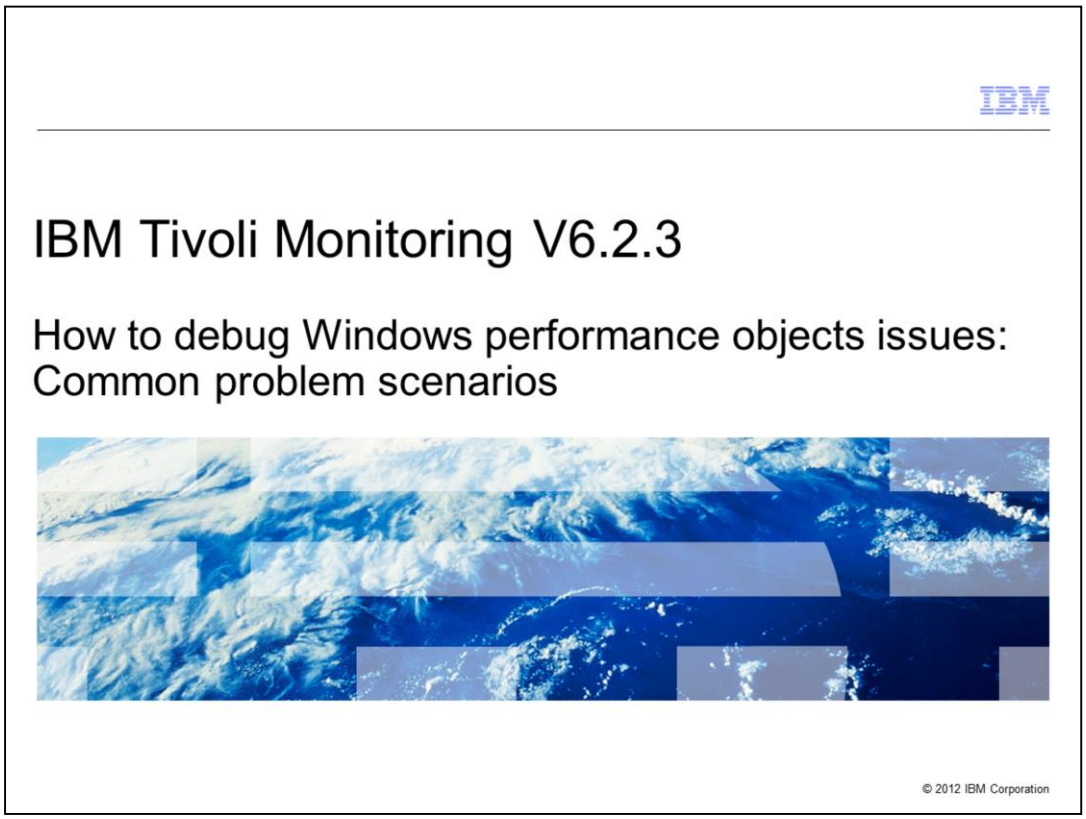
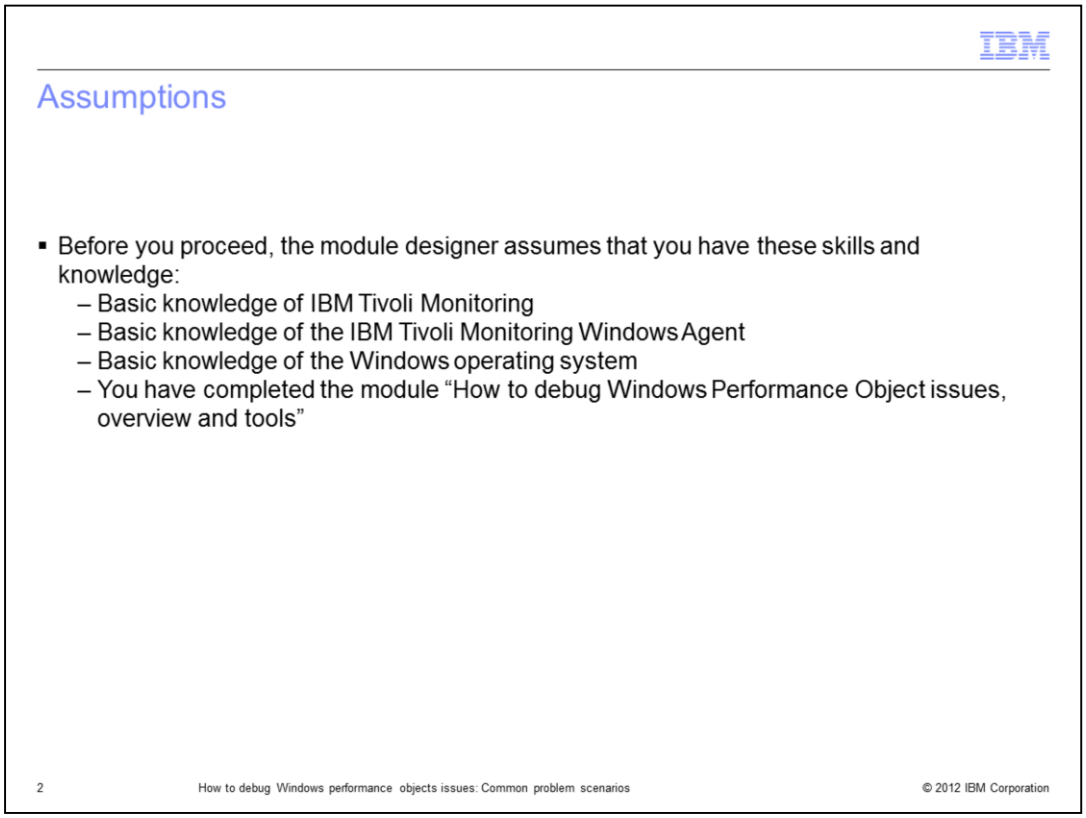


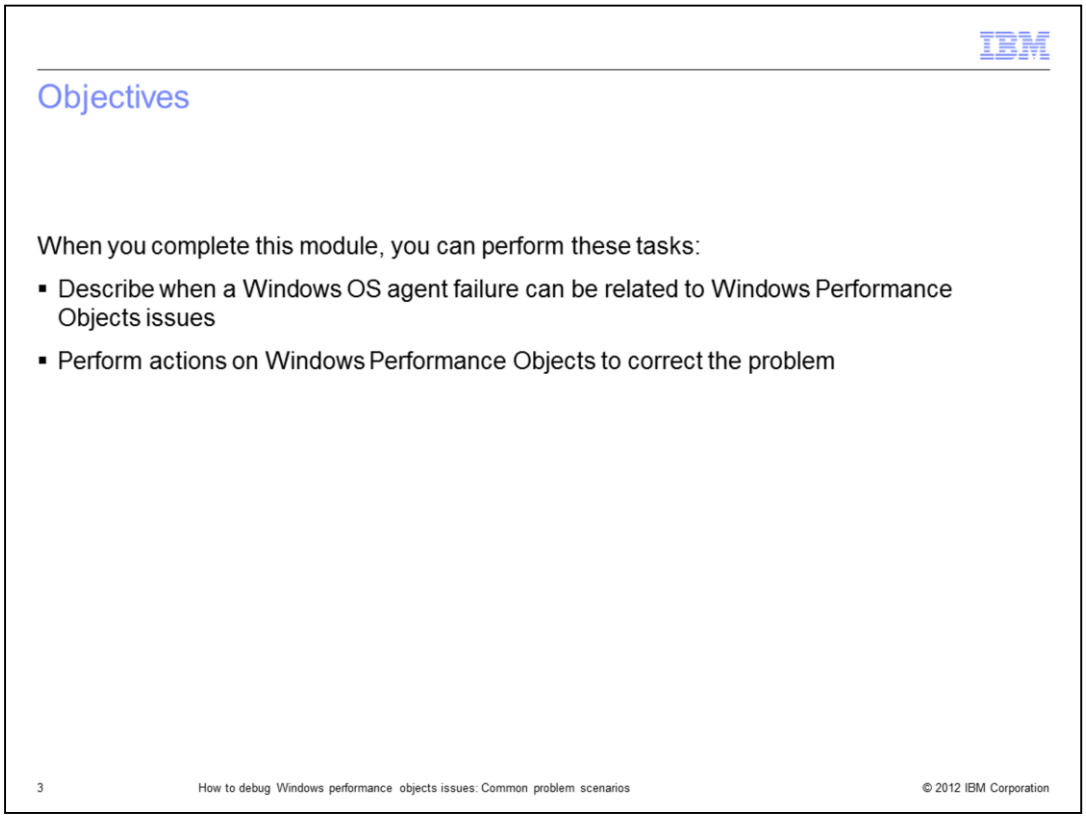
IBM Tivoli® Monitoring V6.2.3, how to debug Windows® performance objects issues, common problem scenarios.



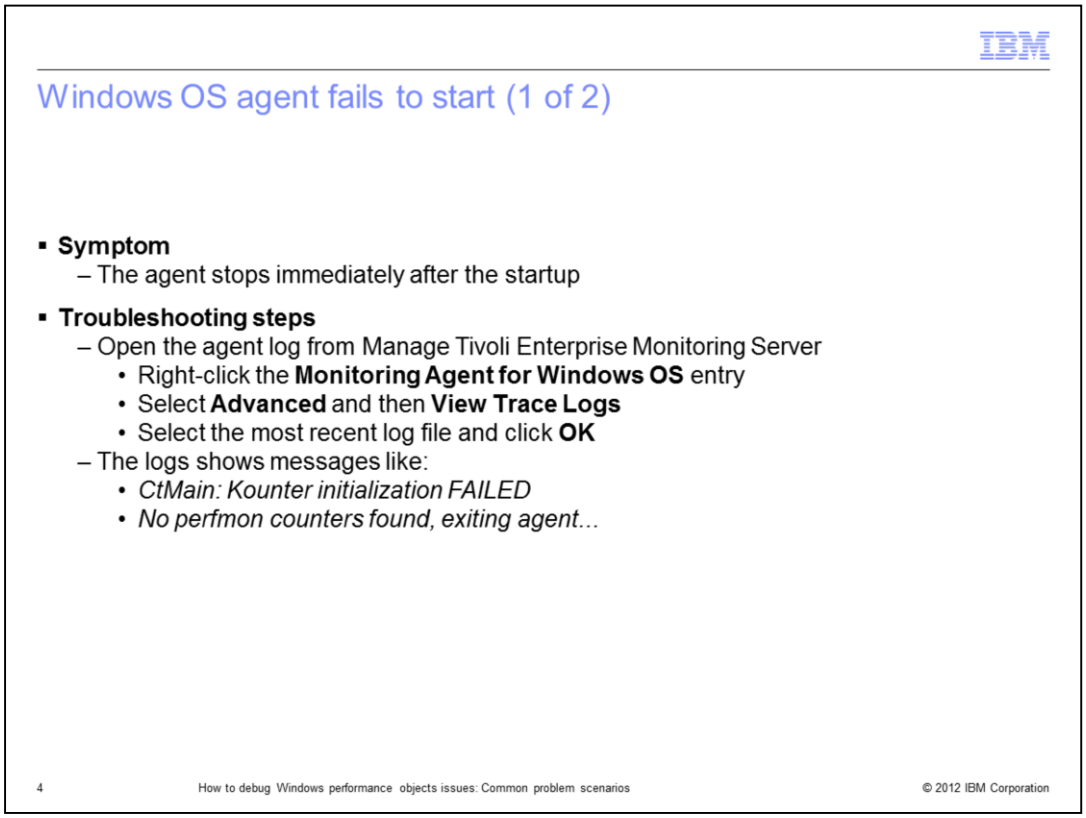
The module developer assumes that you have an understanding of basic IBM Tivoli Monitoring concepts and have a basic knowledge of the Windows OS agent.

It is also useful if you have a basic knowledge of the Windows operating system so that you understand Windows-specific concepts like the Windows Performance Objects libraries.

In order to effectively follow this module, first complete a module called “How to debug Windows Performance Object issues, overview and tools”. This module provides an introduction to Windows Performance Objects and the suggested tools for troubleshooting their problems.



When you complete this module, you can describe when a Windows OS agent failure can be related to Windows Performance Objects issues. You can also perform actions on Windows Performance Objects to correct the problem.



In this module, you see the most common problems that are related to the interaction between the Windows OS agent and the performance monitor libraries.

The agent might fail to start, stopping a few seconds after you tried to start it.

There might be several different reasons for the agent process to close unexpectedly, but in this module, you see only the one related to performance monitor libraries.

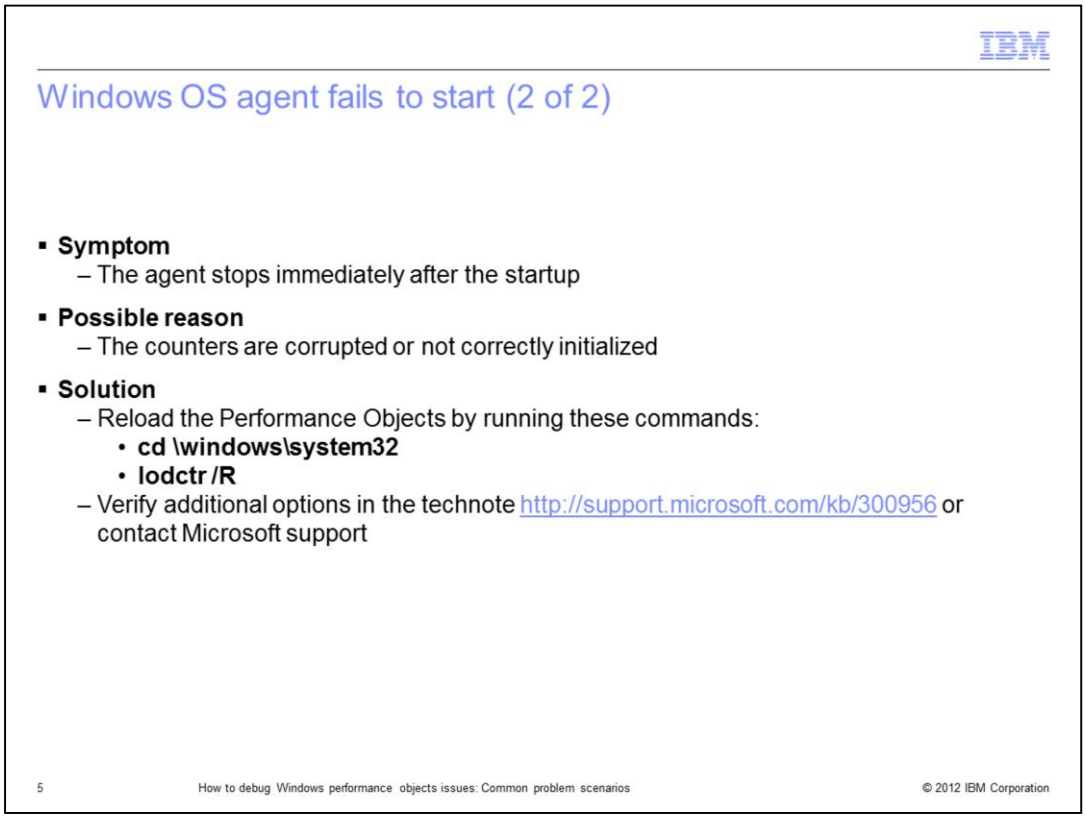
When the agent fails, look at the agent log to immediately see information about the possible root cause.

Using the Manage Tivoli Enterprise Monitoring Server tool, right-click the **Monitoring Agent for Windows OS** entry, select **Advanced**, and then **View Trace Logs**.

You are shown a list of available log files. Choose the most recent one and click OK.

The agent log file is formatted by using human readable time stamps, and you can see what caused the process closure.

If you find messages like *No perfmon counters found, exiting agent*, you can be sure that the problem is with Windows Performance Objects libraries.



Sometimes the Performance Objects are not correctly loaded or initialized when the operating system is started, or they might be corrupted.

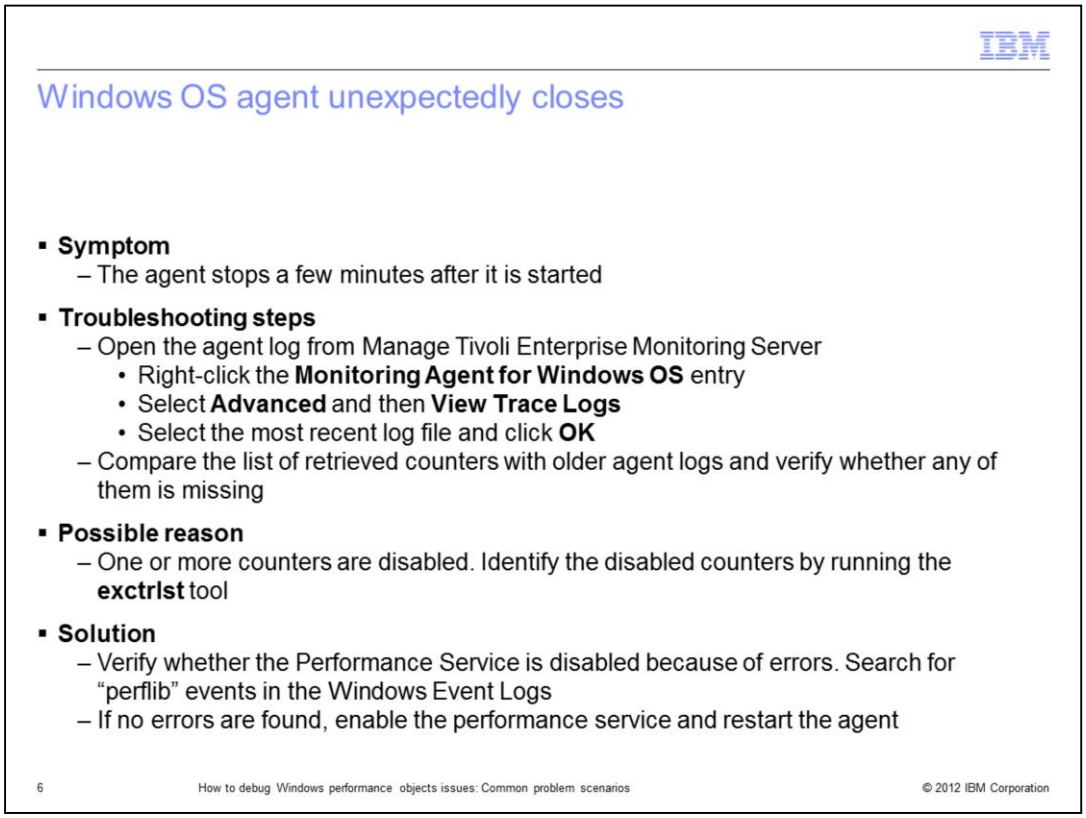
99% of the time the problem is easily resolved by reloading the counters by running these two commands:

cd \windows\system32 …

lodctr /R …

The next slides show the lodctr command is used to try to fix most of the issues with performance libraries. This command forces the operating system to clear the existing cache and reload the performance counters from the dynamic link library (DLL) definitions.

If the lodctr command does not help, you can look at other solutions in the Microsoft technotes. As a last option, you might need to contact Microsoft support to have the performance counters appropriately recovered.



This scenario is similar to the previous one, but in this case the agent seems to be able to complete initialization and continue to run for a few minutes.

The agent log shows no meaningful error about counters. The agent closes unexpectedly after some time.

Some of the basic performance services are expected to be available; otherwise the agent might generate an exception and close.

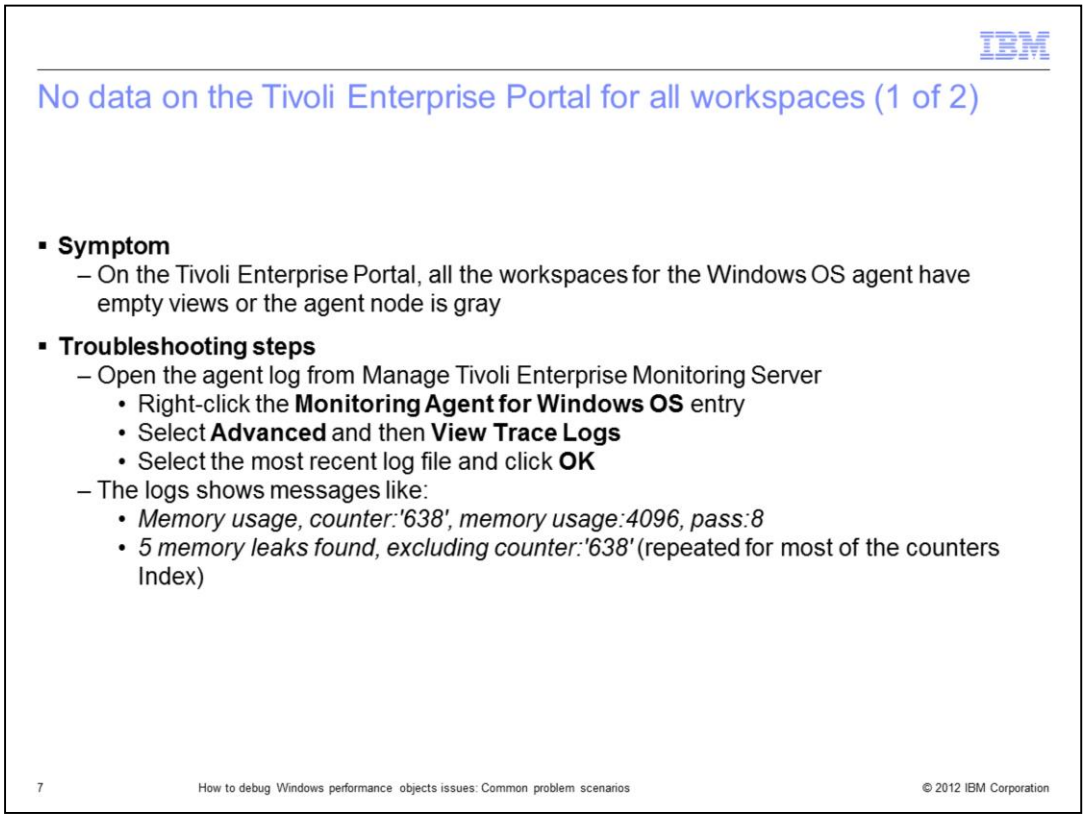
In the example, **PerfNet** and **PerfOS** are always expected to be enabled.

This time, you can use the agent log to verify whether all the expected counters are available, comparing the list with the logs from previous agent executions.

Run the **extctrlst** tool to check whether all the expected services are enabled.

If one or more services are disabled, before reactivating them, check whether they are disabled because of execution generated errors. You can do this by searching for any "perflib" event in the Windows Event Log.

If the event logs are clear of perflib errors, then you can safely enable the pertinent performance service and restart the agent.



In the second example, you can see how Performance Object corruption can lead to other unexpected agent behaviors.

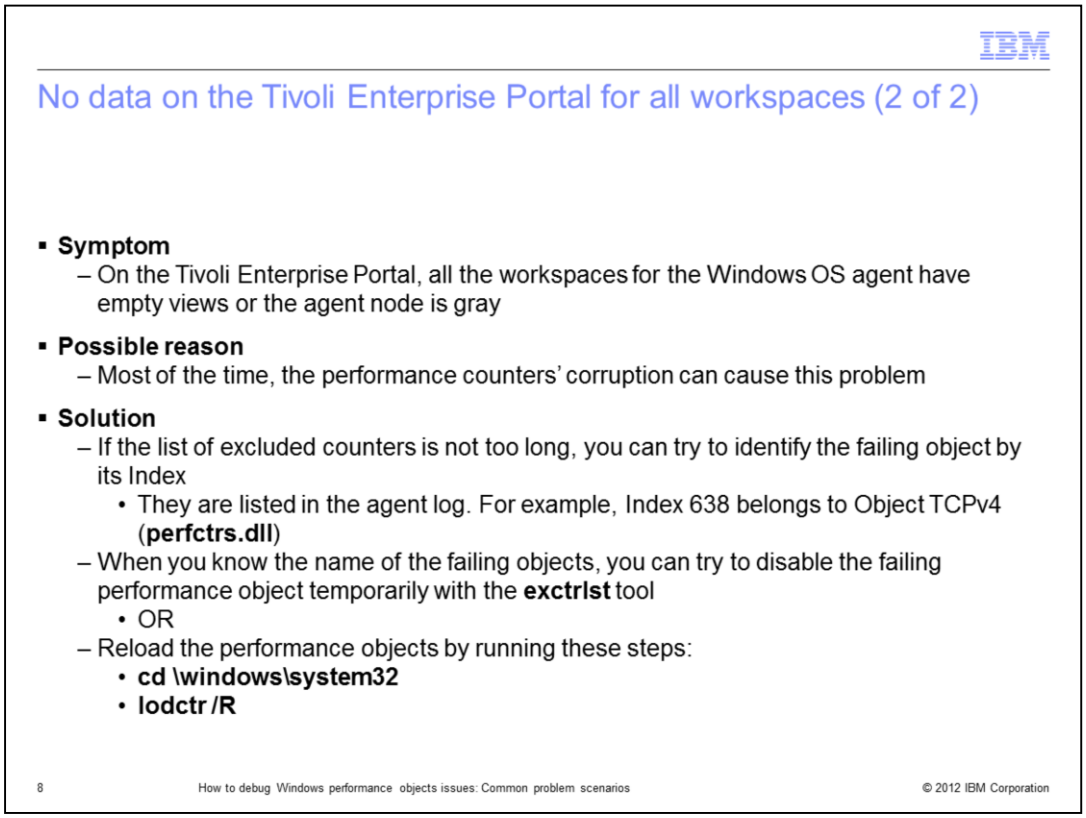
In this case, the agent starts and seems to work, but no data is returned for all the Tivoli Enterprise Portal workspaces, or for most of them.

Like the previous example, you can use the agent log to obtain information to perform quick troubleshooting.

After you open the agent log, you can find messages like the following two messages:

Memory usage, counter:'638', memory usage:4096, pass:8 …

5 memory leaks found, excluding counter:'638' (repeated for most of the counters Index)



Similar to the previous scenario, in this case the problem seems to be one or more corrupted performance counters.

The agent identified possible memory leak conditions, and for this reason, it started to exclude the suspect counters.

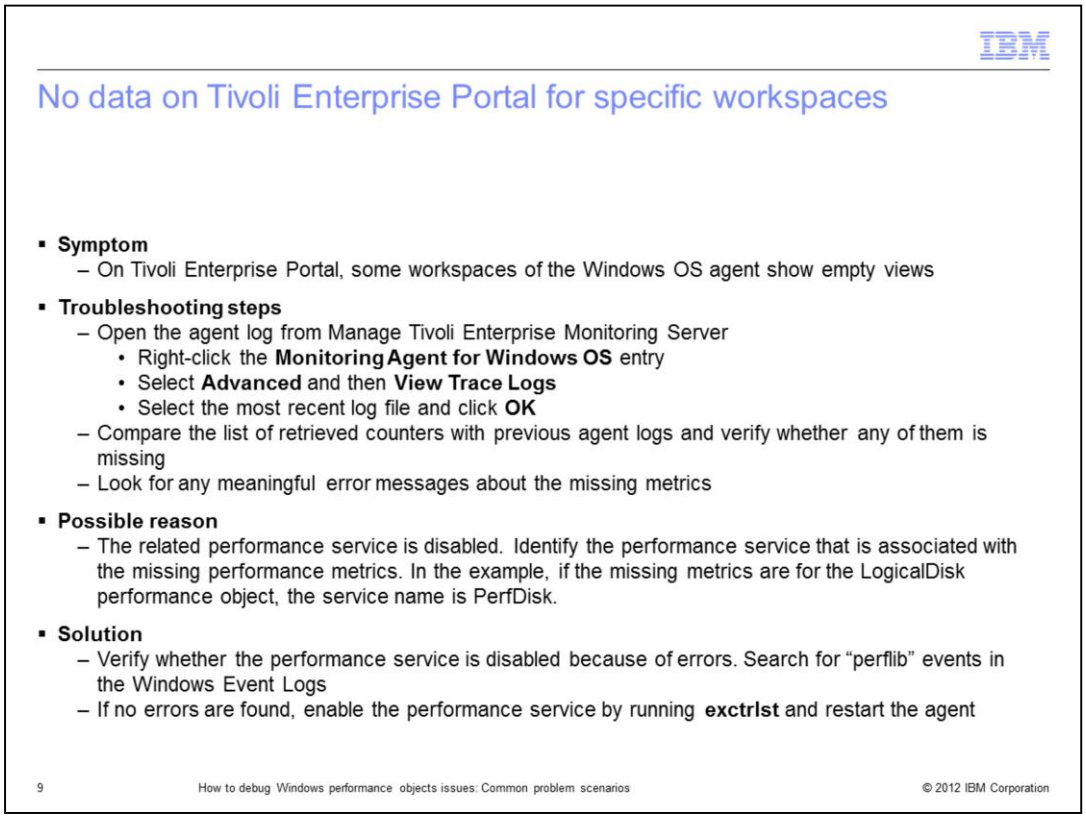
In order to solve the problem, you can try to reload the counters by using the command lodctr /R.

You can also try to identify the failing performance objects by using the index reported in the error message, and disable the related performance object service temporarily.

In the example, Index 638 belongs to Performance Object TCPv4 contained in the DLL called **perfctrs.dll**.

Using the **exctrlst** command, you can try temporarily disabling the failing Performance Objects to see whether the agent is able to work with the data collection for other performance metrics.

The metrics from the disabled objects are not shown in any case.



You saw that if one or more of the basic performance services are disabled, the agent can unexpectedly close. Actually, most of the time you face a less severe symptom than a process closure. You miss the metrics that belong to the disabled performance service.

The agent is up and running, but when you click a specific workspace, it shows empty views. As usual, the first troubleshooting step is to check the agent log for any meaningful error messages.

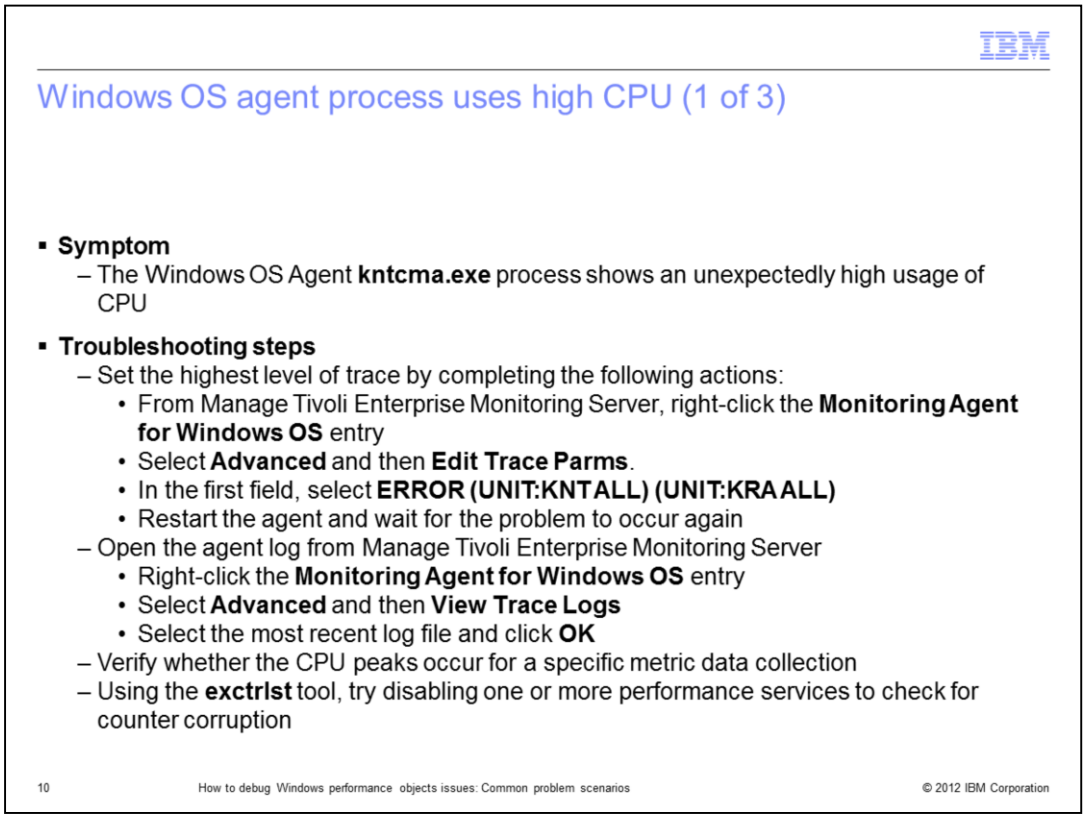
But in this specific scenario, it is likely that the log information is not useful.

More likely, you need to verify whether the performance service associated with the missing metric is enabled or not.

Identify the performance service name that is associated with the performance metric. To do this, you use the table that is shown in the module "How to debug Windows Performance Object issues, overview and tools", slide 9.

In the example, if you are missing data from logical disk view, the related service name is **PerfDisk**.

By using the **exctrlst** command, you can check whether the service name **PerfDisk** is enabled or not. If it is disabled, before enabling it again, verify why it was disabled. Perhaps the operating system automatically disabled it because of previous errors. You can check it within the Windows Event Logs. If no errors are found in the Windows Event Logs, you can safely enable the performance service with the **extctrlst** tool.



Another common problematic scenario concerns performance problems. The root causes can be different, and most of the time are not strictly related to Performance Objects.

This lesson focuses on problems where the Performance Objects have a role. There are also a few cases where the Performance Objects corruption or malfunction cause high CPU or high memory consumption.

The external symptoms typically concern high CPU from the **kntcma.exe** process that is the process of the Windows OS agent.

As previously mentioned, there might be several reasons for this, and the first troubleshooting action you can put in place is activating the agent traces.

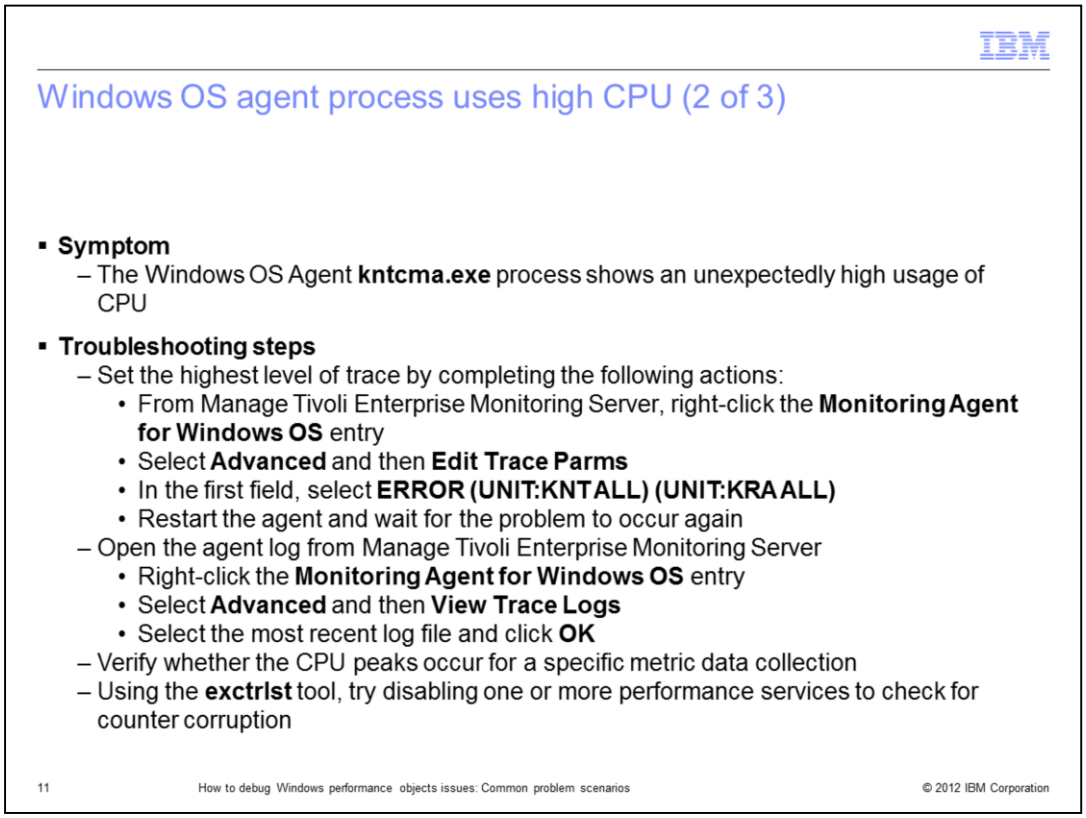
From the Manage Tivoli Enterprise Monitoring Server, right-click the **Monitoring Agent for Windows OS** entry, select **Advanced**, and then the **Edit Trace Parms** option.

In the first field, select the highest trace level available, as shown in this slide.

Restart the agent and wait for the problem to occur again.

When it occurs, open the trace files, again from the Manage Tivoli Enterprise Monitoring Server tool.

(continued on the next slide)



(continued from the previous slide)

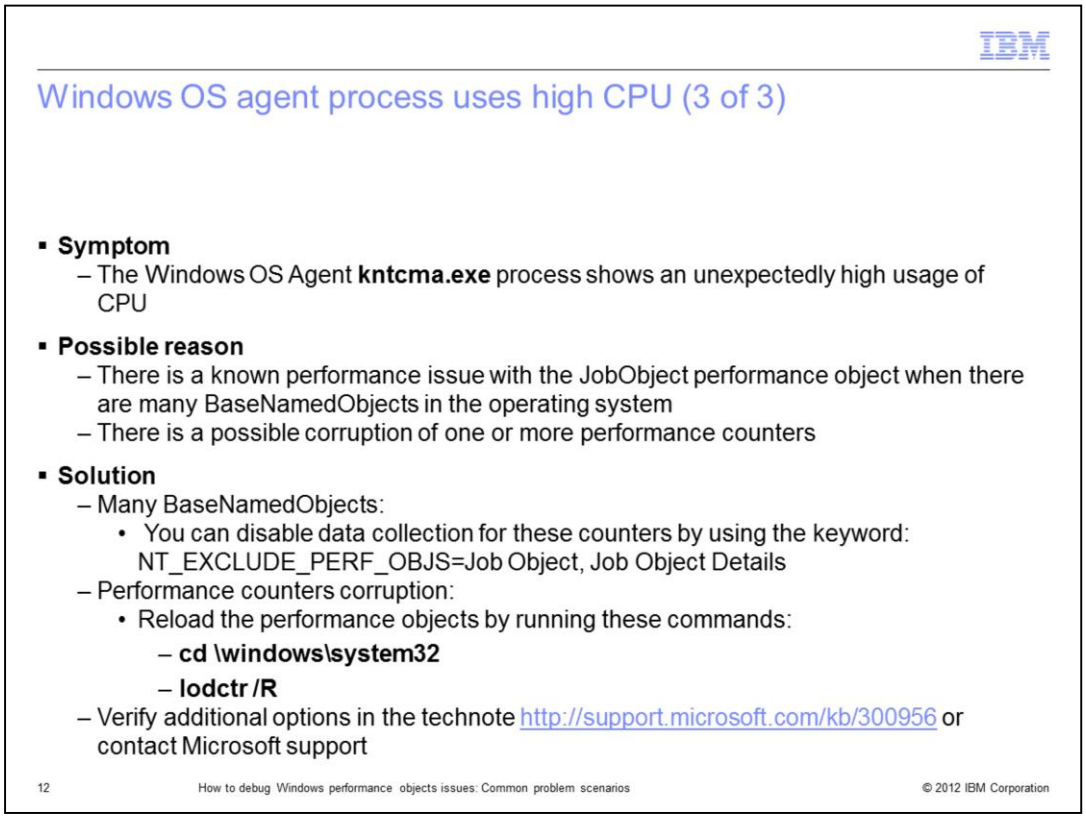
By looking at the log's timestamp, and by using an external profiler such as the Microsoft **perfmon** tool, you can verify whether the CPU peaks occur every time for a specific metric data collection.

Or, you can check the timestamp when the problem started occurring, so that you know what the agent was doing when the problem occurred.

In this way, you can identify if a specific attribute group data collection is causing the problem. In that case, you can investigate the related Performance Object.

Perhaps it is corrupted, or perhaps the amount of data that it retrieved is unexpectedly high.

If you suspect that the problem is with performance counter corruption, you can try disabling one at a time the performance services by using the **exctrlst** tool. With this approach, you can verify whether one or more Performance Objects are causing the issue, without having to spend too much time with agent log analysis.

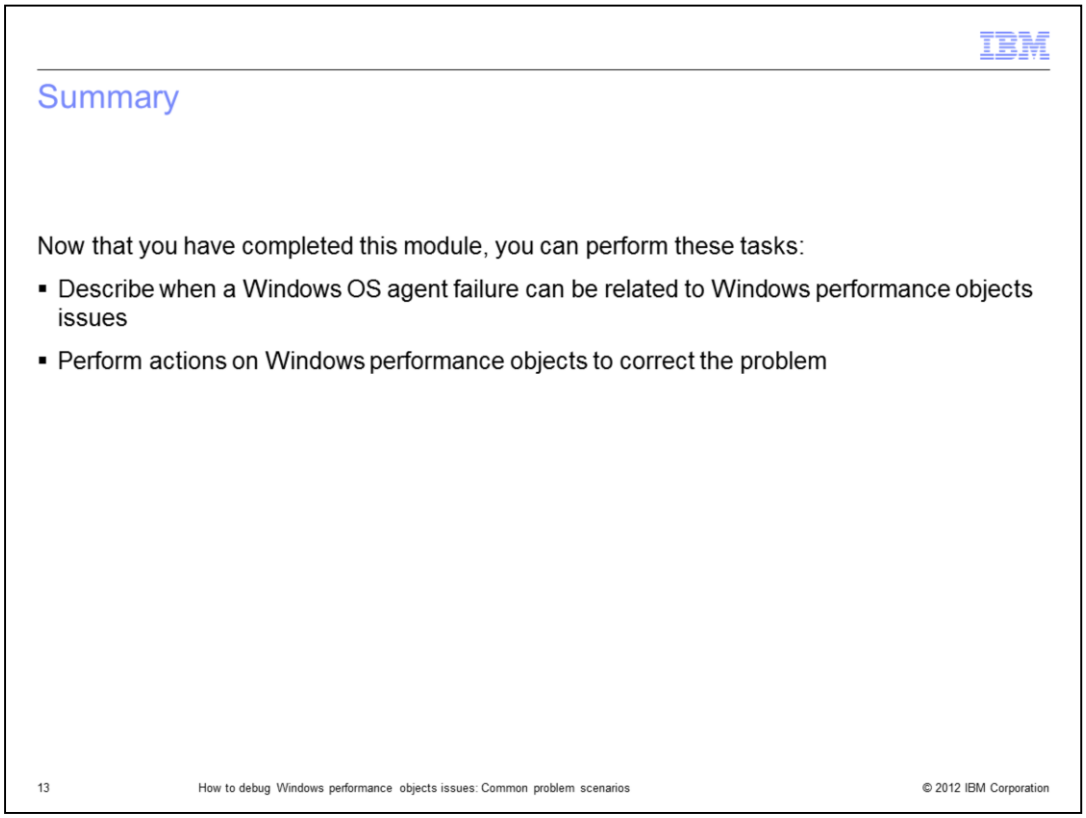


There are known problems with the **JobObject** Performance Object that cause high CPU use when there is a large amount of **BaseNamedObject** in the operating system. In this case, disable the collection for the **JobObject** and **JobObjectDetails** performance counters. This is the only solution unless the Windows administrator is able to reduce the amount of **BaseNamedObject** without impacting the system and services functionalities.

You can disable data collection for some Performance Objects by using the keyword **NT_EXCLUDE_PERF_OBJS**, as shown here.

It is important to note that the code of the Windows OS agent is not causing this issue. Even if the high CPU use is shown for the Windows OS agent process, the CPU is busy with the Windows Performance Object responsible for gathering **JobObject** metric details.

If you are able to identify a corrupted counter as the source for the high CPU use, you can reload the Performance Objects, as described in the other scenarios. If this action does not help, you can follow the steps that are described in the Microsoft technote reported on the slide.



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

- describe when a Windows OS agent failure can be related to Windows performance objects issues

- perform actions on Windows performance objects to correct the problem

Trademarks, disclaimer, and copyright information