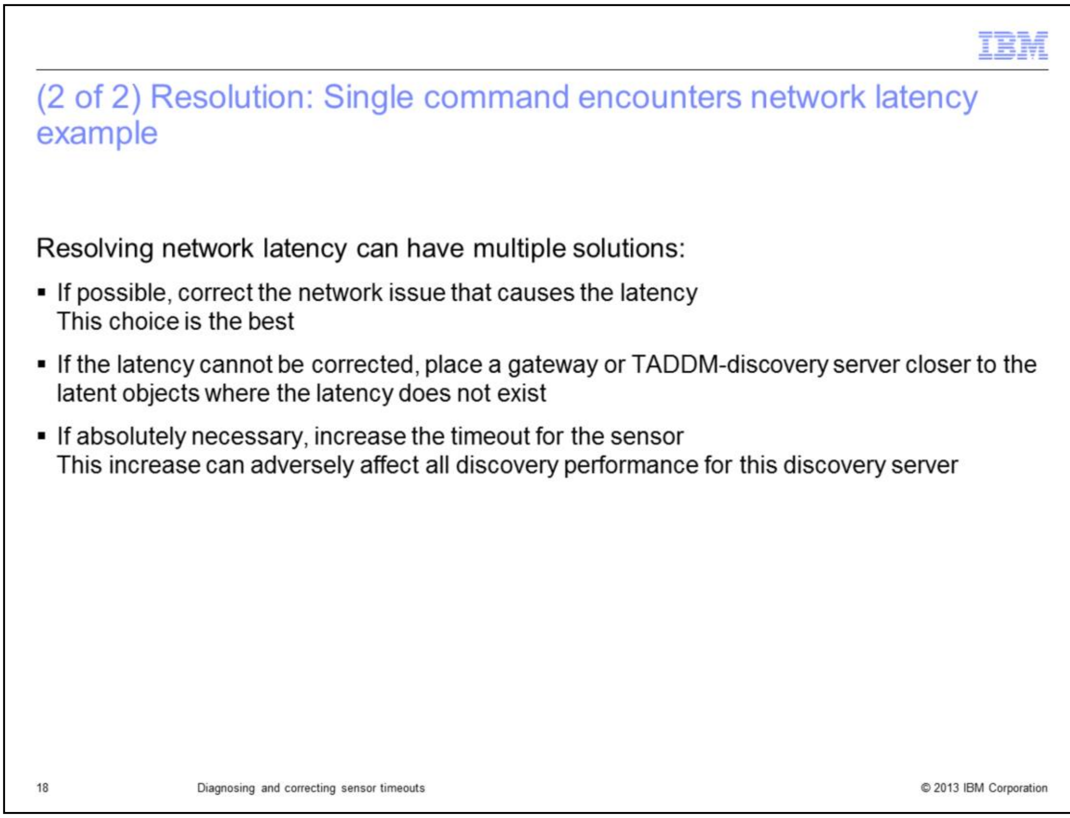
(1 of 2) Resolution: Single command encounters network latency example

- If you find that the command runs quickly on the target and with reasonable data returned, network latency can be the problem
  - True if it is much slower on the gateway
  - Try also simple tests like using file transfer protocol (FTP) to transfer a file between the target and the TADDM server and gateway, if applicable, and compare those durations to a working target
  - This test helps to isolate network latency without using any TADDM-specific calls
- If the gateway is almost as fast as the target, then it is more likely that the problem is between TADDM-server and the gateway or in the processing time of the result set

If the command runs quickly on the target and with reasonable data returned, the problem might be network latency.

You can also use simple tests like transferring a file between the target and the TADDM server and gateway, if applicable, with FTP and comparing those times to a working target. This test helps isolate network latency without using any TADDM-specific calls.

If you find that the gateway is nearly as fast as the target, the problem is more likely to be between the TADDM server and the gateway or in the processing time of the result set.



There are several paths to resolving network latency.

1. The first choice is to correct the latency issue so that commands return within a reasonable amount of time.

2. If the latency cannot be corrected, the next best choice is to put a TADDM discovery server or gateway closer to the target so that the sensor can gather the results more quickly. The data to be sent to the storage server is smaller, and latency is often not an issue with that transfer.

3. If absolutely necessary, increase the timeout for the sensor, but as noted earlier, timeout changes apply to all sensors of the same type, and increases can cause longer discovery run times.

This presentation shows information about timeout parameters later.

Ping sensor timeout example

2011-05-17 20:00:59,713 DiscoverManager [DiscoverWorker-12] PingSensor-1.2.3.4~1.20.3.4 INFO ping.PingSensor - !PingAgent.I.2!

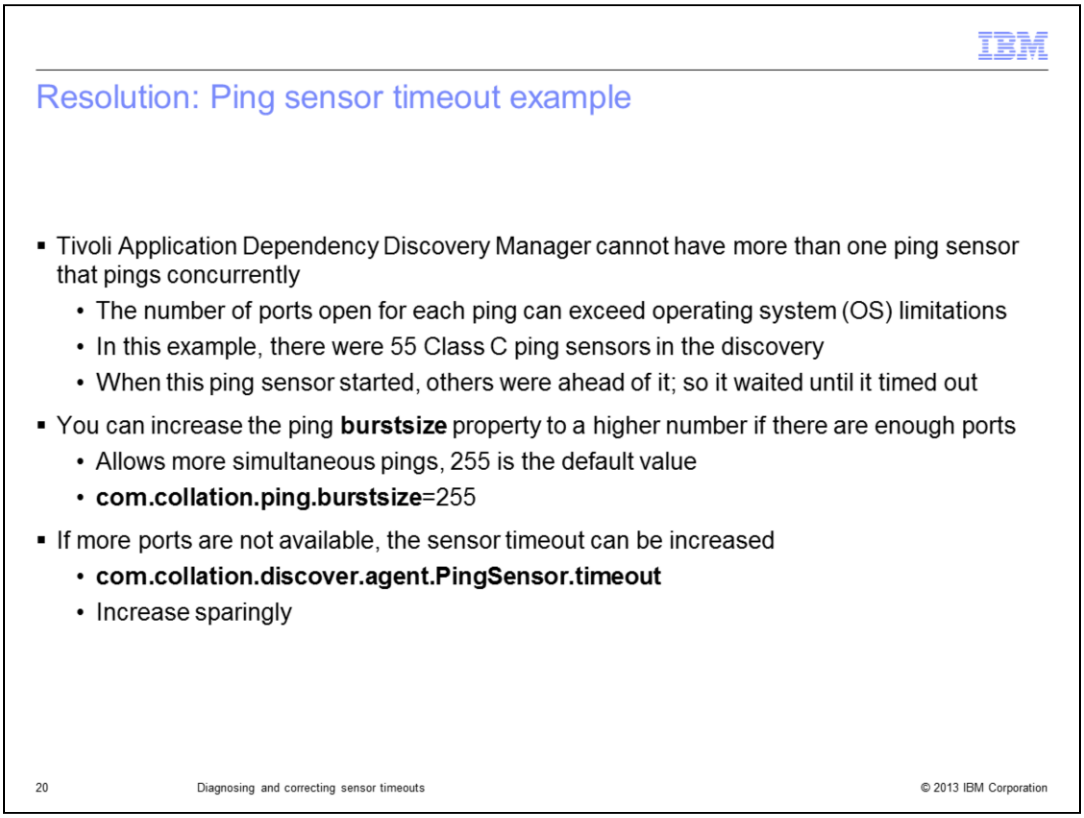
..

2011-05-17 20:10:59,724 DiscoverManager [DISCOVER_SENSOR_CLEANUP_DiscoverWorker-12]

PingSensor-1.2.3.4~1.20.3.4 DEBUG session.Ping - pinging 255 addresses on loop 0

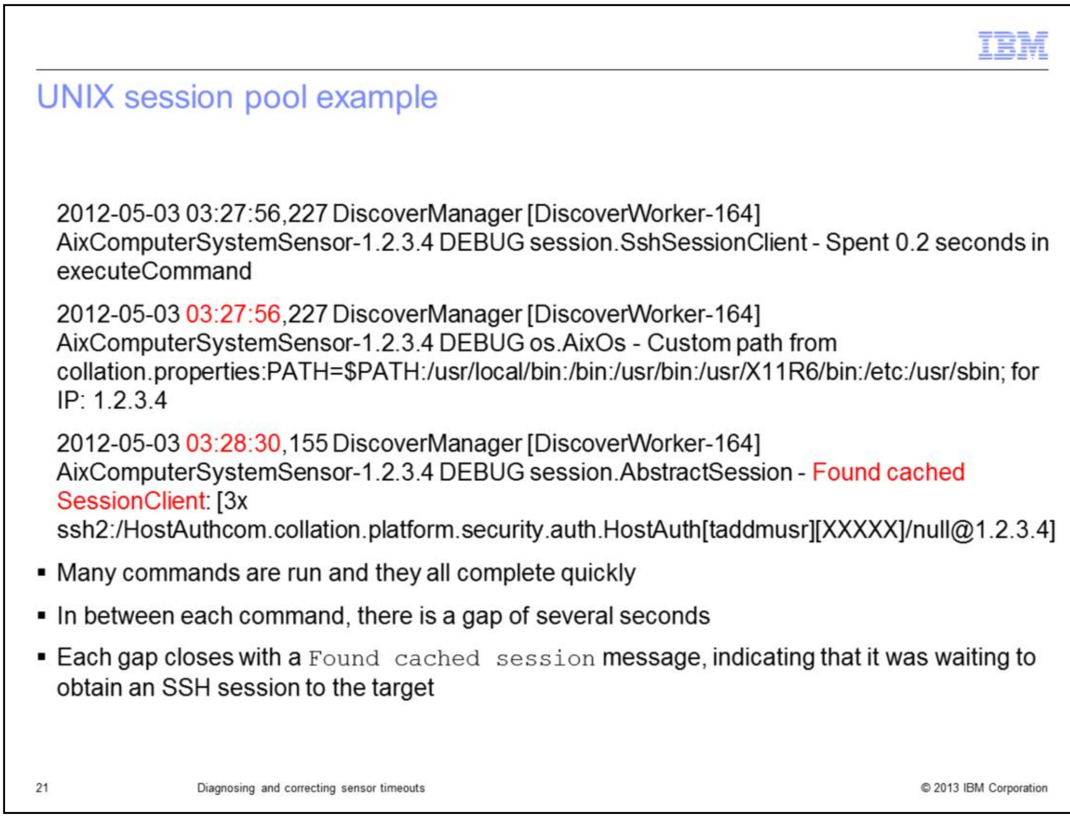
In this example, the Ping sensor did nothing for 10 minutes and then timed out

The next timeout example is specifically for the **ping sensor**. In this example, the ping sensor timed out after 10 minutes and looks like it did not ping any targets.



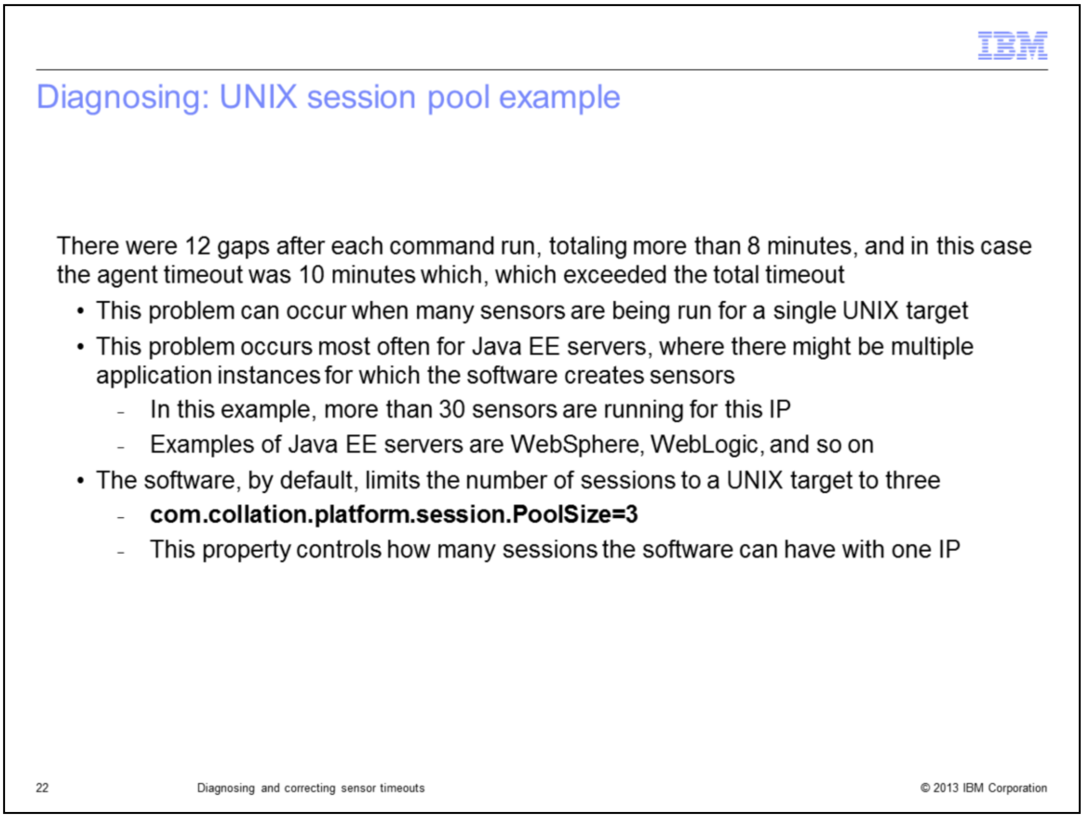
The **ping sensor** attempts to contact every IP in scope; often this action means that many ping sensors are created for the particular **runid**. Because of operating-system limitations, only one ping sensor can actively ping at a time. This limitation means the other ping-sensors must wait while each command completes. If there are many ping-sensors, some can time out.

You can control only the value of how many pings a sensor can run concurrently, the default number is 255. If your operating system permits more ports to be open, then you can increase the ping **burstsize** in the **collation.properties** file with the **com.collation.ping.burstsize** property. Setting this property can often resolve the timeout issue. If necessary, you can increase the ping sensor timeout slightly, perhaps to **15** minutes, instead of the default value of **10** minutes, if you cannot increase the number of pings.



The next example relates to how many SSH sessions Tivoli Application Dependency Discovery Manager can have for a single UNIX system at one time. By default, only three sessions are allowed to one UNIX IP at any one time. If you configure many sensors to run on a single computer concurrently, it can lead to timeouts while waiting for a session. This problem typically occurs on larger application servers.

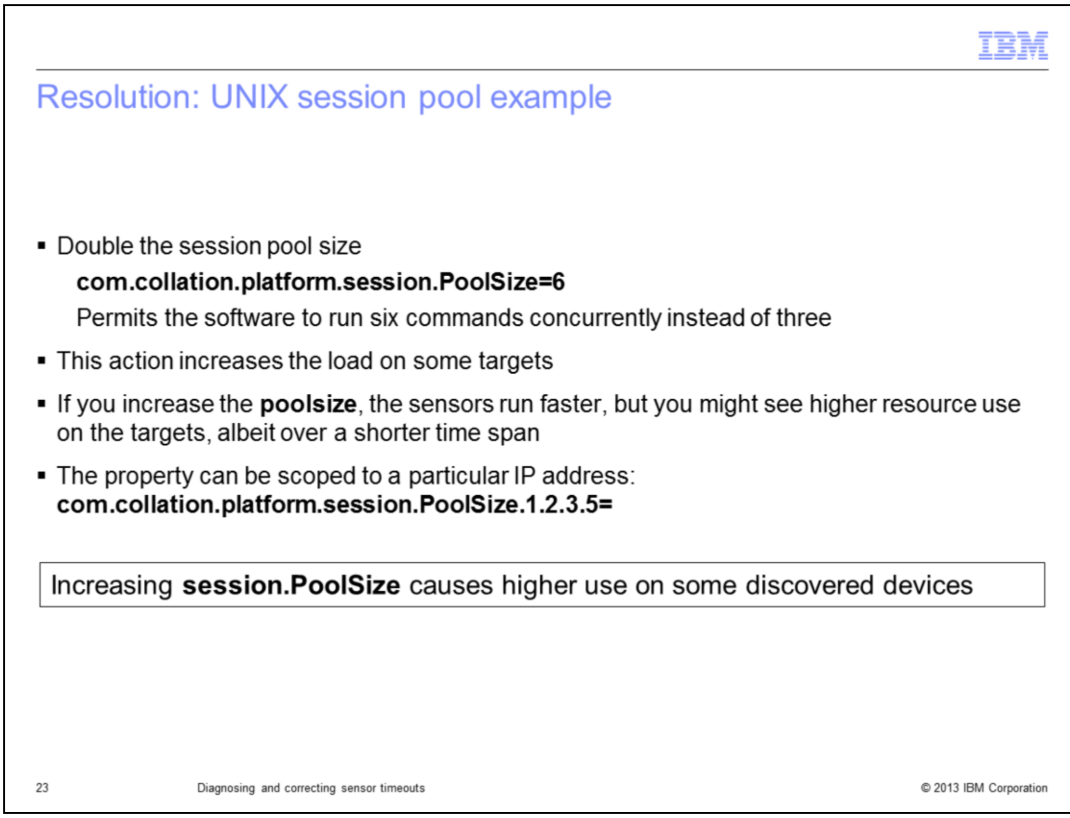
When this problem occurs, the symptom in the log file is gaps between the messages. In this example, you can see that the sensor ran a command and is preparing to run the next command, but there is a 34-second gap while waiting for that session. In this log file, this gap is repeated frequently, and the sensor then timed out after 10 minutes. You can see this gap by looking for the **Found cached SessionClient** messages and checking whether there is frequently a gap between it and the prior message.



In this example, you can identify 12 gaps that total more than 8 minutes, which caused the agent to timeout at 10 minutes even though the individual commands ran quickly.

This problem is more likely to occur on large application servers where the software is running more than 10 sensors on the single IP concurrently. This problem is more likely to occur in smaller discovery scopes since the smaller the scope, the more opportunities there are for a single IP to run multiple sensors concurrently.

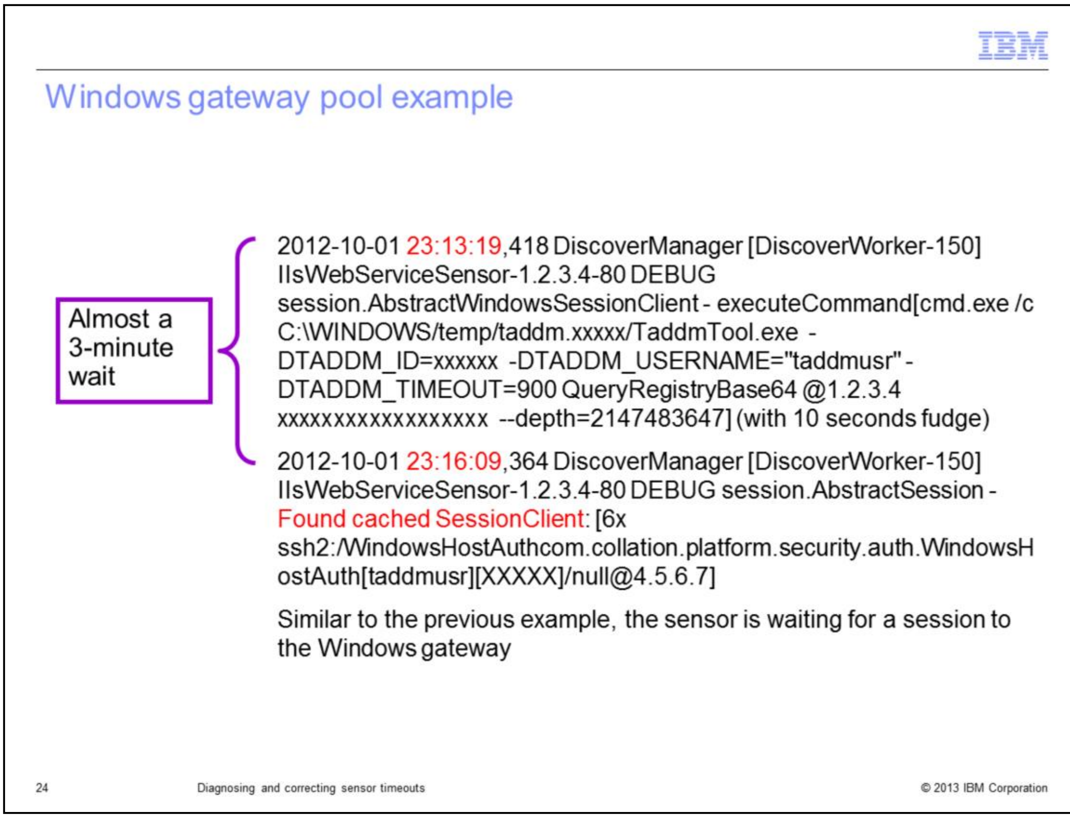
The software limits the total number of running sensors with the discovery worker property on the discovery server, otherwise known as **dwcount**. Normally the default value is 32 sensors can run at one time. However, many customers with enough resources, increase this value, which can lead to running many more sensors for a single IP than typically seen in a smaller environment. When more sensors run, the **poolsize** property throttles the sessions to a single IP to ensure that the software does not use too many resources on the target. As previously mentioned, the default value is 3 sessions at one time. So if 20 sensors are running for an IP, that means 17 are waiting while the others run.



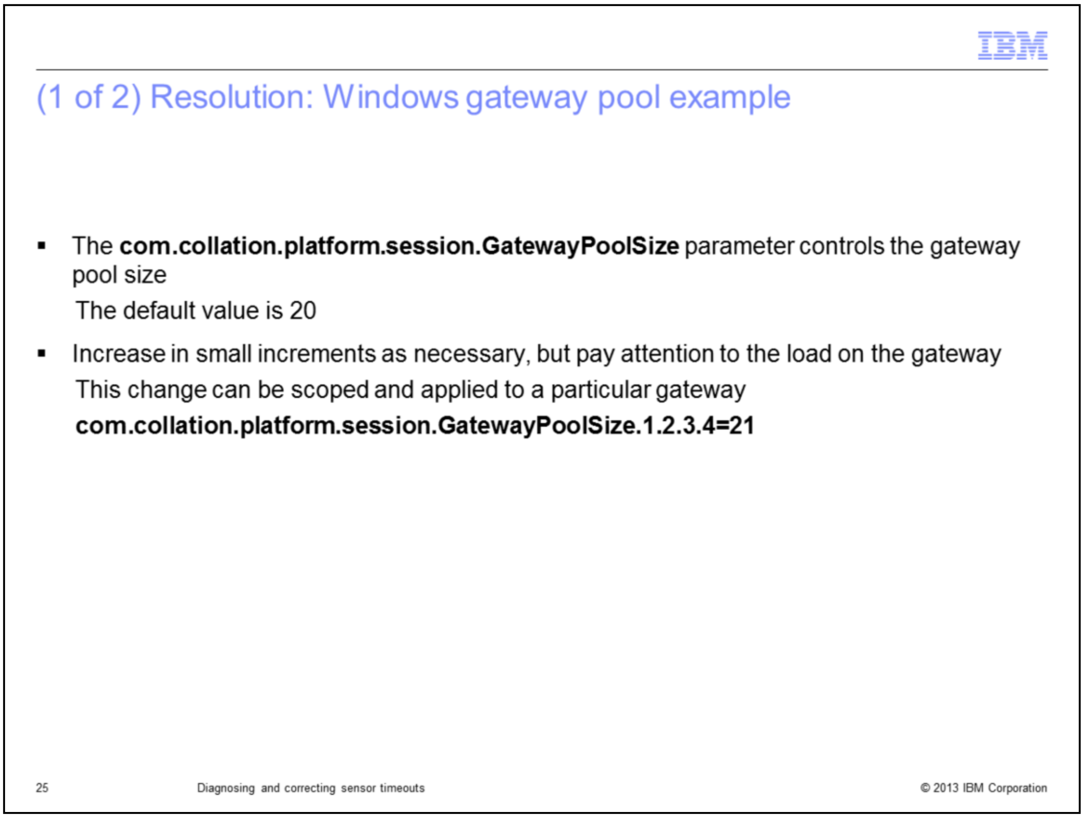
The best resolution to this issue, if it occurs frequently, is to increase the **poolsize** so that more sessions can run concurrently on the target. However, increasing the pool size means that you can run more commands concurrently and therefore increase the use of the target. If possible, scope the property to the addresses that display this issue so that you have control over the increased use.

Additionally, larger scopes and lower **dwcount** can make this problem less likely to occur. The reason is because the likelihood of the same IP running many sensors concurrently decreases as the scope size increases and the thread count is lower. There is a fine balance, since a higher **dwcount** can often lead to improved discovery times. If the higher **dwcount** causes timeouts, and you do not want to increase target use with **poolsize**, you might have to decrease **dwcount** to limit the number of sensors that run concurrently.

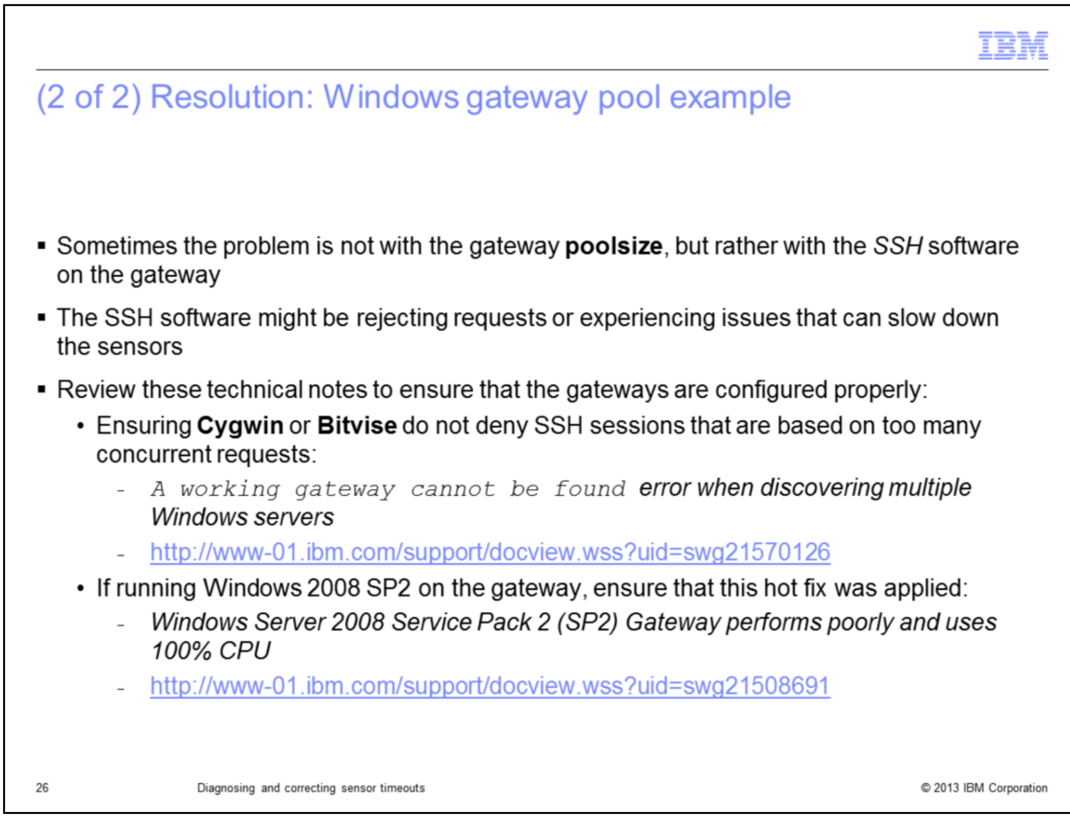
IBM does not recommend sensor timeout increases for this issue because the sensor that fails can vary, and given the global nature of sensor timeouts, increasing several can have a large impact on discovery performance.



The prior example was for UNIX; what about Windows? You can see the same issue here in a Windows log file. Note the almost 3-minute wait for the session in the log messages on this slide. In this example, the discovery server is waiting for an SSH session to the Windows gateway.



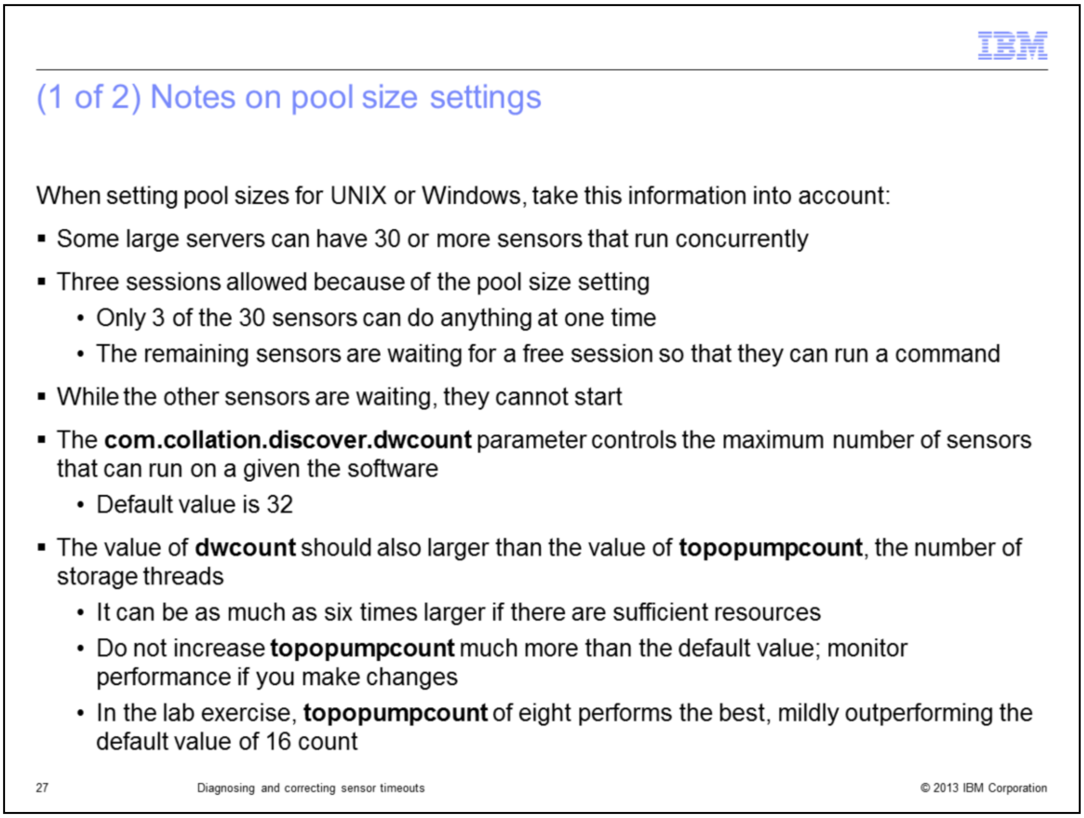
A **poolsize** parameter **com.collation.platform.session.GatewayPoolSize** also controls the number of sessions to the Windows gateway. The default value is 20. You can set one value for each gateway by scoping it to the gateway IP or setting one value that applies to all gateways. You can increase this size, but you must monitor the processor and memory on the gateway before and after any changes to ensure that you do not overuse it. **TADDMTool** is very processor-intensive, and increasing the number that can run concurrently can cause the gateway to perform poorly, resulting in more, not fewer, timeouts. You should also monitor the SSH software logs to ensure no errors as you increase this property. Do not excessively increase this property; it should be lower than **dwcount**.



Also, note that by default, most gateways do not accept even 20 sessions at the same time. Often these sessions are considered as attacks and can be denied by the SSH software.

There are links to two technotes shown. The first one describes how to configure Cygwin and Bitvise so that sessions are not denied based on too many requests at once. The second one relates to Windows 2008 SP2 gateways running **getInstalledSoftware**. If you are running Windows 2008 SP2 on the gateway, be sure to apply the fix that is indicated in the technote.

It is wise to monitor the **cygwin/bitvise** log file on the gateway for any changes to ensure that increased traffic does not create other errors.

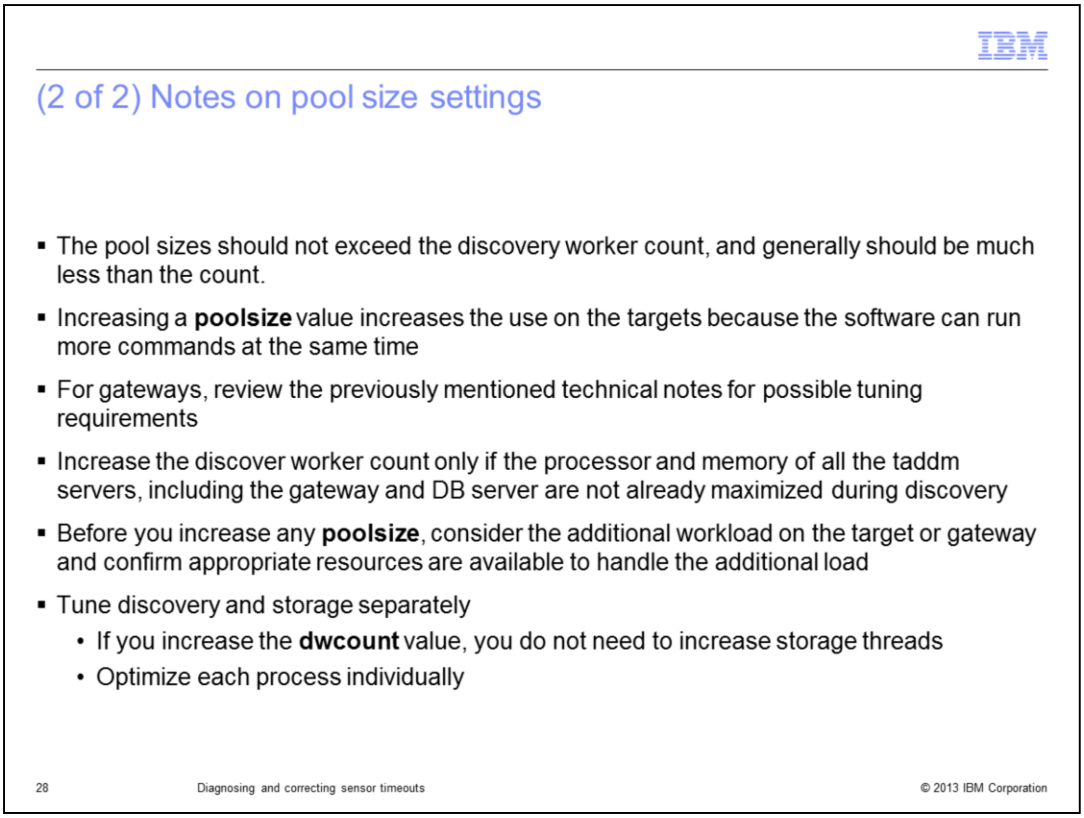


When you set pool sizes for UNIX or Windows, be aware that there is a relationship between the pool size, the size of the discovery, and the discover worker thread count or **dwcount**.

If you discover a computer system with many applications on it, the software might run 30 or more sensors concurrently in the case of a small discovery or a large **dwcount**. A **poolsize** that is set to 3 means that only 3 of those sensors can actively work at one time; the other sensors wait and no new sensors can start.

Some customers increase the **dwcount** value to improve sensor throughput when available resources on their discovery servers, gateways, and anchors are present. If you increase the **dwcount** value, you do not increase the storage threads; typically 16 storage threads can saturate a storage server. Lab testing confirmed that the lower the storage thread count, the better the performance, with 8 being the optimal setting. You set storage threads with the **com.collation.discover.observer.topopumpcount** property.

When you increase any thread count, be careful to monitor the software system resources (processor, memory, and IO statistics) to ensure that the changes do not overload the available resources.



To summarize the pool sizes, perform these steps:

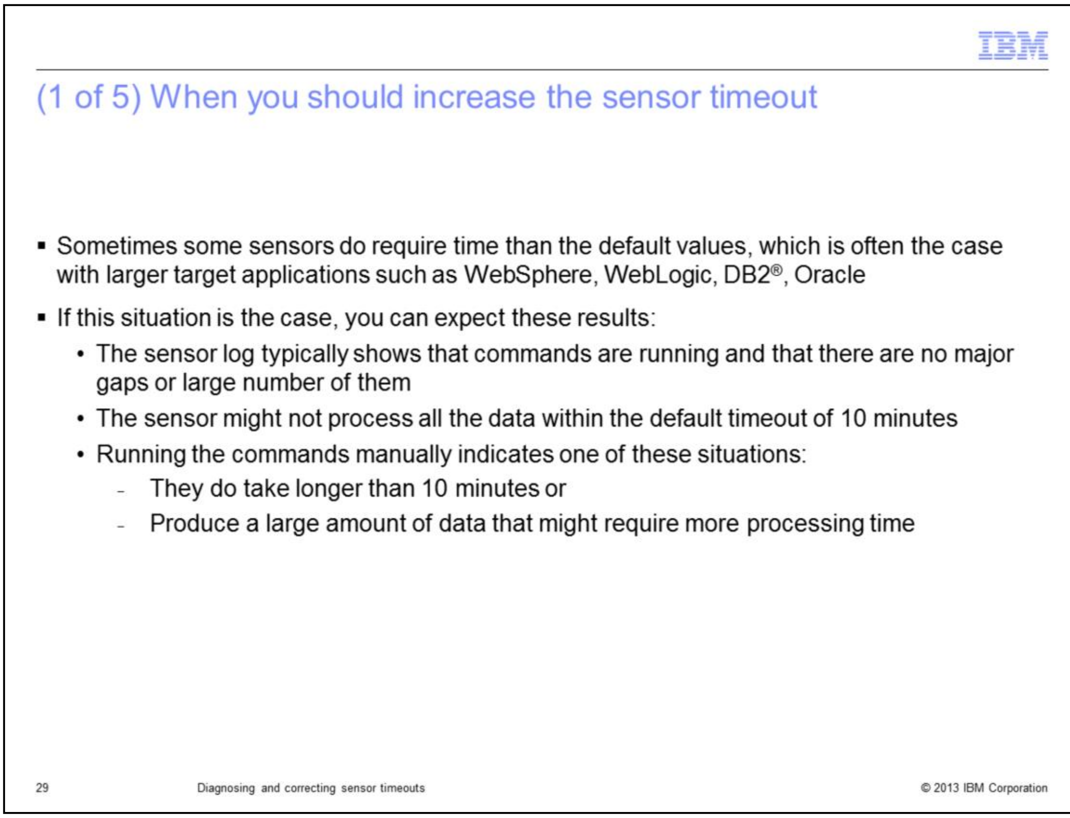
1. The total of the pools should not exceed the **dwcount**, which is the total of all sensors.

2. Increasing the **poolsize** can increase the concurrent load on a single target because it can run more commands together.

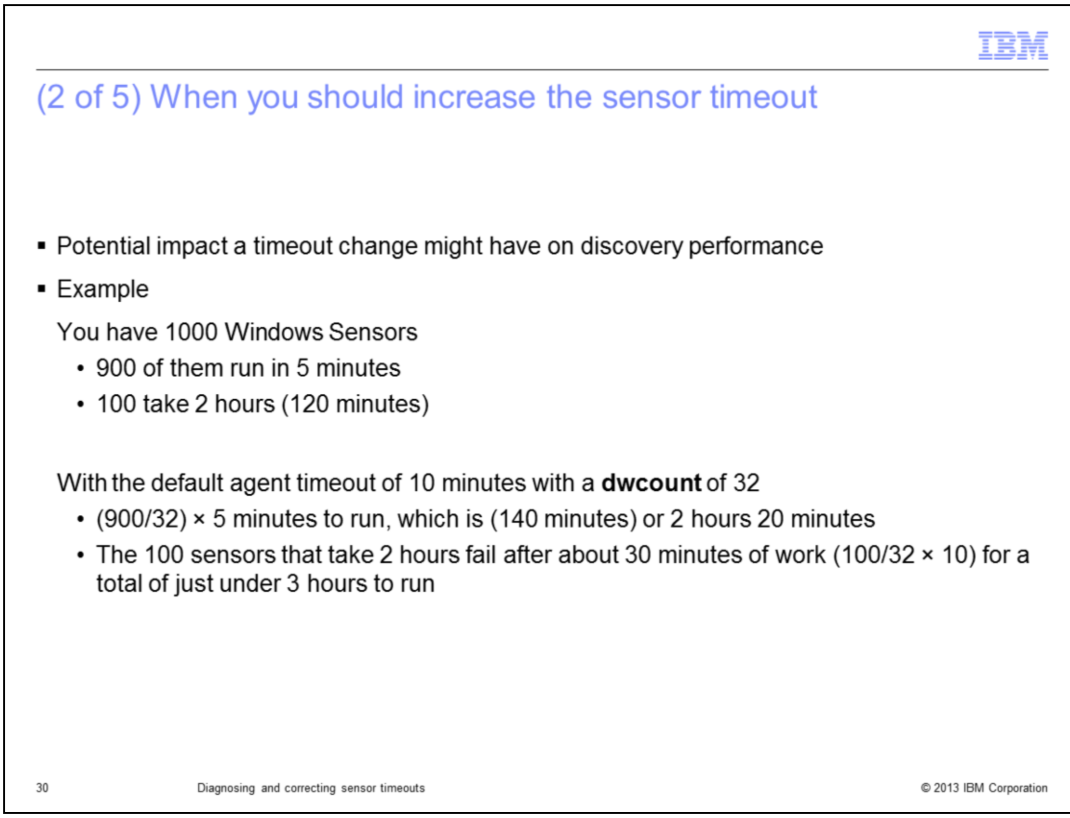
3. Review both of the technotes that regard tuning the gateway and make any appropriate changes.

4. Increase the **dwcount** only if you evaluated that you have enough resources on the discovery server, anchors, and gateways.

5. If you increase the **dwcount**, it is not necessary to increase the storage threads; typically 16 storage threads can saturate a storage server. Increasing the **topopumpcount** is risky and can lead to unpredictable results.



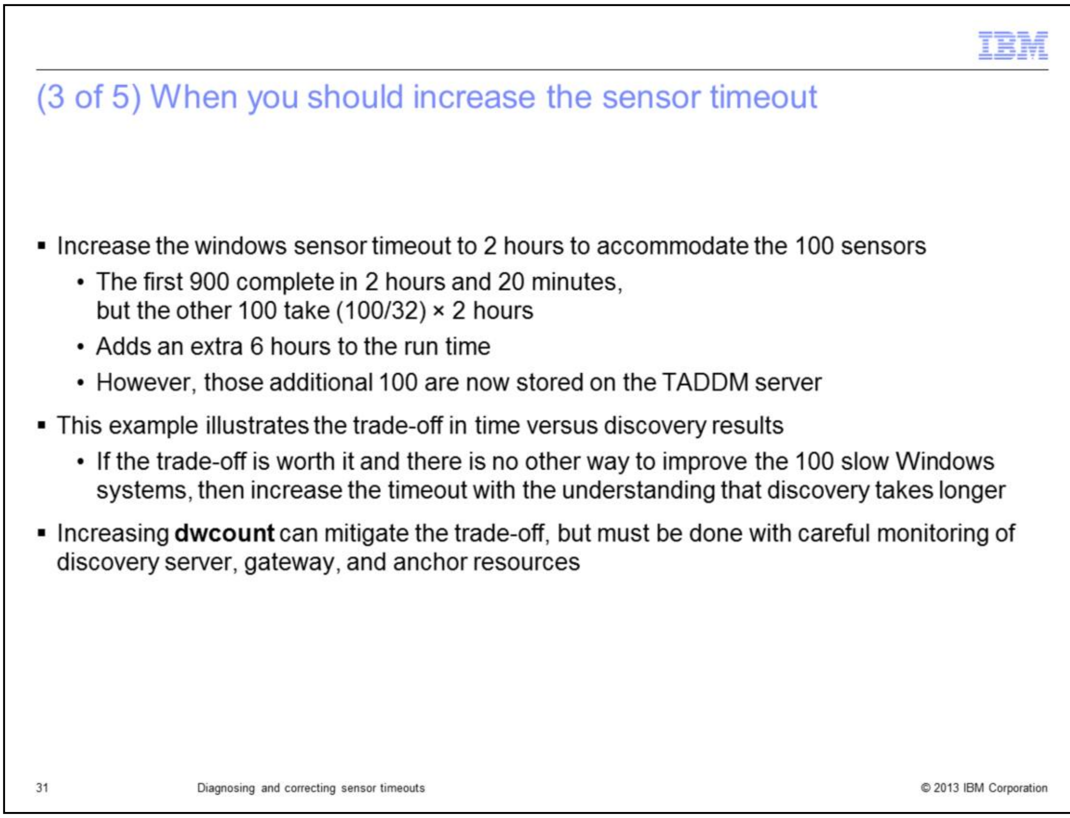
Now that you went through several examples, you are ready to learn about when to increase a sensor timeout value. Sometimes it is necessary, especially for larger applications like WebSphere or large database servers. When this problem occurs, the log file typically shows that the commands are all running. Some commands might take a few minutes, but the results set is large and expected. There are not many gaps of time in the log file to indicate a pooling issue. If so, then increasing the timeout value might be the only option.



Before you increase the timeout value, look at an example of the potential discovery time trade-off that occurs when timeout values are increased.

For example, if you run 1000 Windows sensors and 900 of them take 5 minutes each to discover, but 100 take 2 hours each, what is the difference in discovery time if you increase the timeout from the default value of 10 minutes to 2 hours?

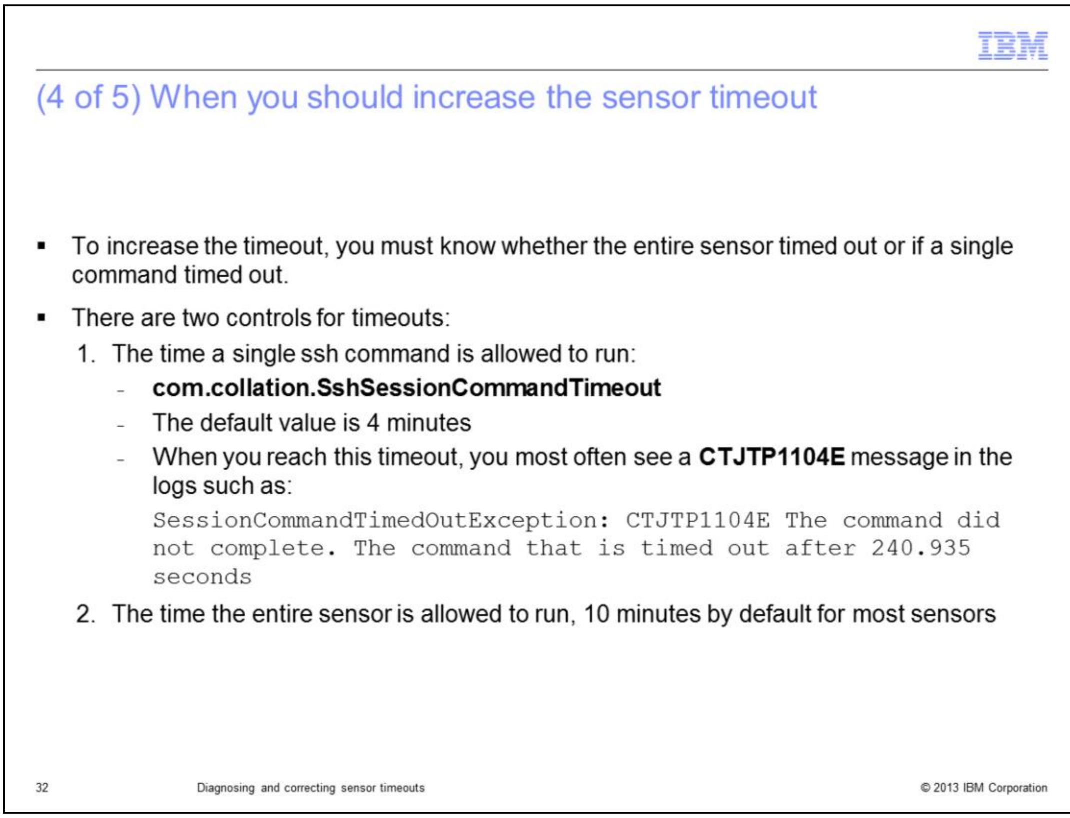
If the discovery worker count is 32, that means the 900 sensors take 900 divided by 32 threads times 5 minutes to run. This duration is approximately 2 hours and 20 minutes. The 100 sensors that take 2 hours fail to discover if the timeout is 10 minutes; it takes 30 minutes more for a total of 3 hours. These 100 Windows computer system are not in Tivoli Application Dependency Discovery Manager.



If the timeout is increased to 2 hours, the 100 Windows computer systems can be discovered. The first 900 sensors still take 2 hours and 20 minutes, but the remaining 100 takes 100 divided by 32 threads times 2 hours to discover, or an extra 6 hours. The initial discovery run that took 3 hours now takes almost 9 hours to complete after the timeout value is increased. However, the 100 sensors that previously failed are now stored in Tivoli Application Dependency Discovery Manager.

This example is basic; it does not account for the threads that are in a more mixed batch. Increasing the discover worker count value can mitigate the performance decrease, but more memory can be used to do so along with correct gateway, anchor, and **poolsize** tuning.

This example illustrates the trade-off of time versus discovery results. The trade-off might be worth it if there is no other way to improve the 100 slow Windows systems, but increase the timeout value with the understanding that discovery must take longer.

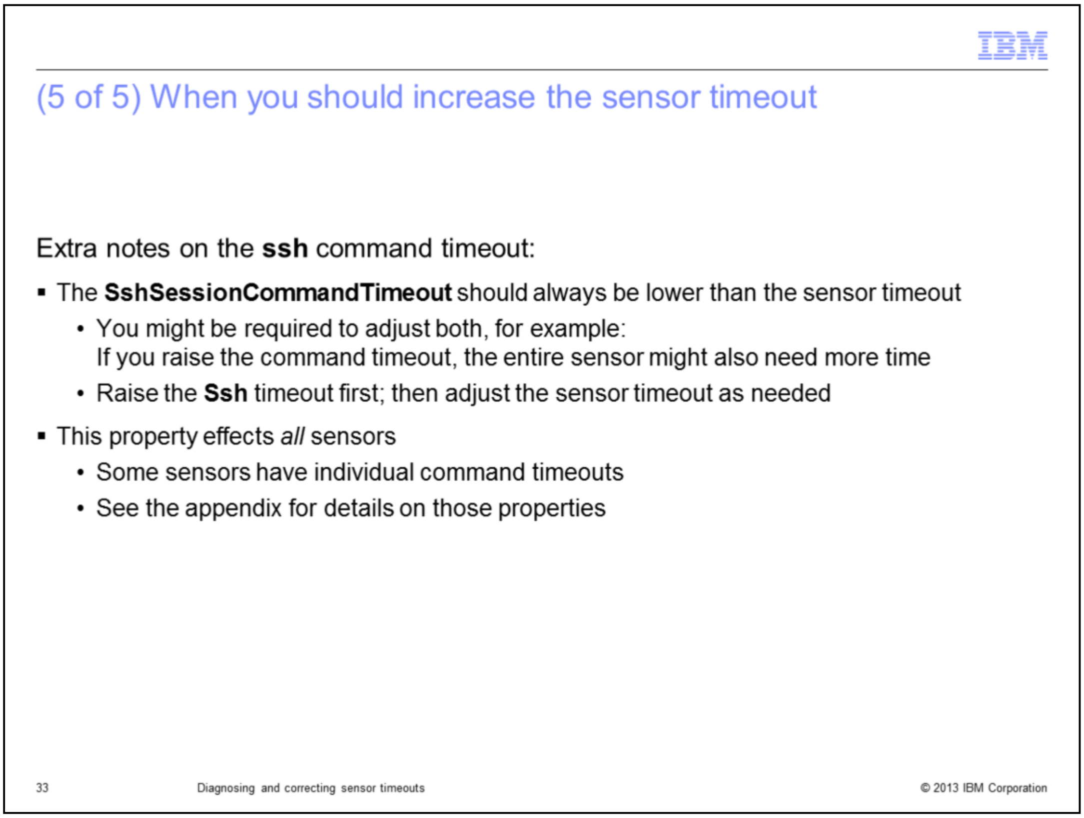


To increase the timeout, you must know if the entire sensor timed out or if it was a single command that timed out. There are two controls in **collation.properties** for timeouts:

> 1. The ssh session command timeout.
> By default the value is 4 minutes and is the maximum length of time a single command can take to return results. This timeout often manifests as a CTJTP1104E message that is shown in some of the previous examples.
>
> 2. The sensor itself times out, all the commands run finished within 4 minutes, but the sum of them exceeded the default 10 minutes.

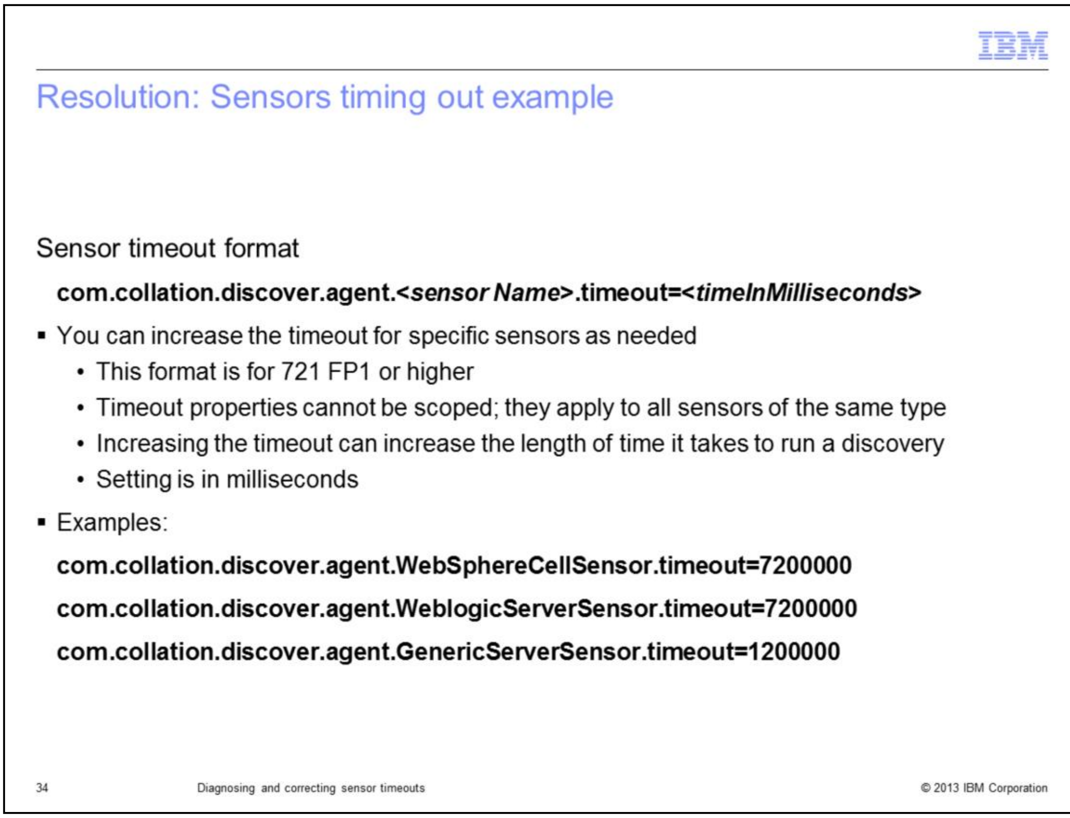
If you determined that an **ssh** command is taking more than 4 minutes and you cannot correct it, you should increase **com.collation.SshSessionCommandTimeout** value appropriately, but not greatly, because it affects *all* sensors.

Syntax discussion is next.



Often if you must increase the **ssh** command timeout duration, you must increase the sensor total timeout value too. If one command runs for 8 minutes, then probably the entire sensor needs more than 10 minutes to complete.

This property affects all sensors. Some sensors have individual command timeouts. Refer to the appendix for details on those properties.



To increase sensor timeouts, follow the format on this slide, where sensor name is typically the name you see within the sensor log file. This format applies to Tivoli Application Dependency Discovery Manager V7.2.1 FP 1 and higher. Before this fix pack, some sensors used *Sensor* and some used *Agent* in the sensor name field. After Fix Pack 1, use *Sensor*. However, *Agent* might still work for older sensors. The time is in milliseconds.

Again, remember that sensor timeouts apply to all sensors of the same type. You cannot use the IP address to scope this property. Increasing the sensor timeout can increase discovery time. For example, if you have five slow WebSphere sensors and increase the timeout to 2 hours, they actually take that long. So another sensor cannot use five of the default 32 threads (**dwcount**) for 2 hours.

IBM

Default sensor timeout

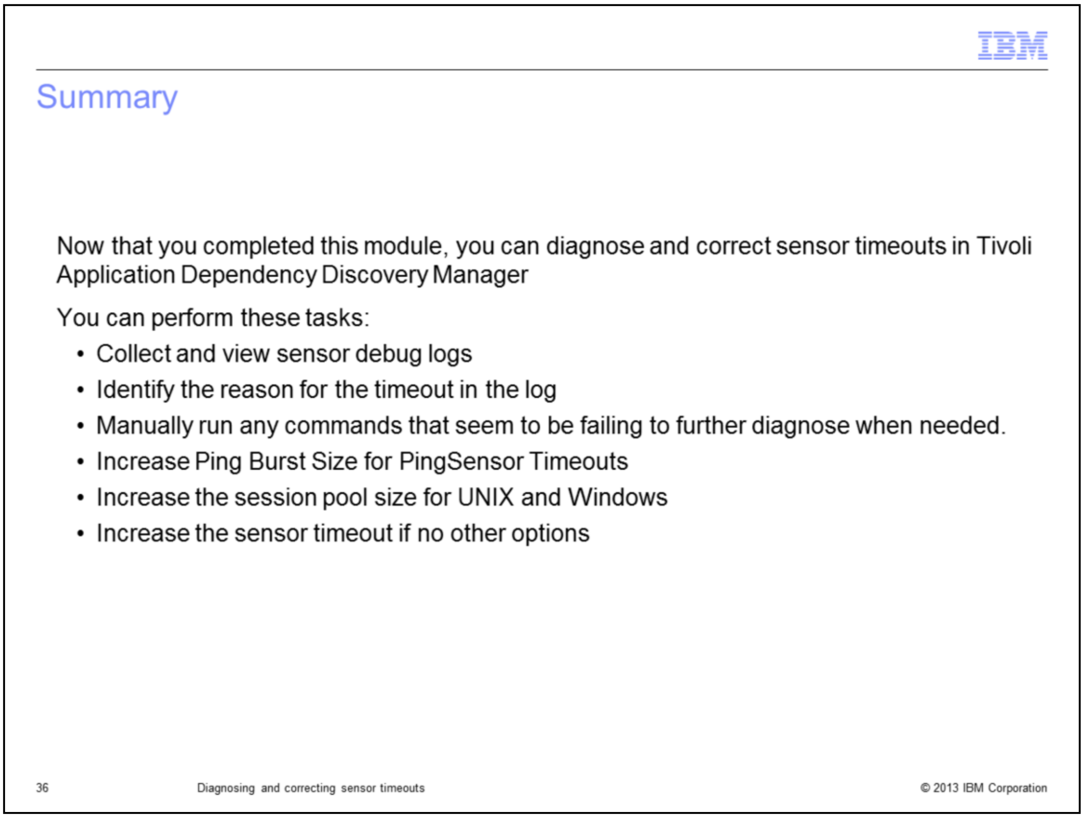
com.collation.discover.DefaultAgentTimeout=600000

- The default sensor timeout default value is **10** minutes
- 600,000 milliseconds equals 10 minutes
- *Never change this value*
  - Property applies to all sensors that do not have a specific timeout
  - Increasing the property value can cause *large increases* in discovery performance time
- Always change only sensor-specific timeouts, if necessary, to avoid introducing performance issues

35 Diagnosing and correcting sensor timeouts

There is a default agent timeout of 10 minutes that you should never change because it is the value that is used if no other sensor timeout is coded. Increasing the value means that every sensor can run longer and can have a much larger effect on discovery performance than if you change a single sensor.

If you must change a timeout value, change the specific sensor only, or the ssh timeout value.



Now that you completed this module, you can diagnose and correct sensor timeouts in Tivoli Application Dependency Discovery Manager.

You can:

- Collect and view sensor debug logs
- Identify the reason for the timeout in the log
- Manually run any commands that appear to be failing to further diagnose when needed
- Increase Ping Burst Size for PingSensor Timeouts
- Increase the session pool size for UNIX and Windows
- Increase the sensor timeout if no other options

Trademarks, disclaimer, and copyright information