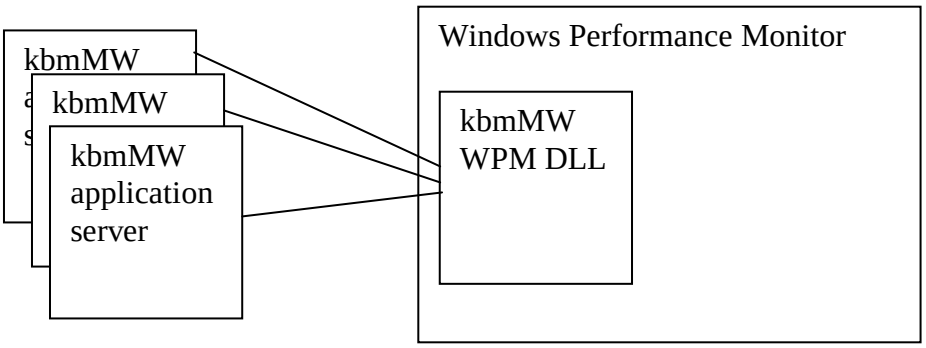
kbmMW and Windows Performance Monitor

kbmMW Enterprise Edition contains support for being monitored using Windows Performance Monitor (WPM).

WPM offers facilities to keep track of the wellbeing of applications who are compatible with it, and the option to automatically trigger an external program, sending an email etc. when some threshold has been reached.

Active monitoring combined with the failover and load-balancing facilities described in another whitepaper will, correctly applied, provide any company with a server side setup with 100% uptime.

A simplified overview of how kbmMW application servers and WPM interacts is given in the following diagram.



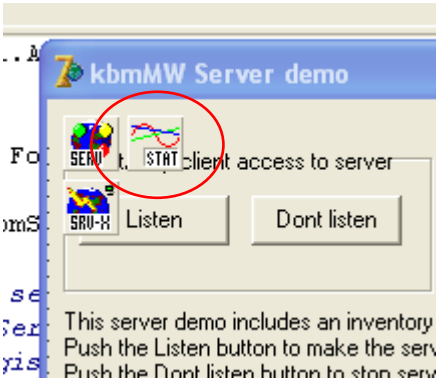
As can be seen, a special kbmMW WPM DLL is used as an intermediate between kbmMW app. servers and WPM.

In the following sections, it will be described how to WPM enable kbmMW application servers, how to make the DLL and install it and how to use WPM to monitor.

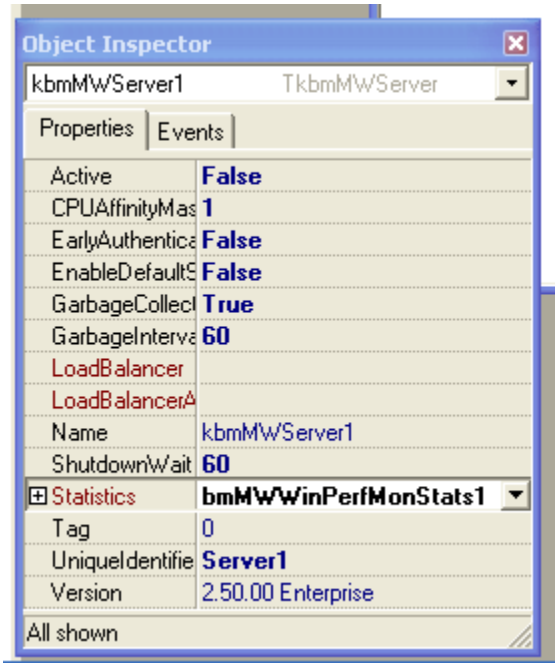
Adding WPM support to a kbmMW application server

In the kbmMW Server component tab, you will find a couple of statistical components, one named TkbmMWLocalStat and another named TkbmMWWinPerfMonStats.

Add the **TkbmMWWinPerfMonStats** to the mainform/datamodule on which **TkbmMWServer** resides.



Set the property **Statistics** on the TkbmMWServer component to point on the performance statistics component.

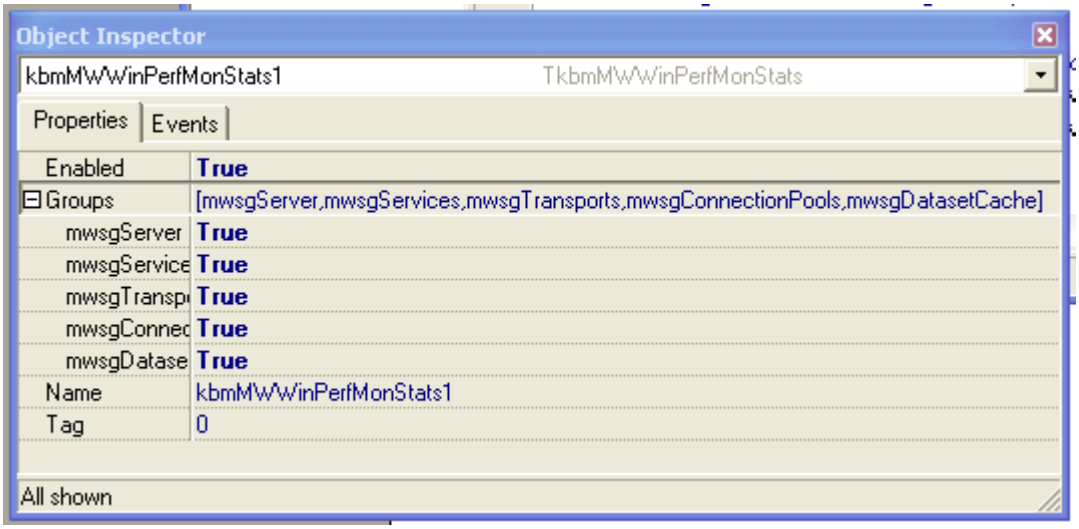


Also remember to set the **UniqueIdentifier** property to something unique.
It's the name/value that this server is identifier. You can set this at runtime too, before setting the TkbmMWServer component active.

Now the application server application supports WPM for server, services and all the transports that are on the same form/module as the TkbmMWServer itself.

If you would also like to prepare WPM for connection pools, remember to set the connection pools Statistics property too.

Next step is to configure the properties of the TkbmMWWinPerfMonStats component.



If **Enabled** is set to false, statistics collection is disabled completely.
Groups should be set to the type of statistics that will be collected. Although statistics collection is relatively fast, it is an overhead. On very busy systems you have the option to only enable collection of selected types of statistics.

That is all you need to do in your application server, so now compile it.

Building the kbmMW WPM DLL and registration helper

You may not need to build the DLL, as kbmMW distributions may include it.
However its source is fully bundled with kbmMW Enterprise Edition, and the following are the steps to build it.

1) Open the kbmMWWinPerfMon.dpr project in Delphi.
2) Compile

Similarly kbmMW Enterprise Edition bundles a WPM DLL registering helper application which we may also need to build (if not bundled with the kbmMW installation).

1) Open the kbmMWWinPerfMonRegistration.dpr project in Delphi.
2) Compile

Isn't that easy? ☺

You now have a dll named kbmMWWinPerfMon.dll and an executable named kbmMWWinPerfMonRegistration.exe somewhere in the kbmMW source directory.

Registering the kbmMW WPM DLL

You can copy the dll to any directory you like, just make sure that the registration helper is copied to the same directory.

To register the dll with WPM, open a DOS prompt and type:

kbmMWWinPerfMonRegistration <server unique identifier> **/register**

To unregister the dll (in case you need to provide a new version of it) type:

kbmMWWinPerfMonRegistration <*server unique identifier*> **/unregister**

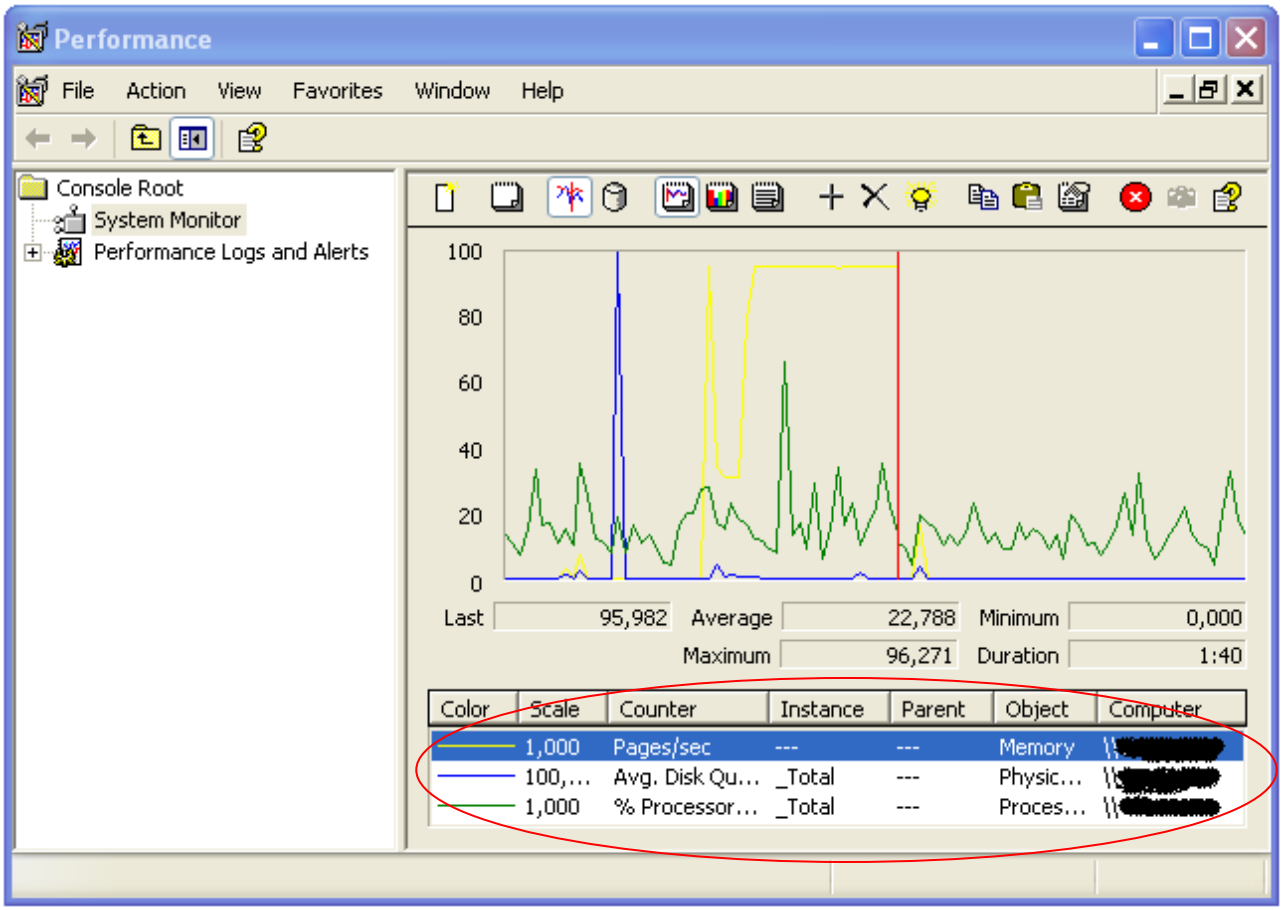
The server unique identifier is the name that has been given in the TkbmMWServer.UniqueIdentifier property at the time the server is activated.

A confirmation of success should be the result of these commands.

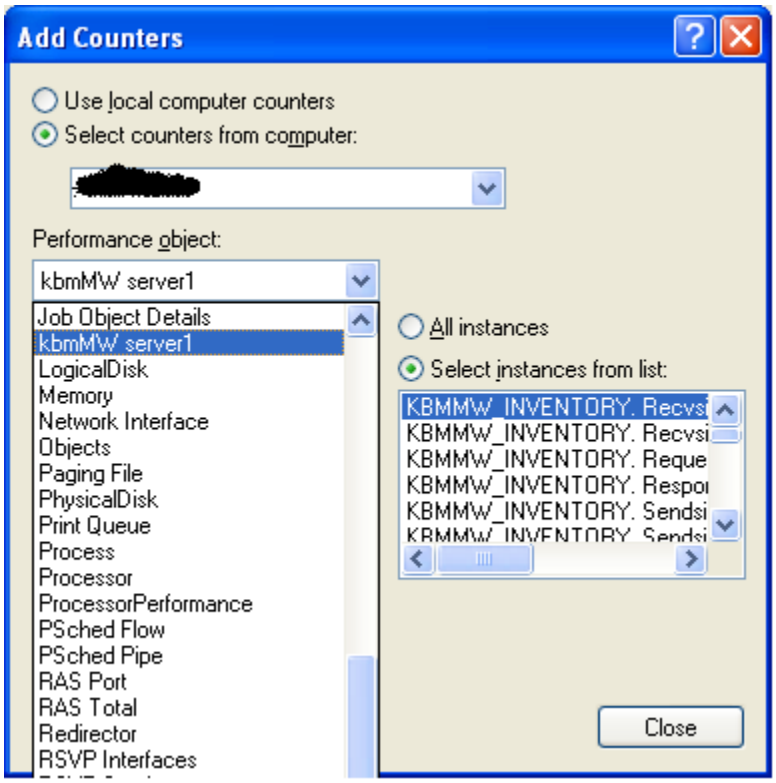
Using WPM

Now start your application server, and let a client access its services.
The performance counters are not available in WPM until they have been triggered first time by a call from a client.

Then open Windows Performance Monitor (Start/Control panel/Administrative tools/Performance).



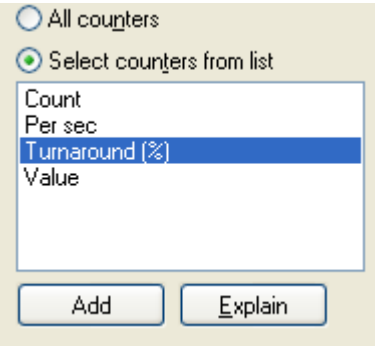
In the bottom pane, right click and select ‘Add counters’ in the popup menu.



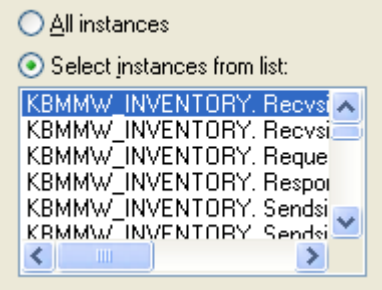
Locate your application server in the Performance object combobox. Notice that its entry is named ‘kbmMW‘ followed by the **UniqueIdentifier** value of your server.
This way you can monitor multiple application servers within the same WPM monitor.

The following is a bit different to standard WPM way of operation, but is due to fact that kbmMW allow for the developer dynamically adding previously undefined statistical measurement points for own custom statistics. WPM actually doesn't support that. However as we are now going to explain, we found a way around that limitation.

In the Counters list you will always only see 4 different counters when you have selected a kbmMW application server: Count, Per sec, Turnaround(%) and Value.



The instances section is where we select the measurement point that you want to watch.



The selection of Counter must match the selection of the Instance section.
That means that if you select KBMMW_INVENTORY.Recvsize, which is a value showing the amount of bytes the application server have received for the inventory service, you must select the Counter to be Value.

If there is a mismatch, the reading will always be 0.

In the next table a list cross reference of which Instances match which Counter types is shown.

When the correct pair (Instance and Counter) has been selected, click Add.
More measurements can be added if needed. Otherwise close the dialog.

The following tables show which measurement points are built into kbmMW and default available.

Application server level measurement points:

Instance	Counter	Description
Server Recvsize	Value	Provides live info about number of bytes received by server.
Server Sendsize	Value	Provides live info about number of bytes sent as a response from server.
Server Response	Turnaround(%)	Provides live info about how big percentage of a sec. it took from a request came in to a response was sent back.
Server Recvsize bytes per sec	Per sec	Provides live info about how many bytes are received per sec. by server.
Server Sendsize bytes per sec	Per sec	Provides live info about how many bytes are sent per sec. by server.

Service level measurement points:

Instance	Counter	Description
xxx Recvsize	Value	Provides live info about number of bytes received by a service.
xxx Sendsize	Value	Provides live info about number of bytes sent as a response from a service.
xxx Response	Turnaround(%)	Provides live info about how big percentage of a sec. it took from a request came in to the service to a response was generated.
xxx Recvsize bytes per sec	Per sec	Provides live info about how many bytes are received per sec. by a service.
xxx Sendsize bytes per sec	Per sec	Provides live info about how many bytes are sent per sec. by a service.

xxx is the combined registered name and version of the service.

Connection pool measurement points:

Instance	Counter	Description
xxx Connections	Count	Current number of connections open towards the backend database.
xxx Locks	Count	Current number of outstanding requests for the backend database.
xxx Locks per sec	Per sec	Number of new outstanding requests for the backend database, per sec.
xxx Tx	Count	Current number of transactions started towards the backend database.
xxx Tx per sec	Per sec	Number of new transactions started towards the backend database, per sec.

xxx is the component name of the connection pool.

Dataset caching measurement points:

Instance	Counter	Description
xxx Cache hits	Count	Total accumulated number of result set and metadata cache hits via this connection pool.
xxx Cache hits per sec	Per sec	Number of cache result set and metadata cache hits per sec.
xxx Cache misses	Count	Total accumulated number of result set and metadata cache misses via this connection pool.
xxx Cache misses per sec	Per sec	Number of cache result set and metadata cache misses per sec.

xxx is the component name of the transport.

Transport level measurement points:

Instance	Counter	Description
xxx Connections	Count	Current number of clients connected to the transport.
xxx Connections per sec	Per sec	Number of new client connections towards this transport per sec.
xxx Requests	Count	Total accumulated number of packets/requests coming in thru this transport since startup.
xxx Requests per sec	Per sec	Number of packets/requests per second coming in thru this transport.

xxx is the component name of the transport.

Custom measurements

kbmMW supports that you can add application specific statistical measurement points which also will show up via WPM.

For this purpose, the kbmMW stats components have 2 methods:

```
procedure AddCounter(const GroupType:TkbmMWStatGroupType;
                     const CounterName:string;
                     const CounterType:TkbmMWStatCounterType;
                     const BufSize:integer);
```

and

```
procedure AddPoint(const GroupType:TkbmMWStatGroupType;
                   const CounterName:string;
                   const User,Info:string;
                   const Value:double;
                   const Base:double);
```

AddCounter is used to prepare a new counter/measurement identified by the CounterName.

GroupType controls what statistical group this counter belongs to. If the statistics component's Groups property do not include this group, AddCounter simply do nothing.
CounterName is the unique identifier for the counter/measurement. If a counter with the same name already exists, AddCounter exits quietly.
CounterType is one of **mwscTurnAround**, **mwscCount**, **mwscValue**, **mwscCountPrSec** or **mwscValuePrSec** and controls how the measurement is used later on.
BufSize is a parameter that controls how many log entries (points with detailed info) that are supposed to be kept for this counter. Windows Performance Monitor do not support this, and thus BufSize is ignored. Usually keep below 100.

After AddCounter has been called, the counter will be available to watch in WPM.

AddPoint is a method that is used for incrementing/decrementing a specific counter by a certain value.

GroupType controls what statistical group this counter belongs to. If the statistics component's Groups property do not include this group, AddPoint simply do nothing.
CounterName is the name of an existing counter created by AddCounter.
User is an optional string identifying the user who are the reason for this measurement point being triggered. Not used by WPM.
Info is an optional string identifying any additional information that is interesting regarding the measurement point. Not used by WPM.
Value is the value you want to add or subtract to/from the counter. Use negative values for subtracting. If the counter is of type mwscTurnAround, then Value should be the difference of two TDateTime values.
Base determines the base for the measurement. It should normally always be one, except if the counter type is mwscValuePrSec, in which case the Base value should contain a TDateTime delta value over how long time the measurement was made.

Default there is a maximum of 200 counters for each application server. If more are needed, open *kbmMWWinPerfMonModule.pas*, locate the constant **KBMMW_PERFMON_MAX_MONITORED_GROUPS_PER_SERVER** and alter its value as needed. Then un-register, recompile and register the WPM DLL and build the kbmMW runtime packages.

Happy monitoring

Best regards
Kim Madsen
Components4Developers