Monitoring .NET Framework with Verax NMS
# Table of contents

Abstract ......................................................................................................................................... 3
1. Adding .NET Framework to device inventory .................................................................... 4
2. Adding sensors for .NET Framework ................................................................................... 7
3. Adding performance counters for .NET Framework .......................................................... 9
4. Creating custom event processing rules for .NET Framework ........................................ 11
5. Verax NMS .NET Framework plugin features overview ................................................... 12
Summary ..................................................................................................................................... 17
Abstract
This publication provides an overview on how to monitor and manage .NET Framework using Verax NMS (including the free Express edition available at http://www.veraxsystems.com/en/downloads - please read terms & conditions for limitations of the Express version).

Tools used:

Agenda:
1. Adding .NET Framework to the list of monitored applications within Verax NMS.
4. Setting up alarms and notification policies.
1. **Adding .NET Framework to device inventory**

In order to include .NET Framework to be monitored by Verax NMS, add an application instance to the device actually running this instance.

**Note:** Verax NMS allows for creating multiple instances for applications of the same type on a single device.

In order to add an .NET Framework to the device running its instance, perform the following steps:

1. Log into the Verax NMS and select **Home** from the main menu.
2. Select a device running the .NET Framework instance from the left-side **aspects** view.

3. In **Summary tab** select **Manage applications** from the **actions** section.

4. A pop-up dialog is displayed.
5. Select **Add application** option from the context menu and click **Go**. A dialog window is displayed. Choose **.NET Framework** from **Application types**.

Verax NMS will ask to enter the following application-specific parameters:

- **Instance name** - Name of the application instance. You can enter any name describing the monitored application instance.
- **Host** - Address of the host running the application instance. In most cases, the host address is an IP address of the device the application instance is assigned to.
- **User** - Username used to connect to the service.
- **Password** – Password used to connect to the service.

**Note:** application-specific parameters depend on the selected application type.
6. Provide the necessary information and click **Save changes**.

7. The system will ask if you want to add a default set of sensors and counters for .NET Framework. Since, (in this example) sensors and performance will be added manually – click **No**.

8. The newly added .NET Framework is now visible in the aspect tree within the host’s node in **Managed Applications** category.

![Figure 5: Aspects tree - Managed Applications](image)

![Figure 6: .NET Framework - properties window](image)
2. Adding sensors for .NET Framework

Sensors are active monitors periodically querying the device services for which they are configured and waiting for their responses. If a query is returned with an expected response, the queried service is considered "available." If a response is not received (timed out), or if the response is not as expected, the queried service is considered "unavailable".

The system includes a number of pre-configured sensors. The following types of sensors for .NET Framework application type are available by default:

- **SNMP** - Checks if SNMPClient can resolve default SNMP Oid (SysObjectID) for supplied device IP address.
- **.NET Framework Sensor** – Checks if given WMI query returns expected result.

In order to add a sensor, perform the following steps:

1. Select device from the aspect tree in **Home view** (.NET Framework in this case).
2. Select **Monitors tab** and switch to sensor list by clicking **Sensor list** link in the upper-right corner of the tab field. The sensor list is displayed.

3. Select **Add** from the global action menu and click **Go**. The wizard dialog is displayed.

Figure 7: Adding a sensor
4. Select the sensor you want to add and click **Next**.

![Add new sensor](image)

**Figure 8: Specify sensor parameters**

5. A dialog shows up with all sensor parameters to be provided. Specify the sensor parameters and click **Finish**.

6. Once the sensors have been added, they are visible on the sensor list (**Monitors tab**).
3. **Adding performance counters for .NET Framework**

Performance counters measure system activity and performance (metrics). The application retrieves their current values in predefined intervals. The aim of probing and collecting data is to analyze and convert the data into a performance graph/chart. The user can define a counter manually or load it from a template.

Performance counter templates are templates with defined probing parameters for specified devices in order to improve and speed up counter creation.

Each Verax NMS counter template is characterized by the following information:

- **Name and description** - unique identifier and optional description,
- **Device type** - type of a device,
- **Protocol type** - protocol used,
- **Probing interval** - pauses between probing.

Counter templates are needed when the counter creation method is set to "from template". Counter templates provide defined probing parameters for specified devices in order to improve and quicken counter creation.

In order to add .NET Framework performance counters, perform the following steps:

1. Select device from aspect tree in **Home view**.
2. Select **Monitors tab** and switch to the counter list by clicking **Counter list** link in the upper-right corner of the tab field. A counter list is displayed.
3. Select **Add** from the global action menu and click **Go**. Select the counter you want to create and click **Next**.

![Figure 9: Adding counter for .NET Framework](image-url)
4. Once the data has been loaded, the edit window shows up with all counter attributes to be provided.

![Add new counter window](image)

Figure 10: Selecting measure metrics

5. Specify the rest of the counter parameters and click **Finish**.

The new counter has been created and is now visible on the counter list.
4. Creating custom event processing rules for .NET Framework

Events are processed by Event Processing Rules running under control of the Event Processing Engine. The Event Processing Engine within the system is able to process events fast without materializing them in database. Verax NMS comes with a set of embedded, flexibly customized processing rules such as: De-duplication, Pairwise matching, Event forwarding, Intermittent failure, Scheduled Maintenance, etc. It also provides users with the ability to implement their own processing rules using JRuby scripting language.

Verax NMS provides complex fault management, such as alarm collection, filtering, blocking, thresholds and correlation (scripted, user-defined rules defining business logic for alarm correlation, cleaning, root-cause, etc.) as well as alarm management actions, e.g. assignment, change of status, clearing, annotation and others. It also enables users to create alarms based on network data etc.

In this example we will show how to assign basic event processing rules:

- Alarm generating
- Event dropping
- Event forwarding
- Severity assigning

To assign an event processing rule, perform the following steps:

1. After selecting the desired host go to **Events tab**.
2. Select the event to assign processing rules to by selecting its corresponding check box in grid view (first column).
3. Select **Assign processing rules** from the global action menu and click **Go**.
4. A dialog window is displayed (see figure below).
5. Select rule category and click **Add new rule**. A dialog window is displayed (see figure below).

![Figure 11: Creating custom processing rule](https://via.placeholder.com/150)

6. The newly created event processing rule is now visible and active (there’s no need to logout).
5. Verax NMS .NET Framework plugin features overview

Verax NMS .NET Framework management plugin supports versions 2.0 and higher of the Microsoft .NET Framework. The communication between the NMS server and managed framework instance takes place using WMI (Windows Management Instrumentation) protocol.

General information view

The view presents general .NET Framework information such as:

- Installed software version
- Number of exceptions thrown by all applications since the .NET Framework instance has been started
- Number of JIT (just-in-time) compiler failures
- Number of running applications
- Number of physical and logical threads
- Number of classes loaded and failed classes
- Number of security checks performed
- Memory usage: total (all applications) heap data usage, amount of committed and reserved data
- Network usage: amount of data sent, amount of data received and number of network connections used by all applications

![General view](image.png)

Figure 12: General view
Memory view

The view displays a detailed memory usage for each running .NET application, including:

- Allocated bytes per second
- Finalization survivors
- Generation 0, 1 and 2 heap sizes and object promotion statistics
- Large Objects heap size
- Number of garbage collector (GC) handles
- Number of GC collections per each heap, pinned objects and induced collections
- % of application time spent on garbage collection
- System virtual memory used by the garbage collector (total, committed, reserved)

The view allows graphical comparison of values of these parameters for any applications selected by a user.

Cache view

The view displays a cache usage for each running .NET application, including:

- Number of cache entries
- Number of cache hits, misses and hit ratio (%)
- Cache trims and turnover rate

The view allows graphical comparison of values of these parameters for any applications selected by a user.
Loading view

The view displays class loader statistics for each running .NET application, including:

- Heap memory committed by the class loader
- Number of application domains (AppDomains)
- Cache trims and turnover rate
- The view allows graphical comparison of values of these parameters for any applications selected by a user.

Security view

The view displays security statistics for each running .NET application, including:

- Number of security access checks at link time and runtime
- % of run time spent on security checks
- Stack depth in the latest security check

The view allows graphical comparison of values of these parameters for any applications selected by a user.
Remoting view

The view displays remote object calls statistics for each running .NET application, including:
- Number of remote channels (connections) established
- Number of proxy objects used for remoting and number of remote contexts
- Total number of remote procedure calls made and number of calls per second
The view allows graphical comparison of values of these parameters for any applications selected by a user.

Exceptions view

The view displays statistics on exceptions raised by each running .NET application, including:
- Total number of exceptions thrown by an application and exceptions thrown per second
- Exception filters and finally clauses executed per second
The view allows graphical comparison of values of these parameters for any applications selected by a user.

Interaction with COM view

The view displays on number of calls to the COM (Component Object Model) objects, including:
- Number of COM-callable-wrappers (CCW)
- Number of times that arguments and return values have been marshaled from .NET (managed) to COM (unmanaged) code
- Number of stubs created between .NET and COM code
The view allows graphical comparison of values of these parameters for any applications selected by a user.

Data provider view

The view displays information on .NET data provider activity, including:
- Number of active, inactive, reclaimed and free connections and connection pools
- Number of non-pooled connections
The view allows graphical comparison of values of these parameters for any applications selected by a user.

Networking view

The view displays information on network traffic statistics for each .NET application, including:
- Number of network connections established
- Numbers of sent and received bytes and packets (datagrams)
- Detailed information on HTTP requests: total number of requests made, requests per second, aborted per second, queue times and others
The view allows graphical comparison of values of these parameters for any applications selected by a user.
Just-in-time (JIT) view

The view displays information on JIT compiler statistics for each running .NET application, including number of methods compiled, compilation speed and others. The view allows graphical comparison of values of these parameters for any applications selected by a user.

Figure 15: Just-in-time (JIT) view

Locks and threads view

The view displays information about application threads and locks (numbers, waiting time and others).
Summary

If you performed all actions described in chapters 1-5 you are now able to monitor .NET Framework application.