Post-installation tasks for Microsoft .NET tier collectors

This section includes the following topics:

- Performing post-installation action items
- Configuring DLLs, classes, and methods to instrument
- Defining permissions settings for a Precise installation folder
- Changing the activity tracking threshold
- About Precise for Microsoft .NET agent ports

When the Microsoft .NET Tier Collectors are installed, you need to perform post-installation action items. These action items vary depending on the .NET application type, and include editing configuration files and restarting Precise agents. Not all action items appear on screen as described in Performing post-installation action items. This section includes instructions regarding action items that appear as part of the installation, as well as an overview of instrumentation action items that are not included.

Performing post-installation action items

To perform Microsoft .NET Tier post-installation tasks

1. In the Microsoft .NET Tier - Installation dialog box, click **Next**.
2. In the Microsoft .NET Tier - Post-Installation dialog box, follow the instructions to perform all post-installation tasks.
3. Click **Finish**.

Configuring DLLs, classes, and methods to instrument

You can add the DLLs you wish to monitor using the DLLs dialog box in the Instrumentation Settings menu, as explained in the Precise for Microsoft .NET User's Guide. But afterwards you must edit the file instrumentation.xml manually to define which classes and methods to instrument.

The Precise for Microsoft .NET activity tracking is based on instrumentation of your application DLLs. This information is configured in the instrumentation.xml file. The way you configure this file (this means which DLLs to instrument and which classes or methods) affect how much information will be tracked and also the overhead of the Precise for Microsoft .NET activity tracking on your application.

The instrumentation.xml file contains many configurable settings, which are explained in the Precise Administration Guide in detail.

The following file is an example section of the instrumentation.xml file which configures the instrumentation of the Microsoft Petshop sample Web application:

```
<instance name="AspNetIIS6">
    <dlls>
        <dll name="pet*.dll" /><!- instrument all DLLs that start with pet* -!>
    </dlls>
    <instrument>
        <classes>
            <class>
                <name>Petshop.*</name><!- instrument all classes of which the namespace begins with Petshop -!>
                <called-method>
                    <methods>
                        <method>
                            <name>*</name><!- instrument all methods in each class -!>
                        </method>
                    </methods>
                </called-method>
            </class>
        </classes>
    </instrument>
....
```

Defining permissions settings for a Precise installation folder

The Precise for Microsoft .NET hooks that are injected into the monitored .NET process, are run under the same user account on which your .NET application is running. This user must have full permission rights to the *<precise_root>* folder and all its sub-folders.

You can change the security settings of the *<precise_root>* folder with the Windows File Explorer Security tab.

> If you have several application pools which translate into different Windows users, all these users must be granted full permission to the *<precise_root>* folder.

Changing the activity tracking threshold

The Precise for Microsoft .NET installation pre-configures the activity data tracking threshold and sets it to 50 milliseconds. This means that only methods, service requests, SQL statements, and Web services that have a response time that is equal to, or greater than 50 milliseconds will be collected.

You can change this default value in the ActivityCollector.xml file, located in the <precise_root>/products/dotnet/config folder any time after the installation:

```
<activity-collector-config>
    <!-- psdn_act_col.exe configuration, configurable per server (default) & per instance -->
    <aggregator>
        <topnsql>5</topnsql><!-- Top n SQL statement to monitor -->
        <sla>5000</sla><!-- Red SLA breach value for top HTTP invocations in msec. -->
        <nearsla>1000</nearsla><!-- Yellow SLA breach value for top http invocations in msec. -->
        <insane-rt>300</insane-rt><!-- Long running thread timeout in seconds -->
        <bit-vector>false</bit-vector>
        <last-level-bit-vector>false</last-level-bit-vector>
    </aggregator>
    <!-- Tracker.dll settings, configurable per server (default) & per instance -->
    <tracker>
        <threshold>50</threshold><!-- Threshold in msec for filtering out events before forwarding it to the Collector agent. -->
    </tracker>
</activity-collector-config>
```

The new setting will only take effect after your .NET application is restarted. For ASP.NET that means an IIS reset or recycling the application pool.

This setting has a significant effect on the overall overhead of Precise for Microsoft .NET agent on your system. Lowering this value too much can cause your Web application and server to slow down considerably. If Precise for Microsoft .NET is installed on a development server, you can set this value to 0 to collect all data. On heavy loaded production systems, it is advisable not to change the default.

More information about controlling the Precise for Microsoft .NET overhead on your .NET application in the Precise Administration Guide.

About Precise for Microsoft .NET agent ports

The Precise for Microsoft .NET data collection is based on establishing a TCP connection between a tracking module (tracker.dll), the activity Collector and instrumentation agent. The default port numbers for the installation are respectively 20755 and 20756. If any of these port numbers are in use by another application on the server, you can change its value respectively in the collector.xml and instrumenter.xml files.

Port conflicts can cause the Precise for Microsoft .NET Collector not to work and to submit errors. The following is an error example that can appear in the dotnet.collector.activity.trc file:

```
========================================
ID : CONFIG-0000000
Time : 2020-11-23 10:03:54.750
GMT+02 i3 Time : 2020-11-23 08:03:54.750
GMT Process : psdn_dncol_act.exe -l (ID:2704)
Logger : com.Symantec.apm.i4dotnet.collector.activity
Msg : Exception in remoting : System.Net.Sockets.SocketException: Only one usage of each socket address (protocol
/network address/port) is normally permitted
========================================
```

The following is an error example that can appear in the dotnet.collector.dncol_isntr.trc file:

```
========================================
ID : CONFIG-0000000
Time : 2020-11-23 10:09:05.842
GMT+02 i3 Time : 2020-11-23 08:09:05.842
GMT Process : psdn_dncol_instr.exe (ID:4972)
Logger : com.Symantec.apm.i4dotnet.collector.dncol_instr
Msg : Unable to setup TCP server on port 20756, check that it is not used by another process, ex=Only one usage
of each socket address (protocol/network address/port) is normally permitted
========================================
```