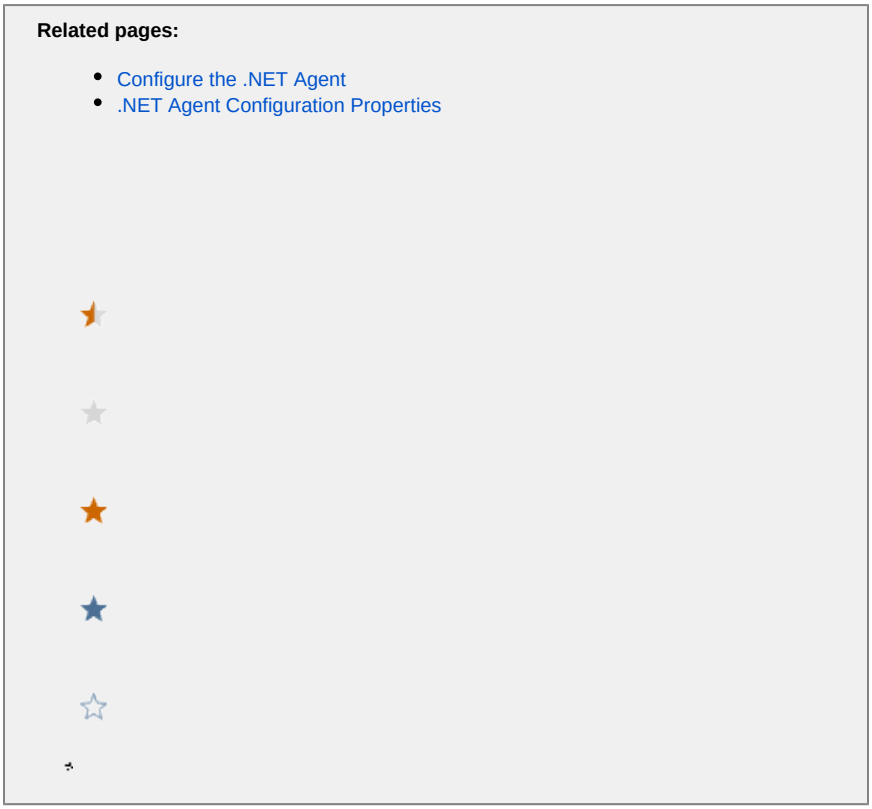
Configure the .NET Agent for Windows Services and Standalone Applications

Related pages:

- Configure the .NET Agent
- .NET Agent Configuration Properties



The .NET Agent automatically instruments IIS applications only. However, you can follow the instructions in this topic to manually configure the .NET Agent to instrument Windows services or standalone .NET applications.

Overview

If your web application is not configured to run as an in-process handler, the application may run as `dotnet.exe` or `mywebappapplication.exe`. In this scenario, the application's site name and application path are not discovered automatically. You need to configure these applications manually and provide tier and node names that best describe the application.

Before Starting

To instrument Windows services and standalone .NET applications, you edit the .NET Agent configuration file, `config.xml`.

> Instrument the default domain if that is where your application runs, as AppDynamics does not automatically instrument the default domain out of the box. To determine whether you should use the default domain, and how to configure it, see Instrument the DefaultDomain for Standalone Applications.

Before starting, verify that the services or applications you want to instrument are .NET applications rather than native applications or another type of application. From a command line, run the following command:

```
tasklist /m "mscor*"
```


This command only works for 64-bit processes. For 32-bit processes, use the process explorer to view the dependencies of the process and whether there are any .NET .dll files loaded into the file.

The output lists the processes that have DLLs starting with mscor*, indicative of .NET processes. Processes that are not on the list are not .NET processes and cannot be instrumented with the .NET Agent.

If you have previously instrumented IIS applications on the server that hosts the Windows services and standalone applications, the server should already have a `config.xml` file that you can edit. If not, perform the following steps to generate one:

1. Install the .NET Agent for Windows.
2. Run the AppDynamics Agent Configuration utility.

If you want to avoid instrumenting IIS applications, choose the manual tier naming approach and omit the step of assigning tiers for the IIS application. This disables instrumentation for the IIS applications, allowing you to instrument only the intended Windows services or standalone applications.

The utility performs these configuration tasks:

1. Changes the location of the logs directory and assign permissions.
2. Configures and tests connectivity to the Controller.
3. Sets the business application for the agent.

Manually Configure the .NET Agent

Once you have configured the Controller properties for the .NET Agent, instrument your Windows service or standalone application by adding the Standalone Applications element to the `config.xml`.

1. Edit the `config.xml` file as an administrator. See Administer the .NET Agent.

   If you have not instrumented any IIS applications, the file contains the minimal configuration for the Controller connectivity and the machine agent. Verify the Controller properties and the Business Application name, as in the following example:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <appdynamics-agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.
   w3.org/2001/XMLSchema">
     <controller host="mycontroller.example.com" port="8090" ssl="false">
       <application name="My Business Application" />
     </controller>
   ...
   </appdynamics-agent>
   ```

   If you have already instrumented IIS applications, those configurations appear under the IIS element.
2. Add a single standalone applications element, `standalone-applications` under the app-agents element, and under the `standalone-applications` element, add a standalone application element, `standalone-application` for each Windows service or standalone application to instrument. For example:

```
...
<app-agents>
   ...
   <standalone-applications>
      <standalone-application executable="MyStandaloneApp.exe">
        <tier name="Standalone Tier 1" />
      </standalone-application>
      <!-- Instrument a standalone application using a partial path. -->
      <standalone-application executable="MyApplication\MyOtherStandaloneApp.exe">
        <tier name="Standalone Tier 2" />
      </standalone-application>
      <!-- Instrument a Windows service using arguments. -->
      <!-- The following example matches the command "MyWindowsService.exe -d -x -r". -->
      <standalone-application executable="MyWindowsService.exe" command-line="-x">
         <tier name="Windows Service Tier" />
      </standalone-application>
   </standalone-applications>
</app-agents>
...
```

In the standalone application element configuration:

- Use the `tier` element to assign the instrumented application to a tier in the Controller. See .NET Agent Configuration Properties.
- Identify the executable file of the application in the Standalone Application element `executable` attribute using one of the following formats:
    - Executable name: For example, `MyStandaloneApp.exe` or `MyWindowsService.exe`. The file extension is optional, so `MyStandaloneApp` also works. If `dotnet.exe` is used for starting the .NET Core application, then specify `"dotnet.exe"` as application executable (see Sample Configuration File for details).
    - Full or partial path to the executable: For example, `C:\Program Files\MyApplication\MyStandaloneApp.exe` or `MyApplication\MyStandaloneApp.exe`. Use the full or partial path when you want to assign different AppDynamics tiers to separate instances of the same executable file running from different paths.
    - To differentiate between two instances of the same executable, specify any unique portion of the command line invocation format of the application, such as an argument, in the Standalone Application command-line attribute.

You can discover the path to a Windows service executable in the Services panel of the administrative tools. In Services, click on the service and choose Properties. The path appears in the General tab.

3. Restart the `AppDynamics.Agent.Coordinator` service.
4. Restart the Windows service or standalone application.
5. If your Windows service or standalone application does not implement an auto-detected framework, you must configure a POCO entry point for a class/method in your service for the agent to begin instrumentation.

Sample Configuration File

This sample `config.xml` demonstrates instrumentation for a Windows service and standalone application:

```
<?xml version="1.0" encoding="utf-8"?>
<appdynamics-agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org
/2001/XMLSchema">
  <controller host="mycontroller.mycompany.com" port="8090" ssl="false">
    <application name="My Business Application" />
  </controller>
  <machine-agent />
  <app-agents>
    <IIS>
        <applications />
    </IIS>
    <standalone-applications>
        <standalone-application executable="MyWindowsService.exe" command-line="-x">
           <tier name="Windows Service Tier" />
        </standalone-application>
        <standalone-application executable="MyStandaloneApp.exe">
           <tier name="Standalone Tier" />
        </standalone-application>
    </standalone-applications>
  </app-agents>
</appdynamics-agent>
```

If the .Net Core application is published as executable, then it can be configured similar to .NET Framework applications by specifying the executable name and optionally the command line. However, if the application is published as a library, then it is started by invoking the `dotnet.exe` tool.

The following sample configuration demonstrates how to instrument a .NET Core standalone application hosted by the `dotnet.exe` :

```
<?xml version="1.0" encoding="utf-8"?>
<appdynamics-agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org
/2001/XMLSchema">
  <controller host="mycontroller.mycompany.com" port="8090" ssl="false">
    <application name="My Business Application" />
  </controller>
  <machine-agent />
  <app-agents>
    <IIS>
        <applications />
    </IIS>
    <standalone-applications>
        <standalone-application executable="dotnet.exe" command-line="MyApp.dll"> //command-line
option instructs the agent to monitor only "MyApp"
           <tier name="DotNet Core Tier" />
        </standalone-application>
      </standalone-applications>
  </app-agents>
</appdynamics-agent>
```

Troubleshooting

Intermittent Loss of Windows Services Instrumentation

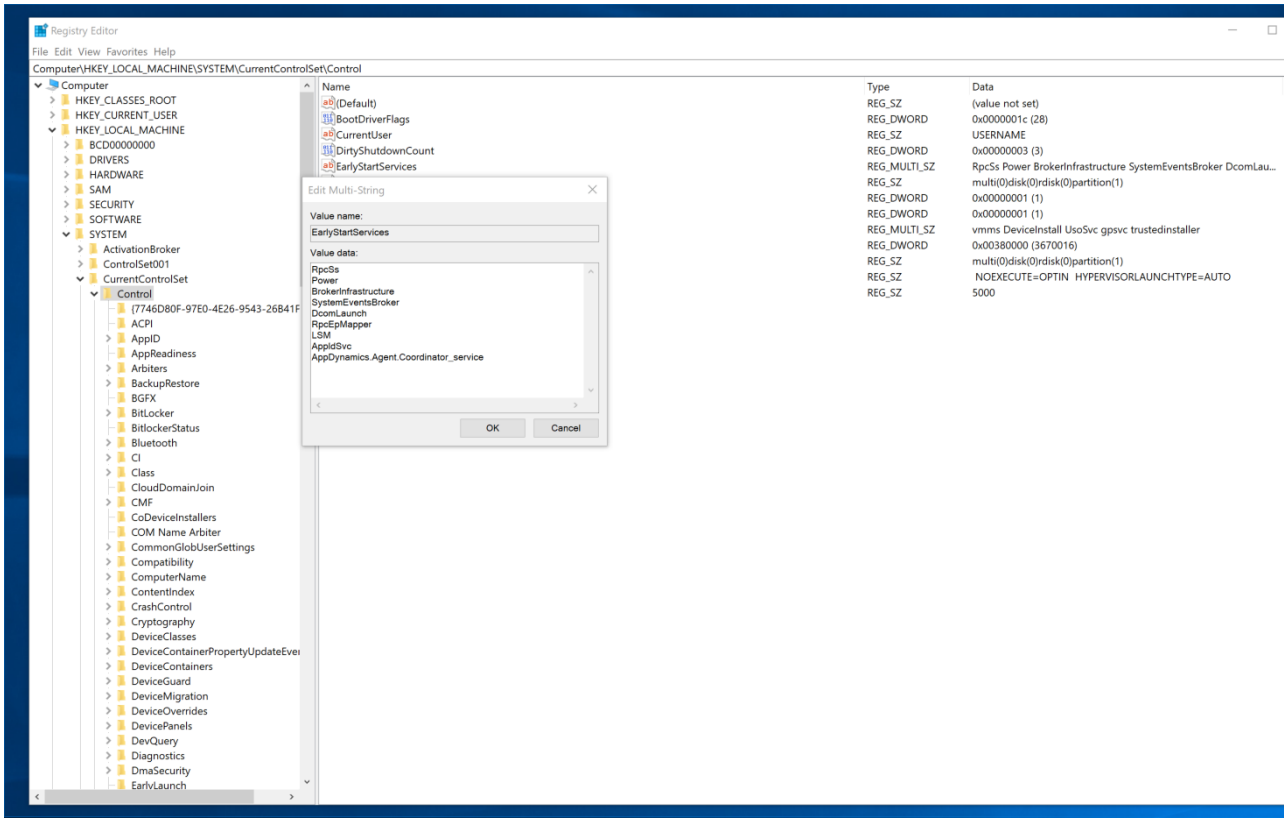
When instrumenting Windows Services, there may be cases where the instrumented service initializes before the `AppDynamics.Agent.Coordinator` service. When this happens the Profiler shuts down and no instrumentation happens. This problem may present intermittently and is not common.

The following describes the procedures required to overcome this problem for Windows 2012 and later, and for Windows 2008.

Windows 2012 and later

To ensure the AppDynamics.Agent.Coordinator service initializes before the instrumented Windows Service, modify the registry as follows:

Edit the Reg key `EarlyStartServices @HKLM:\SYSTEM\CurrentControlSet\Control` and add `AppDynamics.Agent.Coordinator_service` to the list of early start services.



Windows 2008

If you are using Windows 2008, please contact AppDynamics Support for assistance.