AppDynamics for .Net

AppDynamics Pro Documentation

Version 3.8.5

AppDynamics for .NET

This information covers using AppDynamics for .NET applications and environments. For general information see AppDynamics Essentials and AppDynamics Features.

Tutorials

Supported Environments and Versions for .NET

Supported Platform Matrix for the App Agent for .NET

Supported Runtime Environments

This section lists the environments where the .NET agent does some automatic discovery after little or no configuration.

OS Versions

- Microsoft* Windows* Server 2003 (32-bit and 64-bit)
- Microsoft Windows Server 2008 (32-bit and 64-bit)
- Microsoft Windows Server 2008 R2
- Microsoft Windows Server 2012

Microsoft .NET Frameworks

- Microsoft .NET Framework versions 2.0, 3.0, 3.5, 4.0, 4.5

Runtime Environments

- Microsoft IIS versions 6.0, 7.0, 7.5, 8.0
- Microsoft SharePoint 2010, 2013 as services running inside IIS
- Managed Windows Services
- Managed Standalone Applications

Microsoft Windows Azure

- Windows Azure Cloud Services (Web Roles and Worker Roles)

Unsupported Frameworks

- Microsoft .NET versions 1.0, 1.1
- Unmanaged native code
- Windows Azure Web Sites

Automatically Discovered Business Transactions

The App Agent for .NET discovers business transactions for the following frameworks by default. The agent enables detection without additional configuration.

Type	Custom Configuration Options	Downstream Correlation
ASP.NET	Yes	Yes
ASP.NET MVC 2 ASP.NET MVC 3 ASP.NET MVC 4 ASP.NET MVC 5	Yes	Yes
.NET Remoting	No	Requires configuration. See Enable Correlation for .NET Remoting.
Windows Communication Foundation (WCF)	No	Yes
Web Services including SOAP	No	Yes
Message Queues		
Apache ActiveMQ NMS framework and related MQs	No	Yes
IBM WebSphere MQ	No	Yes
RabbitMQ	Yes	Yes
TIBCO Enterprise Message Service	No	Yes
TIBCO Rendezvous	No	Yes

New in 3.8.4, the App Agent for .NET automatically discovers entry points for ASP.NET web forms with the Async property set to "true" in the Page directive.

Supported Loggers for the App Agent for .NET

- Log4Net
- NLog
- System Trace
- Windows Event Log

If you are using a different logger, see Custom Logger Definitions.

Remote Service Detection

The App Agent for .NET automatically detects the following remote service types. The agent enables detection by default. You don't need to perform extra configuration.

Type	Custom Configuration Options	Async Detection †	Downstream Correlation
Directory Services, including LDAP	No	No	N/A
HTTP	Yes	Requires configuration. See Monitor Async Transactions for .NET .	Yes
.NET Remoting	Yes	No	Requires configuration. See Enable Correlation for .NET Remoting.
WCF	Yes	Requires configuration. See Monitor Async Transactions for .NET .	Yes
WCF Data Services	Yes	No	No
Web Services, inlcuding SOAP	Yes	Requires configuration. See Monitor Async Transactions for .NET .	Yes
Message Queues			
Apache ActiveMQ NMS framework and related MQs	Yes	No	Yes
IBM WebSphere MQ (IBM XMS)	Yes	No	Yes
Microsoft Message Queuing (MSMQ)	Yes	No	N/A
Microsoft Service Bus / Windows Azure Service Bus	No	No	Yes

TIBCO Enterprise Message Service	Yes	No	Yes
TIBCO Rendezvous	Yes	No	Yes
Windows Azure Queue	No	No	No

† The agent discovers asynchronous transactions for the Microsoft .NET 4.5 framework. See Monitor Async Transactions for .NET for details.

Supported Windows Azure Remote Services

Type	Configuration can be customized	Downstream Correlation
Azure Blob	No	No
Azure Queue	No	No
Microsoft Service Bus	No	Yes

Data Storage Detection

In a .NET environment, AppDynamics Agent for .NET automatically detects the following data storage types. The agent enables detection by default. You don't need to perform extra configuration.

Type	Configuration Can Be Customized	AppD for Databases?
ADO.NET (see supported clients below)	Yes	No
Windows Azure Blob Storage	No	No

Supported ADO.NET Clients

We can monitor any ADO.NET client version and type. Clients we've tested include the following:

Database Name	Database Version	Client Type
Oracle	10, 11, 12	ODP.NET
Oracle	10, 11, 12	Microsoft Provider for Oracle
MySQL	5.x	Connector/Net and ADO.NET
Microsoft SQL Server*	2005, 2008, 2012	ADO.NET

* *Microsoft, SQL Server,* and *Windows* are registered trademarks of Microsoft Corporation in the United States and other countries.

Install the App Agent for .NET

This topic describes how to install the App Agent for .NET. To learn about upgrading the App Agent for .NET, see Upgrade the App Agent for .NET.

Install the App Agent for .NET once on each system that hosts managed .NET applications: IIS applications, Windows services, or standalone applications. At start up, the agent initializes an individual instance of itself for each application running in the CLR.

Installing the App Agent for .NET

Installation Prerequisites

Microsoft Distributed Transaction Coordinator (MSDTC)

COM+
See Verify COM+ Services are enabled.

Installation Procedure

To install the App Agent for .NET

1. Download the App Agent for .NET MSI installer package from the AppDynamics Download Zone.

2. Run the MSI installer package.

3. Read the **End User Agreement** and click to accept. Click **Next**.

4. Optionally change the destination directory for the App Agent for .NET and click **Install**.

By default, the agent installs to the following directory:

```
C:\Program Files\AppDynamics\AppDynamics .NET Agent
```

5. Click **Yes** on the **User Account Control** window to allow the installer to make changes to the

computer.

If the current account does not have administrator privileges, the installer prompts you to supply the password for an administrator account.

6. Wait for the installation to complete.

See Agent Configuration to decide whether or not to launch the AppDynamics Agent Configuration utility.

If you encounter problems installing, see Resolve App Agent for .NET Installation and Configuration Issues.

Agent Configuration

Use the following guide to decide whether or not to launch the AppDyanmics Agent Configuration utility:

Launch the agent configuration utility

- If this is a **new installation** and you are instrumenting **IIS applications**, see Configure the App Agent for .NET.
- If this is a **new installation** and you are instrumenting **Windows services**, see Enable the App Agent for .NET for Windows Services.
- If this is a **new installation** and you are instrumenting **standalone applications**, see Enabl e the App Agent for .NET for Standalone Applications.
- If this is an **upgrade from 3.7.7 or earlier** and you instrument **IIS or Windows services** wit h **automatic tier assignment**, see To let AppDynamics automatically name the tiers.
- If this is an **upgrade from 3.7.7 or earlier** and you instrument **IIS or Windows services** wit h **manual tier assignment** and you wish to clean up old configurations, see To clean up legacy configurations.

Don't launch the agent configuration utility

- If this is an **upgrade from 3.7.8** or later.
- If this is an **upgrade from 3.7.7** or earlier and you instrument **IIS** or **Windows services** with **manual tier assignment** and you don't wish to clean up old configurations at this time.

Silent Install Options

See Unattended Installation for .NET.

Learn More

- Configure the App Agent for .NET
- Unattended Installation for .NET
- Logical Model
- App Agent for .NET Configuration Properties

Configure the App Agent for .NET

The App Agent for .NET requires information about your .NET applications: IIS applications, Windows services, or standalone applications. Configure the App Agent for .NET according to what kind of application you want to monitor:

- For IIS applications, use the configuration utility with either automatic or manual tier naming. See the instructions in this topic.
- For Windows services, use the configuration utility, then manually update the config.xml. See Enable the App Agent for .NET for Windows Services.
- For standalone applications, use the configuration utility, then manually update the config.xml See Enable the App Agent for .NET for Standalone Applications.

Use the App Agent for .NET Configuration Utility to configure the agent just after installation, or to make changes to existing agent configurations.

Configuration Considerations

AppDynamics recommends that you install the Controller, or have access credentials to a SaaS Controller, before installing an agent.

Prior to configuration, run the .NET Agent Installer. See Install the App Agent for .NET.

The utility configures one agent at a time.

AppDynamics implements the profile API of the .NET CLR. Since a Windows machine only allows use of one profiler at a time, you must uninstall any pre-existing profiler, such as Ant, VS 2010 Performance Tools, or others. The utility alerts you if it finds a pre-existing profiler.

⚠ To apply configurations, the .NET Agent Configuration Utility must restart IIS. The utility offers you the option to restart IIS or not. If you choose not to restart, configurations apply the next time IIS restarts.

File System Security Settings

For the App Agent for .NET to instrument your application correctly, ensure the following Windows accounts have the required file system access permissions:

- The account you use to run your web application as defined by its application pool or the Windows service account.
- The account you use to run the AppDynamics Agent Coordinator, by default the Local System account.

The required permissions are as follows:

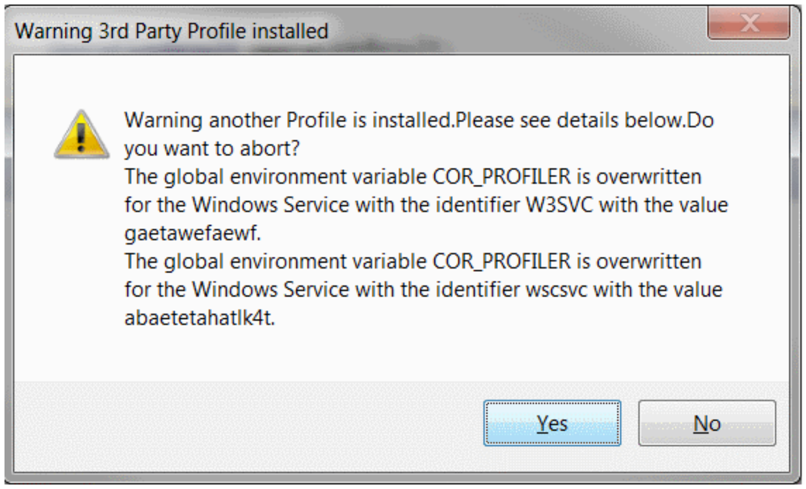
- **Write** permission to the .NET App Agent logs directory:
  For agent version 3.7.8 or later, the default is as follows:
  Windows Server 2008 and later: `%ProgramData%\AppDynamics\DotNetAgent\Logs`
  Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs`
  For agent version 3.7.7 or earlier, the default is C:\Appdynamics\Logs.
- **Read** and **Execute** permissions to to the .NET App Agent install directory, by default `C:\Program Files\AppDynamics\AppDynamics .NET Agent`
- **Read** and **Execute** permissions the web application installation directory, for example `C:\inetpub\wwwroot\myapp`

Configuring an App Agent for .NET

To access the .NET Agent Configuration Utility

1. In the Windows menu, click **AppDynamics -> .NET Agent -> AppDynamics Agent Configuration**.

2. If the "Warning: 3rd Party Profiler installed" message displays, it means that the configuration utility has discovered another profiler. Windows only allows one profiler per machine. Because AppDynamics uses a profiler you must uninstall any other profilers.


Warning 3rd Party Profile installed

Warning another Profile is installed.Please see details below.Do you want to abort?
The global environment variable COR_PROFILER is overwritten for the Windows Service with the identifier W3SVC with the value gaetawefaewf.
The global environment variable COR_PROFILER is overwritten for the Windows Service with the identifier wscsvc with the value abaetetahatlk4t.

Click **Yes** to exit and uninstall any pre-existing profiler. Check the registry to make sure that the uninstall process cleaned up the registry entries. Use the warning message to identify any undeleted profiler environment variables.

Video Tutorial: Manual Installation And Configuration

3. When the configuration utility detects legacy agent configurations from version 3.7.7 or earlier, it displays the **Upgrade Configuration** window.

- Answer **Yes** to remove legacy configurations. For a list of affected configurations, See To clean up legacy configurations.

  ⚠ Removing legacy configurations modifies web.config files causing IIS to restart affected applications.

- Answer **No** to leave legacy configurations in place. You can remove them at a later time.

4. When the utility discovers no further profiler conflicts or after any configuration clean up, the welcome window displays.

Click **Next** to advance to **Log directory permissions**.

To set up the logs and account permissions

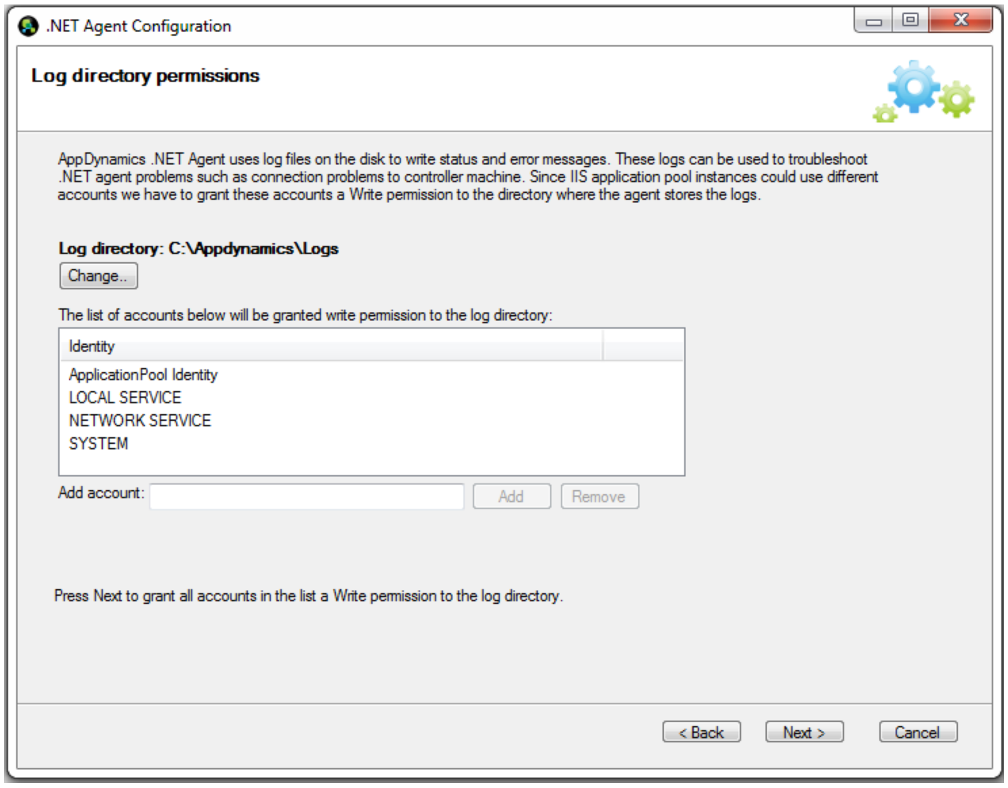
The first window helps you set up the location of the agent logs and provide the correct account access to the logs.

1. If you want to change the default location of the log directory, click **Change** and select a new location.

ℹ The default logs directories are as follows:
Windows Server 2008 and later: `%ProgramData%\AppDynamics\DotNetAgent\Logs`
Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs`

2. If needed, add accounts for log directory permissions.
Commonly-used accounts are provided. If your application uses another account, enter the Windows account you use to run your web application as defined by its application pool or the

Windows service account. The account name must be valid on the operating system and have permission to write to the log files directory.


.NET Agent Configuration

Log directory permissions

AppDynamics .NET Agent uses log files on the disk to write status and error messages. These logs can be used to troubleshoot .NET agent problems such as connection problems to controller machine. Since IIS application pool instances could use different accounts we have to grant these accounts a Write permission to the directory where the agent stores the logs.

Log directory: C:\Appdynamics\Logs

Change..

The list of accounts below will be granted write permission to the log directory:

Identity
ApplicationPool Identity
LOCAL SERVICE
NETWORK SERVICE
SYSTEM

Add account: Add Remove

Press Next to grant all accounts in the list a Write permission to the log directory.

< Back Next > Cancel

3. Click **Add**. If you get a warning message, make sure that the account is valid on the system.

4. Click **Next** and the wizard confirms the list of accounts.

5. Click **Next** to advance to **Controller Configuration**.

To provide Controller configuration information

1. Enter the Controller access information and credentials.

   - The App Agent for .NET configuration utility only supports configuration of one Controller and business application per server. Use tiers to organize different applications you instrument on a single server.
   - For a SaaS Controller, enter the server name or IP, port number, account name, and access key as provided to you by AppDynamics.
   - For an on-premise Controller, if you haven't already installed it, cancel this installation and see Install the Controller. Otherwise enter the server name and port number of an existing Controller.
   - For a secure connection, click **Enable SSL**.
     The Controller must use a trusted certificate.
   - If needed, fill in the proxy information. Proxies that use authentication require additional configuration.

2. Click **Test Controller Connection** to verify the connection.

3. Click **Next** to advance to the **Application Configuration**

AppDynamics retrieves existing business application information from the Controller and displays it in the left column. Controller connection status displays on the right.

4. Click **Existing Applications from the Controller** to select business applications from the Controller.

If you haven't defined business applications in the Controller, the utility displays an empty list.

Click **New Application** to define a new business application. Be careful about spellings and capitalization and note down the exact name.

Do not use ampersands in the business application name; they are not supported at this time.

5. Click **Next** to advance to **Assign IIS applications to tiers**.

To map business applications, tiers, and nodes to your application environment

1. Read about how AppDynamics uses business applications, tiers, and nodes to organize application performance monitoring. In summary:

- A business application is a set of modules and distributed services that together provide business functionality.
- A node is the basic unit of processing that AppDynamics monitors.
- A tier represents a module in an application environment, such as an eCommerce website or Inventory application.

See Logical Model.

2. Decide how to identify and name the tiers. Either AppDynamics will automatically configure tier names, or you can manually configure them.

Use these guidelines for deciding whether to use automatic or manual naming:

- To have AppDynamics instrument all IIS applications on a machine, choose automatic.
- AppDynamics names tiers using this pattern:

```
IIS_site_name-IIS_application_name
```

- To select the IIS applications on a machine to instrument with AppDynamics, choose manua l.
- Using manual naming, you supply the tier names and AppDynamics updates the configuration file.

To let AppDynamics automatically name the tiers

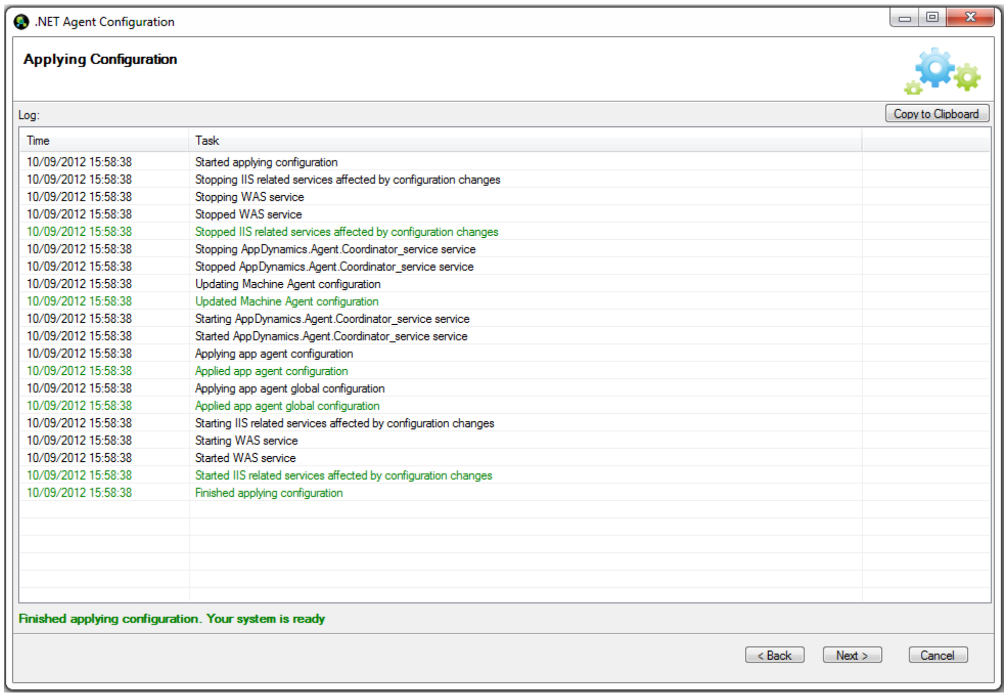
1. In the **Assign IIS applications to tiers** window click **Automatic**.

2. If prompted, click **OK** to confirm Automatic configuration.

The configuration utility summarizes the configuration settings.

3. By default when you click **Next** the configuration utility restarts IIS.
⚠ If you do not want to apply the configuration right away, uncheck the box. The Configuration Utility saves the information and applies it the next time you restart IIS.

4. If you proceed and click **Next**, the configuration utility logs its activities, including stopping and restarting IIS, and reports any problems. Review the summary for any issues in red font. Green font indicates the more interesting logged events. The summary shows any Warnings (W) or Errors (E). If you have errors, contact AppDynamics Support.



```
Time                 Task
10/09/2012 15:58:38  Started applying configuration
10/09/2012 15:58:38  Stopping IIS related services affected by configuration changes
10/09/2012 15:58:38  Stopping WAS service
10/09/2012 15:58:38  Stopped WAS service
10/09/2012 15:58:38  Stopped IIS related services affected by configuration changes
10/09/2012 15:58:38  Stopping AppDynamics.Agent.Coordinator_service service
10/09/2012 15:58:38  Stopped AppDynamics.Agent.Coordinator_service service
10/09/2012 15:58:38  Updating Machine Agent configuration
10/09/2012 15:58:38  Updated Machine Agent configuration
10/09/2012 15:58:38  Starting AppDynamics.Agent.Coordinator_service service
10/09/2012 15:58:38  Started AppDynamics.Agent.Coordinator_service service
10/09/2012 15:58:38  Applying app agent configuration
10/09/2012 15:58:38  Applied app agent configuration
10/09/2012 15:58:38  Applying app agent global configuration
10/09/2012 15:58:38  Applied app agent global configuration
10/09/2012 15:58:38  Starting IIS related services affected by configuration changes
10/09/2012 15:58:38  Starting WAS service
10/09/2012 15:58:38  Started WAS service
10/09/2012 15:58:38  Started IIS related services affected by configuration changes
10/09/2012 15:58:38  Finished applying configuration
```

Finished applying configuration. Your system is ready

5. When there are no errors, click **Next**.

6. Click **Done** to close the Configuration Utility.

To manually name the tiers

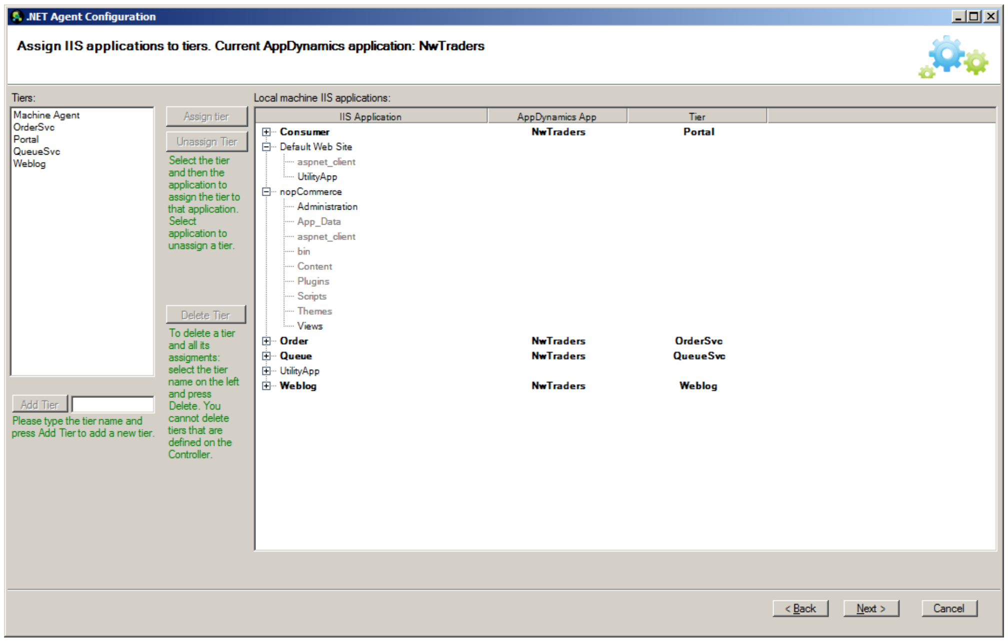
1. In the **Assign IIS applications to tiers** window click **Manual**, then click **Next**.

2. Assign IIS Applications to AppDynamics tiers.

Select a tier on the right and click a business application on the left. The assigned tier will be highlighted in boldface.

For large IIS installations, use the Max IIS tree depth pulldown to display all the projects. A large tree depth may take some time to view.

To create new tiers, enter a name and click **Add Tier**.



Assign IIS applications to tiers. Current AppDynamics application: NwTraders

Tiers:

- Machine Agent
- OrderSvc
- Portal
- QueueSvc
- Weblog

Select the tier and then the application to assign the tier to that application. Select application to unassign a tier.

To delete a tier and all its assigments: select the tier name on the left and press Delete. You cannot delete tiers that are defined on the Controller.

Please type the tier name and press Add Tier to add a new tier.

Local machine IIS applications:

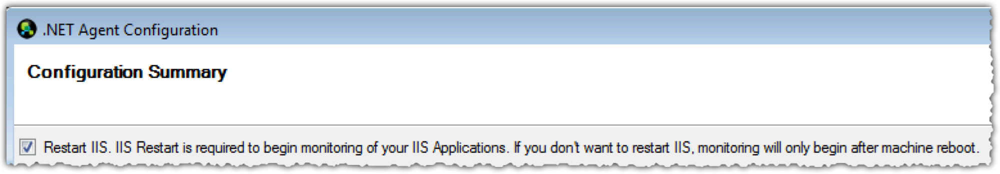
IIS Application	AppDynamics App	Tier
⊞ Consumer	NwTraders	Portal
⊟ Default Web Site		
aspnet_client		
UtilityApp		
⊟ nopCommerce		
Administration		
App_Data		
aspnet_client		
bin		
Content		
Plugins		
Scripts		
Themes		
Views		
⊞ Order	NwTraders	OrderSvc
⊞ Queue	NwTraders	QueueSvc
⊞ UtilityApp		
⊞ Weblog	NwTraders	Weblog

5. When you are done click **Next**. AppDynamics displays a configuration summary.

6. Review the configuration. If you need to make changes click **Back**.

7. On the **Configuration Summary** window, un-check **Restart IIS** if you don't want to immediately restart IIS.

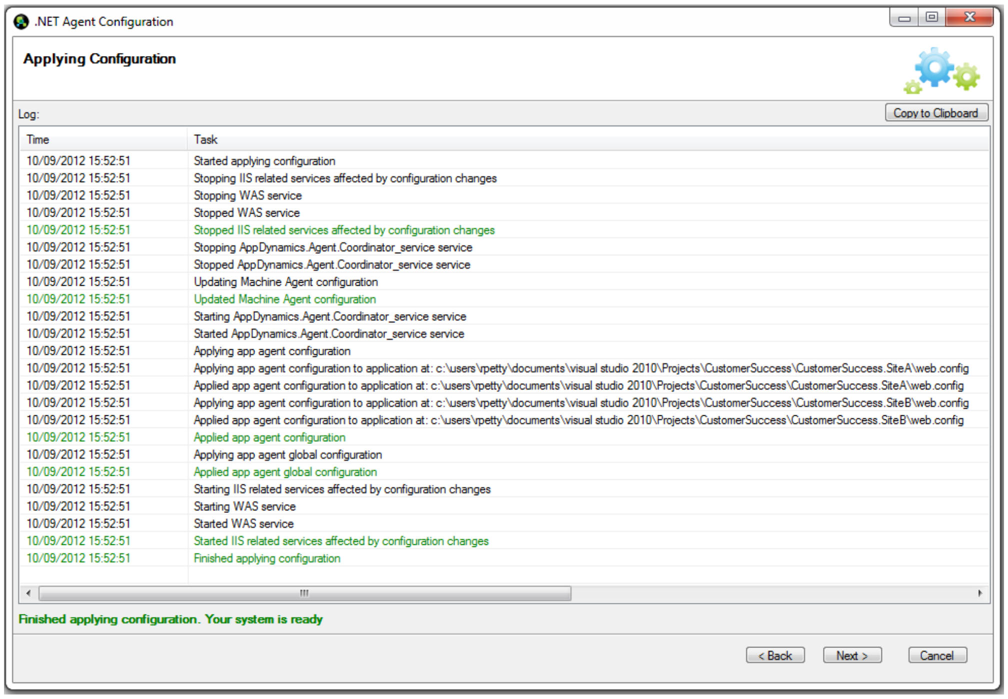
You may restart later to apply your changes, or they will take effect after a reboot.



Configuration Summary

☑ Restart IIS. IIS Restart is required to begin monitoring of your IIS Applications. If you don't want to restart IIS, monitoring will only begin after machine reboot.

7. If you proceed and click **Next**, the Configuration Utility logs its activities, including stopping and restarting IIS, and reports any problems.



Time	Task
10/09/2012 15:52:51	Started applying configuration
10/09/2012 15:52:51	Stopping IIS related services affected by configuration changes
10/09/2012 15:52:51	Stopping WAS service
10/09/2012 15:52:51	Stopped WAS service
10/09/2012 15:52:51	Stopped IIS related services affected by configuration changes
10/09/2012 15:52:51	Stopping AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Stopped AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Updating Machine Agent configuration
10/09/2012 15:52:51	Updated Machine Agent configuration
10/09/2012 15:52:51	Starting AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Started AppDynamics.Agent.Coordinator_service service
10/09/2012 15:52:51	Applying app agent configuration
10/09/2012 15:52:51	Applying app agent configuration to application at: c:\users\vpetty\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteA\web.config
10/09/2012 15:52:51	Applied app agent configuration to application at: c:\users\vpetty\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteA\web.config
10/09/2012 15:52:51	Applying app agent configuration to application at: c:\users\vpetty\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteB\web.config
10/09/2012 15:52:51	Applied app agent configuration to application at: c:\users\vpetty\documents\visual studio 2010\Projects\CustomerSuccess\CustomerSuccess.SiteB\web.config
10/09/2012 15:52:51	Applied app agent configuration
10/09/2012 15:52:51	Applying app agent global configuration
10/09/2012 15:52:51	Applied app agent global configuration
10/09/2012 15:52:51	Starting IIS related services affected by configuration changes
10/09/2012 15:52:51	Starting WAS service
10/09/2012 15:52:51	Started WAS service
10/09/2012 15:52:51	Started IIS related services affected by configuration changes
10/09/2012 15:52:51	Finished applying configuration

Finished applying configuration. Your system is ready

8. Review the configuration log summary.
As it applies the configuration, AppDynamics generates a log of the configuration activities and displays a summary. Review the summary for any issues in red font. Green font indicates the more interesting logged events. The summary shows any Warnings (W) or Errors (E). If you have errors, contact AppDynamics Support.

9. Click **Next**. The wizard completes.

For troubleshooting information see Resolve App Agent for .NET Installation and Configuration Issues.

Using a Configuration File from the Command Line

You can set up a .NET Agent configuration file and run it from the command line. This is useful when you have multiple agents to configure.

To create a configuration file

1. From a command line, start the configuration utility:

```
AppDynamics.Agent.Winston.exe -s <SetupConfigurationFilePath>
```

For example:

```
c:\Program Files\AppDynamics\AppDynamics .NET
Agent\AppDynamics.Agent.Winston.exe -s
"c:\temp\configurationSavedSetupConfiguration.xml"
```

The utility starts.
2. Configure the agent as described in the previous sections. The configuration is applied.
In addition, when the configuration completes, AppDynamics creates a setup file.

Use this setup file as an argument to the command line utility.

To perform an unattended installation, pass the configuration file directly to the MSI installer package. See Unattended Installation for .NET.

To run the configuration utility from the command line

1. Start the .NET Configuration Utility from the command line. Change the file path and setup file path as needed.

```
AppDynamics.Agent.Winston.exe -c <SetupConfigurationFilePath>
```

For example:

```
c:\Program Files\AppDynamics\AppDynamics .NET
Agent\AppDynamics.Agent.Winston.exe -c
"c:\temp\configurationSavedSetupConfiguration.xml"
```

The utility runs in command line mode; the user interface does not launch.

When it finishes, the utility exits the process with status 0 for success or any other number for failure.

2. Review the Winston.txt log file in the default logs directory for details.

Learn More

- Video Tutorial: App Agent for .NET Manual Installation and Configuration
- Install the App Agent for .NET
- Naming Conventions for .NET Nodes

App Agent for .NET Directory Structure

This topic details the default installation directories for the App Agent for .NET.

App Agent for .NET Directory Structure

The App Agent for .NET installs to the following directories. To navigate to a directory, copy the path and paste it into the Windows Explorer address bar.

Executables

The agent executables and supporting files install to the **AppDynamics .NET Agent** directory, the same as previous versions:

Windows Server 2003 and later

```
%ProgramFiles%\AppDynamics\AppDynamics .NET Agent
```

Log files

The **Logs** directory defaults to the following location:

Windows Server 2008 and later

```
%ProgramData%\AppDynamics\DotNetAgent\Logs
```

Windows Server 2003

```
%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs
```

Agent configuration files

The unified configuration file config.xml installs to the **Config** directory:

Windows Server 2008 and later

```
%ProgramData%\AppDynamics\DotNetAgent\Config
```

Windows Server 2003

```
%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Config
```

Coordinator Service configuration files

The AppDynamics Agent Coordinator service writes configuration files from the Controller to the **D ata** directory:

Windows Server 2008 and later

```
%ProgramData%\AppDynamics\DotNetAgent\Data
```

Windows Server 2003

```
%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Data
```

About Windows Server system directory variables

%ProgramFiles% is located at **<system drive>\Program Files** for Windows Server 2003 and later.

%ProgramData% is located at **<system drive>\Program Data** for Windows Server 2008 and later.

%AllUsersProfile% is located at **<system drive>\Documents and Settings\All Users** for Windows 2003.

Unattended Installation for .NET

This topic describes the process to run an unattended installation for the App Agent for .NET. For more detail about how to install and configure the agent manually see Install the App Agent for .NET.

Prepare for Unattended Installation

The App Agent for .NET MSI installer package allows you to specify the path to a setup configuration file to perform an unattended installation. The setup configuration file contains all the properties you need to enable instrumentation for your .NET applications.

To create a setup configuration file

You must run the .NET MSI installer package on one computer before you can use the AppDynamics Agent Configuration utility to create a setup configuration file. See Install the App Agent for .NET.

Setup configuration files created in previous versions of the AppDynamics Agent Configuration utility work with the 3.8 installer.

1. Launch the AppDynamics Agent Configuration utility from the command line. Use the **-s** parame

ter to specify the setup configuration file destination. See To create a configuration file.

```
AppDynamics.Agent.Winston.exe -s <path to setup configuration file>
```

2. Go through the configuration wizard normally.

The configuration utility saves the setup configuration file to the path you specified.

The configuration utility only configures instrumentation for IIS applications.

3. Optional: To perform unattended installation for Windows services or for standalone applications, you must edit the setup configuration file manually.

- For Windows services, add a Windows Services block as a child of the App Agents element. See Configuring the App Agent for .NET for Windows Services.
- For standalone applications, add a Standalone Applications block as a child of the App Agents element. See Configuring the App Agent for .NET for Standalone Applications.

Sample setup configuration file

The following example shows a setup configuration file that instruments two IIS Applications (MainBC and SampleHTTPService), a Windows service (BasicWindowsService), and a standalone application (MyStandaloneApp.exe).

The configuration file sets the log directory as C:\ProgramData\AppDynamics\DotNetAgent\Logs and grants write permission to four accounts.

```
<winston>
  <logFileDirectory
directory="C:\ProgramData\AppDynamics\DotNetAgent\Logs" />
  <logFileFolderAccessPermissions defaultAccountsEnabled="false">
    <account name="NT AUTHORITY\LOCAL SERVICE" displayName="LOCAL
SERVICE" />
    <account name="NT AUTHORITY\SYSTEM" displayName="SYSTEM" />
    <account name="NT AUTHORITY\NETWORK SERVICE" displayName="NETWORK
SERVICE" />
    <account name="IIS_IUSRS" displayName="ApplicationPool Identity"
/>
  </logFileFolderAccessPermissions>
  <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>
          <application path="/" site="MainBC">
            <tier name="Main Site" />
          </application>
          <application path="/" site="SampleHTTPService">
            <tier name="HTTP Services" />
          </application>
        </applications>
      </IIS>
      <windows-services>
         <windows-service name="BasicWindowsService">
            <tier name="Service Tier"/>
         </windows-service>
      </windows-services>
      <standalone-applications>
         <standalone-application executable="MyStandaloneApp.exe">
            <tier name="Standalone App" />
         </standalone-application>
      </standalone-applications>
    </app-agents>
  </appdynamics-agent>
</winston>
```

Perform Unattended Installation

After you have created a setup configuration file, use it to run an unattended installation.

Installation prerequisites

You must enable COM+ for the agent to function. See Verify COM+ Services are enabled.

To perform an unattended installation

1. Uninstall any existing agent.

2. Launch an elevated command prompt with full administrator privileges. See Start a Command Prompt as an Administrator.

⚠ Logging on to Windows as a member of the Administrators group does not grant sufficient permissions to run the installer.

3. Run the agent MSI installer package from the elevated command prompt. Use the **AD_SetupFile** parameter to pass the absolute file path to the setup configuration file.

```
msiexec /i dotNetAgentSetup64.msi /q /norestart /lv
%ProgramData%\AppDynamics\DotNetAgent\AgentInstaller.log AD_SetupFile=<absolute
path to setup config.xml>
```

✓ Optionally use the **INSTALLDIR** parameter to customize the agent installation directory.

```
INSTALLDIR=<custom agent install directory>
```

For example:

```
msiexec /i dotNetAgentSetup64.msi /q /norestart /lv
%ProgramData%\AppDynamics\DotNetAgent\AgentInstaller.log
AD_SetupFile=C:\temp\SetupConfig.xml INSTALLDIR=D:\AppDynamics
```

4. Restart the AppDynamics.Agent.Coordinator.

```
net stop AppDynamics.Agent.Coordinator
net start AppDynamics.Agent.Coordinator
```

5. Restart applications you have instrumented: IIS, Windows services, or standalone applications.

For example, to restart IIS:

```
iisreset
```

Setup Configuration File Properties

Winston element

The Winston element is the root element for the configuration file.

Required element: `<winston>`

Log File Directory element

The Log File Directory element is a child element of the Winston element. Use the **directory** attrib ute to specify the log directory. If you omit the Log File Directory element, we use the default directory:

Windows Server 2008 and later: `%ProgramData%\AppDynamics\DotNetAgent\Logs`
Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs`

Optional element: `<logFileDirectory directory="C:\ProgramData\AppDynamics\DotNetAgent\Logs" />`

Log File Folder Access Permissions element

The Log File Folder Access Permissions is a child element of the Winston element. Unless you set the **default accounts enabled** attribute to false, we grant write access to the logs folder for the default accounts:

- LOCAL SERVICE
- SYSTEM
- NETWORK SERVICE
- ApplicationPool Identity

Optional element: `<logFileFolderAccessPermissions defaultAccountsEnabled="false">`

Account element

The Account element is a child element of the Log File Folder Access Permissions element. Create an **Account** element for the Windows account you use to run your application.

Set the **name** attribute to the name of account you use to run your application: the account for the application pool for IIS or the Windows service account.

The **display name** attribute is a user-friendly name you choose for the account. The display name shows up in log entries about assigning permissions for the account.

Optional element: `<account name="MyAppPoolIdentity" displayName="Custom ApplicationPool Identity" />`

For example, if you run a Windows service under a domain account:

```
<account name="MYDOMAIN\service_acct" displayName="Domain Service
Account" />
```

AppDynamics Agent element

The AppDynamics Agent element is a child of the Winston element. It follows the same format as the config.xml to define the agent configuration for all your .NET applications. See App Agent for .NET Configuration Properties.

Required element: `<appdynamics-agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">`

Learn More

Configure the App Agent for .NET
App Agent for .NET Configuration Properties
Enable the App Agent for .NET for Windows Services
Enable the App Agent for .NET for Standalone Applications

Upgrade the App Agent for .NET

This topic describes how to upgrade to the App Agent for .NET version 3.8. The instructions vary based upon your current version of the App Agent of .NET.

Upgrade the App Agent for .NET from Version 3.7.8 or Later

The MSI installer package for the new version of the App Agent for .NET (the agent) installs the updated agent files and maintains all legacy configurations. After you complete the installation, start the AppDynamics.Agent.Coordinator service and instrumented applications to finish the upgrade.

To uninstall the old version of the agent

1. Stop IIS, instrumented Windows services, and instrumented standalone applications.

⚠ Failing to stop instrumented applications before uninstalling the App Agent for .NET may require you to reboot the machine.

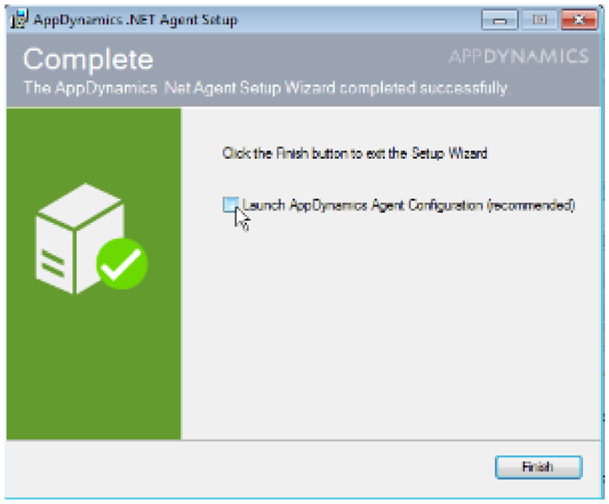
2. Stop the **AppDynamics.Agent.Coordinator** service.

3. In the Control Panel, select Add/Remove Programs. Remove the **AppDynamics .NET Agent**.

To install the new version of the agent

See To install the App Agent for .NET.

✓ Don't launch AppDynamics Agent Configuration when the installer completes unless you want to make changes to the existing configuration. The installer maintains existing agent configurations.


AppDynamics .NET Agent Setup

Complete

The AppDynamics .Net Agent Setup Wizard completed successfully

Click the Finish button to exit the Setup Wizard

[ ] Launch AppDynamics Agent Configuration (recommended)

To resume monitoring

1. Start the **AppDynamics.Agent.Coordinator** service.

```
net start AppDynamics.Agent.Coordinator
```

2. Start IIS, instrumented Windows services, and instrumented standalone applications.

Upgrade the App Agent for .NET from Version 3.7.7 or Earlier

Identify the right upgrade path based upon the method of tier naming and assignment (manual or automatic) and the type of application you instrument:

- If you use manual tier naming and assignment, the installer package upgrades configurations for IIS applications and Windows services.
- If you used automatic tier naming and assignment, run the configuration utility to update configurations.
- If you used standalone applications with 3.7.7 or earlier, follow the steps to Enable the App Agent for .NET for Standalone Applications.

The MSI installer package for the new version installs the updated agent files. After installing, you may need to run the configuration utility to update your configuration and optionally remove legacy configurations. Finally, restart the AppDynamics.Agent.Coordinator service and instrumented applications.

To uninstall the old version of the App Agent for .NET

1. Stop IIS and instrumented Windows services.

⚠ Failing to stop instrumented applications before uninstalling the App Agent for .NET may require you to reboot the machine.

2. Stop the **AppDynamics.Agent.Coordinator** service.

3. In the Control Panel, select Add/Remove Programs. Remove the **AppDynamics .NET Agent**.

To install the new version of the App Agent for .NET

See To install the App Agent for .NET.

If you used the following environment variables with the earlier version, the MSI installer migrates the configurations to the new configuration file:

- AppDynamicsAgent_CallGraphOptions
- AppDynamicsAgent_DisableAppPools
- AppDynamicsAgent_EnableInProcesses
- AppDynamicsAgent_IgnoreCLREnv
- AppDynamicsAgent_Profiler_Classes

Configure the App Agent for .NET

Configure the agent based on your method of tier generation and assignment: automatic or manua l.

ℹ The App Agent for .NET configuration utility only supports configuration of one Controller per server. If you previously used unsupported methods to configure different applications on the same server to connect to different Controllers, you must reconfigure your logical model.

To configure the agent using automatic tier generation and assignment

If you used automatic configuration with the earlier version of the App Agent for .NET, run the configuration utility to configure the agent:

1. Use the .NET Agent Configuration utility to reconfigure instrumentation for IIS applications. Choose **Automatic** for the method of tier generation and assignment. See Configure the App Agent for .NET.

2. Configure instrumentation for Windows services manually. See Enable the App Agent for .NET for Windows Services.

To configure the agent using manual tier generation and assignment

For manual systems using manual tier generation and assignment, the installer package migrates the configurations for IIS applications and for Windows services to the config.xml. At this stage, the configuration for IIS applications and Windows services is complete.

If you choose not to launch the configuration utility and clean up legacy configurations, restart the **AppDynamics.Agent.Coordinator** service.

```
net stop AppDynamics.Agent.Coordinator
net start AppDynamics.Agent.Coordinator
```

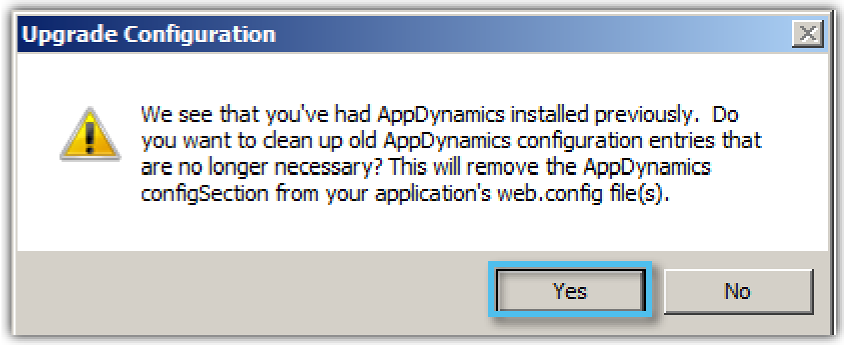
To clean up legacy configurations

You can clean up legacy configurations by launching the AppDynamics Agent Configuration utility. When the utility detects agent settings from a previous version, it offers you the option to clean up.

The clean up procedures modifies the web.config files causing an IIS restart.

1. Launch the AppDynamics Agent Configuration utility.
The Upgrade Configuration window opens.

2. Answer **Yes** to clean up old AppDynamics configurations.



We see that you've had AppDynamics installed previously. Do you want to clean up old AppDynamics configuration entries that are no longer necessary? This will remove the AppDynamics configSection from your application's web.config file(s).

The utility removes the following configurations:

- AppDynamics configSections from web.config files for IIS applications and from application.config files for Windows services.
- Environment variables:
  - AppDynamicsAgent_IgnoreCLREnv
  - AppDynamicsAgent_CallGraphOptions
  - AppDynamicsAgent_EnableInProcesses
  - AppDynamicsAgent_DisableAppPools
  - AppDynamicsAgent_Profiler_Classes

3. Proceed through the wizard normally:

- Verify or update the log directory and grant write permissions to it.
- Verify the controller connection information.
- Verify or update manual tier assignment.

To resume monitoring

Start IIS and instrumented Windows services.

Updated App Agent for .NET Directory Structure

To learn more about the updated App Agent for .NET Directory Structure, See App Agent for .NET Directory Structure.

Learn More

- Install the App Agent for .NET
- Configure the App Agent for .NET
- Uninstall the App Agent for .NET
- Agent - Controller Compatibility Matrix
- Release Notes for AppDynamics Pro
- Resolve App Agent for .NET Installation and Configuration Issues

Enable SSL for .NET

This topic covers how to configure the App Agent for .NET (the agent) to connect to the Controller using SSL. It assumes that you use a SaaS Controller or have configured the on-premise Controller to use SSL.

Before You Begin

Before you configure the agent to enable SSL, gather the following information:

- Identify if the Controller is SaaS or on-premise.
- Identify the Controller SSL port.
    - For SaaS Controllers the SSL port is 443.
    - For on-premise Controllers the default SSL port is 8181, but you may configure the Controller to listen for SSL on another port.
- Identify the signature method for the Controller's SSL certificate:
    - A publicly known certificate authority (CA) signed the certificate. This applies for Verisign, Thawte, and other commercial CAs.
    - A CA internal to your organization signed the certificate. Some companies maintain internal certificate authorities to manage trust and encryption within their domain.
    - The Controller uses a self-signed certificate.

Enable SSL for the App Agent for .NET

There are two ways to update the SSL settings for the agent. You can use the AppDynamics Agent Configuration Utility. Otherwise, edit the settings directly in the config.xml, see Where to Configure App Agent Properties.

When you enable SSL for the App Agent for .NET, you automatically enable SSL for the .NET Machine Agent.

To configure SSL using the AppDynamics Agent Configuration utility

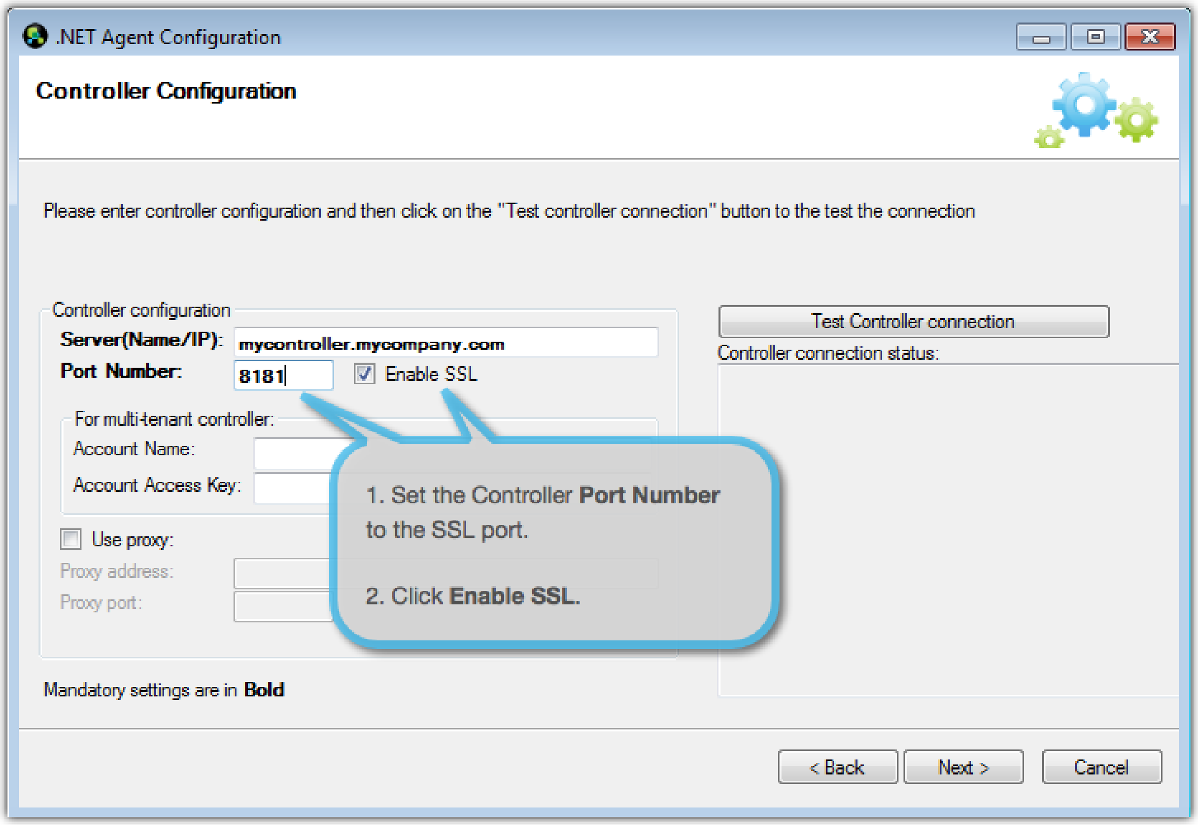
1. Launch the AppDynamics Agent Configuration utility.

2. In the **Controller Configuration** window, set the **Port Number** to the SSL port for the Controller.

- For a **SAAS Controller**, set the **Port Number** to 443.
- For an **on-premise Controller**, set the **Port Number** to the on-premise SSL port. The default is 8181.

3. Click **Enable SSL**.

This example demonstrates connection to an on-premise Controller listening for SSL on port 8181:


.NET Agent Configuration

Controller Configuration

Please enter controller configuration and then click on the "Test controller connection" button to the test the connection

Controller configuration

Server(Name/IP): mycontroller.mycompany.com

Port Number: 8181 ☑ Enable SSL

For multi-tenant controller:

Account Name:

Account Access Key:

☐ Use proxy:

Proxy address:

Proxy port:

Controller connection status:

Mandatory settings are in **Bold**

Test Controller connection
< Back
Next >
Cancel
1. Set the Controller Port Number to the SSL port.
2. Click Enable SSL.

4. Click **Next** and proceed with the rest of the windows to complete the configuration.

5. Restart instrumented applications: IIS applications or application pools, Windows services, or standalone applications.

If you use automatic tier configuration, restart IIS. For example, open a command prompt and run:

```
iisreset
```

Upon restart the agent connects with the Controller via SSL.

To configure SSL in the config.xml

1. Open the config.xml file as administrator. See Where to Configure App Agent Properties.

2. Update the SSL settings. See Controller Element.

- Set the **Controller port attribute** to the on-premise SSL port. The default is 8181. See Controller port attribute.
- Set the **Controller SSL attribute** to "true". See Controller ssl attribute.

3. Save your changes.

4. Restart the AppDynamics.Agent.Coordinator service.

5. Restart instrumented applications: IIS applications or application pools, Windows services, or standalone applications.

If you use Automatic configuration, restart IIS. For example, open a command prompt and run:

```
iisreset
```

Upon restart the agent connects with the Controller via SSL.

Sample SaaS SSL config.xml configuration

```
<?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="mycompany.saas.appdynamics.com" port="443" ssl="true">
    <application name="MyDotNetApplication" />
  </controller>
...
</appdynamics-agent>
```

Sample on-premise SSL config.xml configuration

```
<?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="8181" ssl="true">
    <application name="MyDotNetApplication" />
  </controller>
...
</appdynamics-agent>
```

Establish Trust for the Controller's SSL Certificate

The App Agent for .NET requires that the Common Name (CN) on the Controller certificate match the DNS name of the Controller. Additionally, certificates for the root CA that signed the Controller's SSL certificate must reside in the **Windows Trusted Root Certification Authorities** s tore for the **Local Computer**.

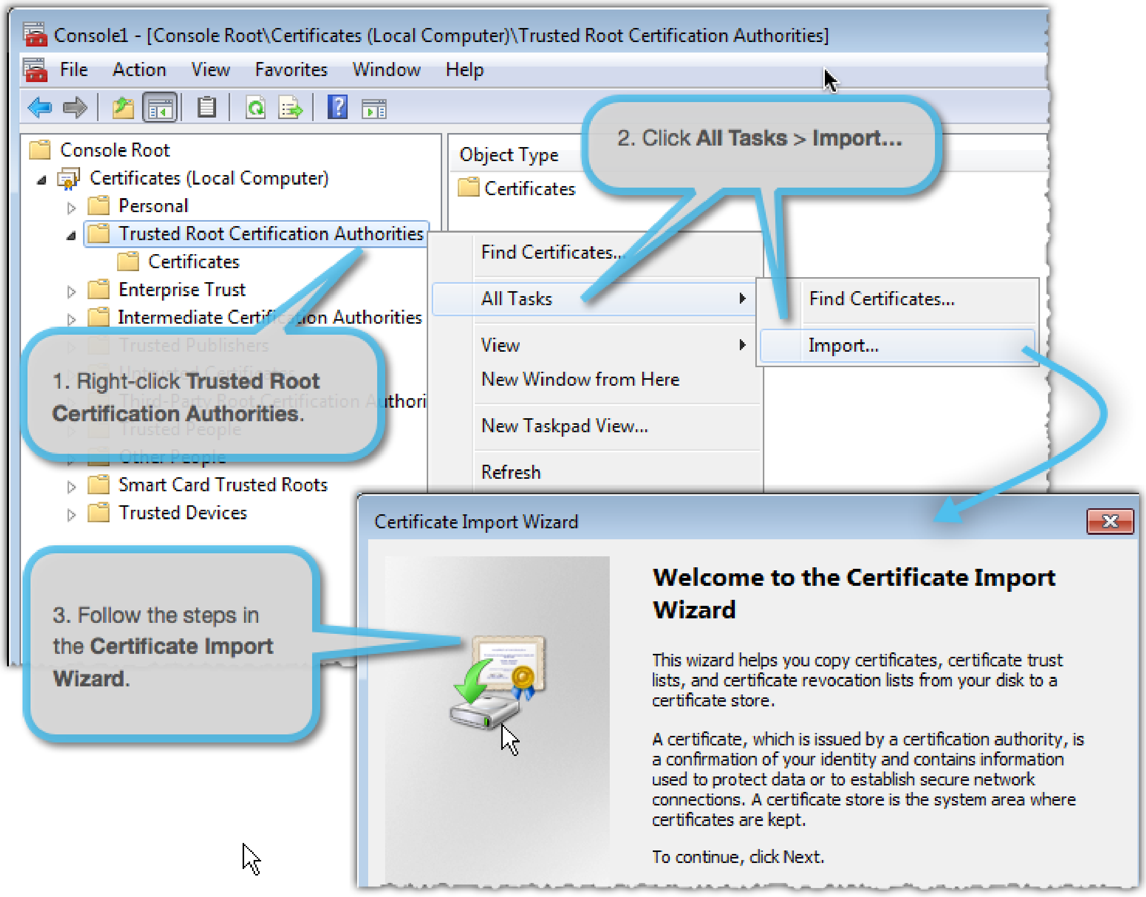
Certificates signed by a publicly known Certificate Authority

The root certificates for most publicly trusted CA signing authorities, such as Verisign, Thawte, and other commercial CAs, are in the Trusted Root Certification Authorities store by default.

Certificates signed by an Internal Certificate Authority

If your organization uses internal CA to sign certificates, you may need to obtain the root CA certificate from your internal security management resource. To import the root certificate, see Add ing certificates to the Trusted Root Certification Authorities store for a local computer.

This example shows how to use the Certificate snap-in for the Microsoft Management Console to import a certificate for a Trusted Root Certification Authority:


Console1 - [Console Root\Certificates (Local Computer)\Trusted Root Certification Authorities]
File Action View Favorites Window Help
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Certificates
Enterprise Trust
Intermediate Certification Authorities
Smart Card Trusted Roots
Trusted Devices
Object Type
Certificates
Find Certificates...
All Tasks
View
New Window from Here
New Taskpad View...
Refresh
Find Certificates...
Import...
1. Right-click Trusted Root Certification Authorities.
2. Click All Tasks > Import...
3. Follow the steps in the Certificate Import Wizard.
Certificate Import Wizard
Welcome to the Certificate Import Wizard

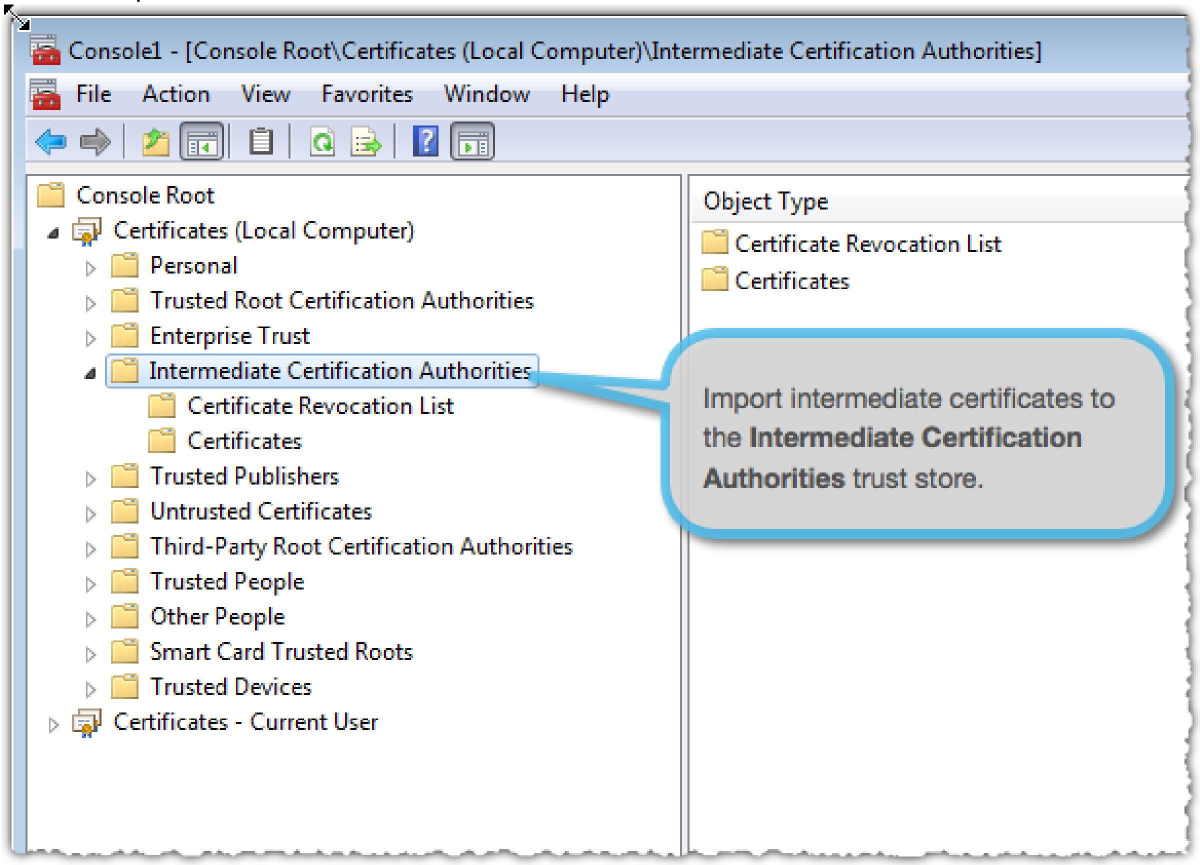
This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

If an intermediate CA signed the Controller's certificate, you must import the certificate for the intermediate CA in addition to the one for the root CA that signed the intermediate CA's certificate. If your controller is publicly accessible, you can use a certificate checker to identify the certificates required to complete the trust chain. See the certificate checker from Thawte.

This examples shows the **Intermediate Certification Authorities** store:


Console1 - [Console Root\Certificates (Local Computer)\Intermediate Certification Authorities]
File Action View Favorites Window Help
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Certificate Revocation List
Certificates
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Other People
Smart Card Trusted Roots
Trusted Devices
Certificates - Current User
Object Type
Certificate Revocation List
Certificates

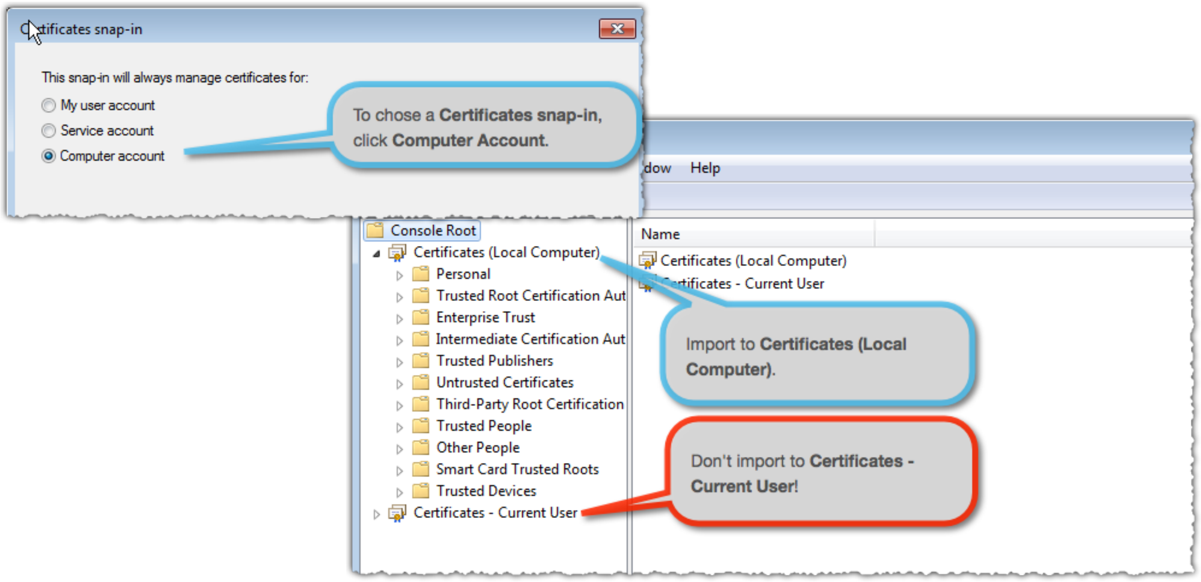
Import intermediate certificates to the **Intermediate Certification Authorities** trust store.

Self-Signed Certificates

The App Agent for .NET does not support self-signed certificates. In order to implement SSL, the Controller must use a certificate signed by a trusted CA signing authority or an internal trusted root CA. See Implement Security.

Troubleshooting Tips

- If you imported certificates for a root or intermediate CA, verify the certificate store where you imported them. Import them to **Certificates (Local Computer)**.


Certificates snap-in
This snap-in will always manage certificates for:
My user account
Service account
Computer account
To chose a Certificates snap-in, click Computer Account.
dow Help
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Aut
Enterprise Trust
Intermediate Certification Aut
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification
Trusted People
Other People
Smart Card Trusted Roots
Trusted Devices
Certificates - Current User
Name
Certificates (Local Computer)
Certificates - Current User
Import to Certificates (Local Computer).
Don't import to Certificates - Current User!

- The AppDynamics SaaS Controller uses certificates signed by Thawte. In some cases, SaaS customers must import the Thawte root certificates into the **Windows Trusted Root Certification Authorities** store.
- In some cases system administrators set up group policies that require external certificates be imported to the **Third-Party Root Certification Authorities** store. If importing the certificate for the root CA to the Windows Trusted Certification Authorities store doesn't work, try the Third-Party Root Certification Authorities store.

Learn More

Implement Security
App Agent for .NET Configuration Properties

Enable the App Agent for .NET for Windows Services

- Preparing to Configure the App Agent for .NET for Windows Services
- Configuring the App Agent for .NET for Windows Services
- Learn More

By default, AppDynamics enables the App Agent for .NET only for IIS worker processes. This topic describes how to edit the config.xml file to enable the agent for Windows services.

Preparing to Configure the App Agent for .NET for Windows Services

Before you enable the App Agent for .NET for Windows services, you must install the agent. We recommend using the AppDynamics Agent Configuration utility to perform basic configuration tasks.

1. If you have not already done so, install the App Agent for .NET. See Install the App Agent for .NET
2. Run the AppDynamics Agent Configuration utility to generate a config.xml. See Configure

the App Agent for .NET.
Even though the Configuration Utility only instruments IIS Applications, it is useful to create the config.xml and specify Controller connection properties.

If you have previously instrumented app agents for IIS applications, don't run the configuration utility. You already have a config.xml.

Use the configuration utility to do the following:

- Change the location of the Logs directory and assign permissions.
- Configure and test connectivity to the Controller.
- Set the Business Application for the agent.

Choose **Manual** for the method of tier generation and assignment. Don't assign assign any tiers for any IIS Applications. This disables instrumentation for all IIS applications.

Configuring the App Agent for .NET for Windows Services

Once you have configured the Controller properties for the App Agent for .NET, instrument your Windows service by adding an XML element for it to the config.xml.

1. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.

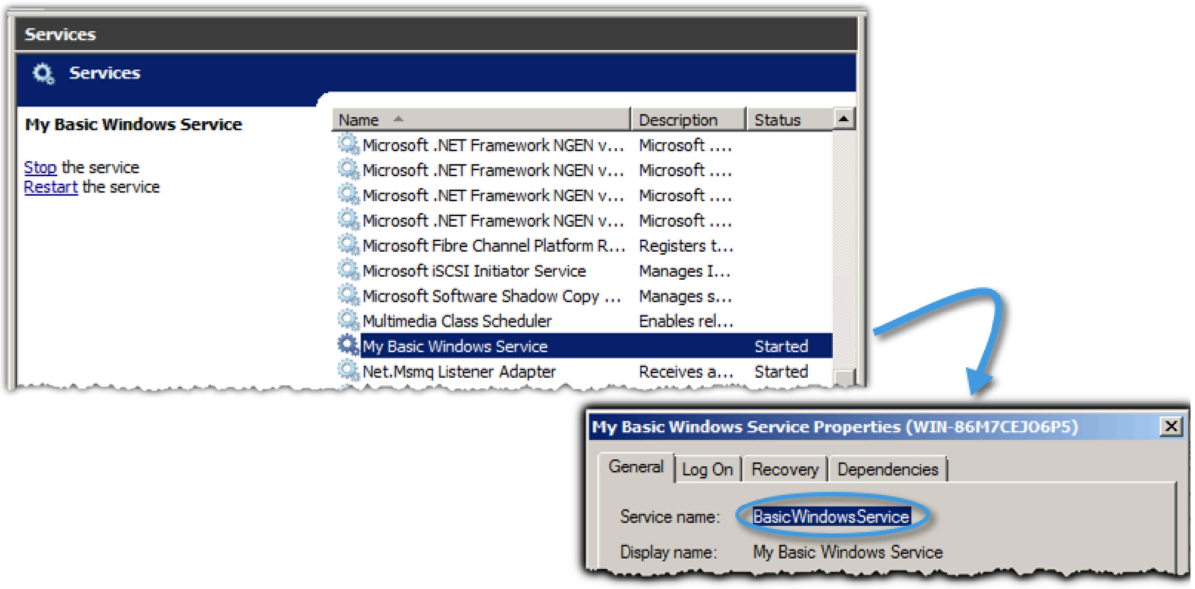
   If you haven't instrumented IIS applications, the file contains minimal configurations for Controller connectivity and the machine agent. Verify the Controller properties and the Business Application name:

   ```
   <?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>
     </app-agents>
   </appdynamics-agent>
   ```

   If you have already instrumented IIS applications, you see their configurations under the IIS element.
2. Add the Windows Services block as a child of the App Agents element. Create a Windows Service element for each Windows service you want to instrument. Use the Tier element to assign the instrumented service to a tier in the Controller. See App Agent for .NET Configuration Properties

```
<windows-services>
   <windows-service name="BasicWindowsService">
     <tier name="Service Tier"/>
   </windows-service>
   <windows-service name="SecondWindowsService">
     <tier name="Service Tier"/>
   </windows-service>
</windows-services>
```

3. Set the Windows Service element name attribute to the service name. The service name may not be the same as the display name in the Services panel of the Server manager.



My Basic Windows Service

Name	Description	Status
Microsoft .NET Framework NGEN v...	Microsoft	
Microsoft .NET Framework NGEN v...	Microsoft	
Microsoft .NET Framework NGEN v...	Microsoft	
Microsoft .NET Framework NGEN v...	Microsoft	
Microsoft Fibre Channel Platform R...	Registers t...	
Microsoft iSCSI Initiator Service	Manages I...	
Microsoft Software Shadow Copy ...	Manages s...	
Multimedia Class Scheduler	Enables rel...	
My Basic Windows Service		Started
Net.Msmq Listener Adapter	Receives a...	Started

My Basic Windows Service Properties (WIN-86M7CEJO6P5)

Service name: BasicWindowsService

Display name: My Basic Windows Service

This sample config.xml demonstrates instrumentation for a Windows service:

```
<?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>
    <windows-services>
        <windows-service name="BasicWindowsService">
           <tier name="Service Tier"/>
        </windows-service>
    </windows-services>
  </app-agents>
</appdynamics-agent>
```

4. Restart the AppDynamics.Agent.Coordinator
5. Restart the Windows service.

If your Windows service does not implement any of the frameworks we instrument by default, you must configure a POCO entry point for a class/method in your service for the agent to begin instrumentation. See Supported Environments and Versions for .NET and POCO Entry Points.

Learn More

- Mapping Application Services to the AppDynamics Model
- Configure the App Agent for .NET
- Install the App Agent for .NET
- App Agent for .NET Configuration Properties

Enable the App Agent for .NET for Standalone Applications

By default, AppDynamics enables the App Agent for .NET only for IIS worker processes. This topic describes how to edit the config.xml file to enable the agent for standalone applications.

Preparing to Configure the App Agent for .NET for Standalone Applications

Before you enable the App Agent for .NET for standalone applications, you must install the agent. We recommend using the AppDynamics Agent Configuration utility to perform basic configuration tasks.

1. If you have not already done so, install the App Agent for .NET. See Install the App Agent for .NET
2. Run the AppDynamics Agent Configuration utility to generate a config.xml. See Configure the App Agent for .NET.
   Even though the Configuration Utility only instruments IIS Applications, it is useful to create the config.xml and specify Controller connection properties.

   If you have previously instrumented app agents for IIS applications, don't run the configuration utility. You already have a config.xml.

   Use the configuration utility to do the following:
   - Change the location of the Logs directory and assign permissions.
   - Configure and test connectivity to the Controller.
   - Set the Business Application for the agent.

   Choose **Manual** for the method of tier generation and assignment. Don't assign assign any tiers for any IIS Applications. This disables instrumentation for all IIS applications.

Configuring the App Agent for .NET for Standalone Applications

Once you have configured the Controller properties for the App Agent for .NET, instrument your standalone application by adding an xml element for it to the config.xml.

1. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.
   If you haven't instrumented IIS applications, the file contains minimal configurations for Controlller connectivity and the machine agent. Verify the Controller properties and the Business Application name

```
<?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>
  </app-agents>
</appdynamics-agent>
```

If you have already instrumented IIS applications, you will see their configurations under the IIS element.

2. Add the Standalone Applications block as a child of the App Agents element. Create a Standalone Application element for each standalone application you want to instrument. Use the Tier element to assign the instrumented application to a tier in the Controller. See App Agent for .NET Configuration Properties

```
<standalone-applications>
   <standalone-application executable="MyStandaloneApp.exe">
     <tier name="Standalone Tier" />
   </standalone-application>
   <standalone-application
executable="MyOtherStandaloneApp.exe">
     <tier name="Standalone Tier" />
   </standalone-application>
</standalone-applications>
```

Set the Standlone Application element executable attribute to the executable file name. Do not include the full path to the executable.

For example, if the full path to the standalone application is C:\Program Files\MyApplication\MyStandaloneApp.exe, set the executable element to "MyStandaloneApp.exe". The file extension is optional, so "MyStandaloneApp" also works.

This sample config.xml demonstrates instrumentation for a 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="MyStandaloneApp.exe">
        <tier name="Standalone Tier" />
      </standalone-application>
    </standalone-applications>
  </app-agents>
</appdynamics-agent>
```

3. Restart the AppDynamics.Agent.Coordinator.
4. Restart the standalone application.

> If your standalone application does not implement any of the frameworks we instrument by default, you must configure a POCO entry point for a class/method in your application for the agent to begin instrumentation. See Supported Environments and Versions for .NET and POCO Entry Points.

Learn More

- Mapping Application Services to the AppDynamics Model
- Configure the App Agent for .NET
- Install the App Agent for .NET

AppDynamics for Windows Azure with NuGet

- Prerequisites
- Register for an AppDynamics for Windows Azure Account
- Add the App Agent for .NET to Your Azure Solution
- Learn More

Monitor your Azure cloud solutions with the AppDynamics for Windows Azure NuGet package. Sign up for the AppDynamics add-on in the Windows Azure portal, then enable the AppDynamics agent in your Visual Studio solution. If you can't use NuGet to install the App Agent for .NET, see Manually Install the App Agent for .NET on Windows Azure.

Prerequisites

- Visual Studio 2010 or later
- A Visual Studio solution to monitor
  Visual Studio must have the following permissions to the solution:
  - **Read** and **Write** permissions to each **project directory**
  - **Read** and **Write** permissions to each **Visual Studio .NET C# Project** (*.csproj) file
  - **Read** and **Write** permissions to the **Service Definition** (ServiceDefinition.csdef) file
- Windows Azure SDK
- Windows Azure account
- If the Visual Studio Standalone Profiler is installed, you must uninstall it and remove related registry entries.

Register for an AppDynamics for Windows Azure Account

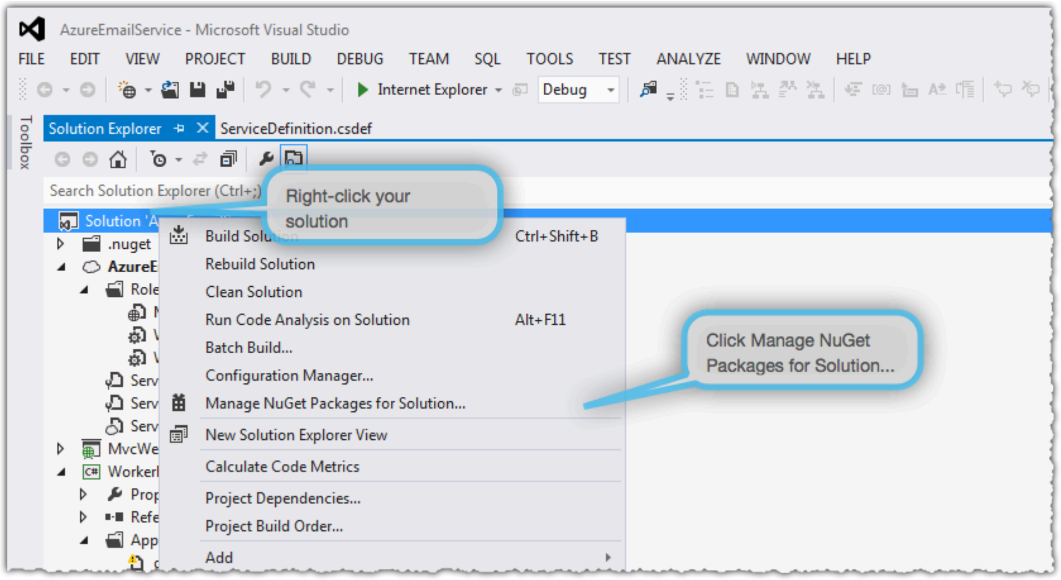
If you haven't registered for an AppDynamics account, see Register for AppDynamics for Windows Azure.

Once you've registered, we will send you a Welcome email with your URL and credentials to connect to the AppDynamics Controller. If you already have AppDynamics credentials from another product, you can sign in using them.

Add the App Agent for .NET to Your Azure Solution

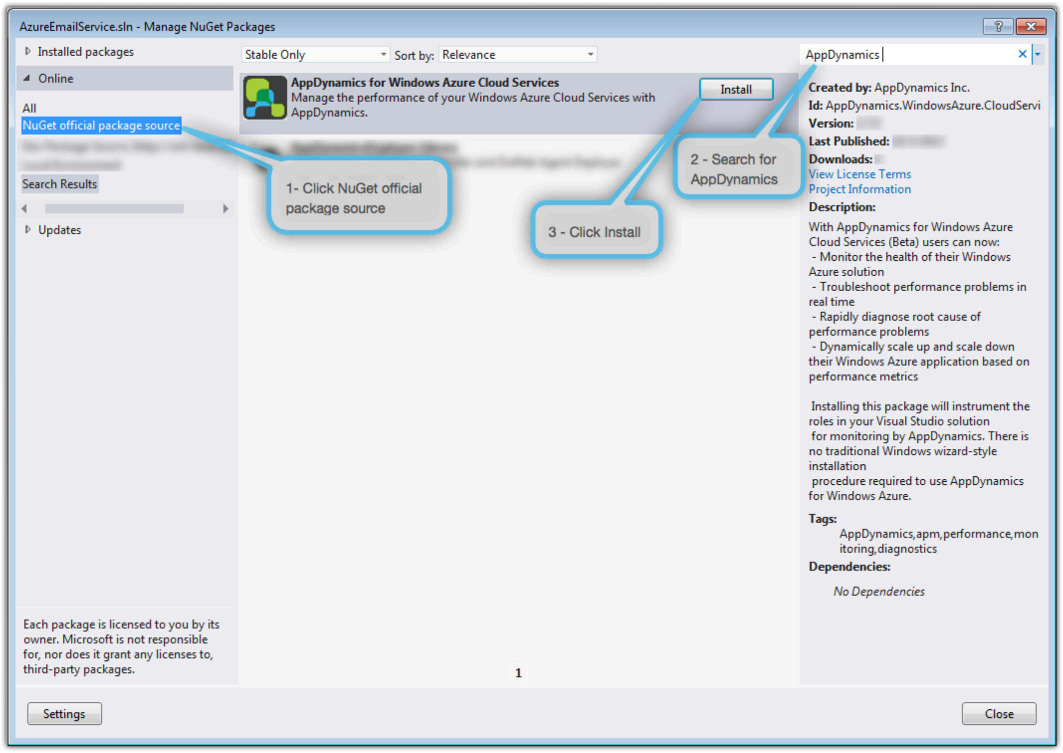
Use the NuGet package manager to add the App Agent for .NET to your Azure solution in Visual Studio.

1. In Visual Studio, right-click your solution name and chose **Manage NuGet Packages for Solution**.


AzureEmailService - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Internet Explorer
Debug
Solution Explorer
ServiceDefinition.csdef
Search Solution Explorer (Ctrl+;)
Right-click your solution
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Run Code Analysis on Solution Alt+F11
Batch Build...
Configuration Manager...
Manage NuGet Packages for Solution...
New Solution Explorer View
Calculate Code Metrics
Project Dependencies...
Project Build Order...
Add
Click Manage NuGet Packages for Solution...

2. Click the **NuGet official package source** in the left menu and type "AppDynamics" in the search bar in the upper right.

3. Select **AppDynamics .NET Agent for Azure** and click **Install**.


AzureEmailService.sln - Manage NuGet Packages
Installed packages
Online
All
NuGet official package source
Search Results
Updates
Stable Only
Sort by: Relevance
AppDynamics
Install
1- Click NuGet official package source
2 - Search for AppDynamics
3 - Click Install
Settings
Close
1

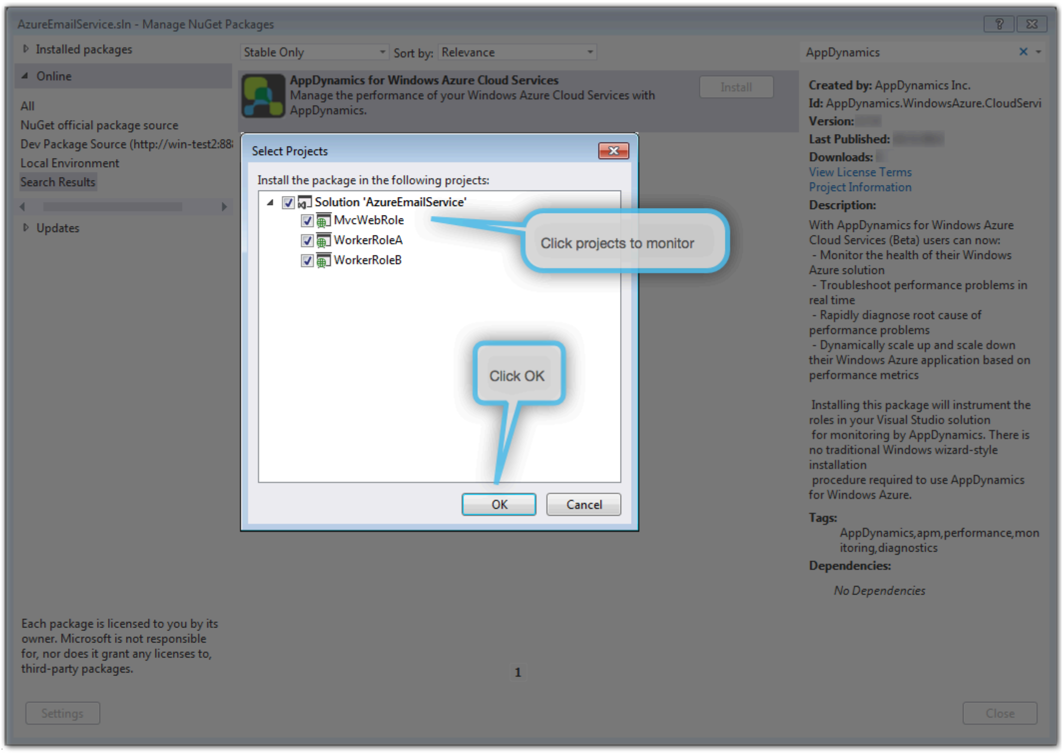
AppDynamics for Windows Azure Cloud Services
Manage the performance of your Windows Azure Cloud Services with AppDynamics.

Created by: AppDynamics Inc.
Id: AppDynamics.WindowsAzure.CloudServi
Version:
Last Published:
Downloads:
View License Terms
Project Information
Description:
With AppDynamics for Windows Azure Cloud Services (Beta) users can now:
- Monitor the health of their Windows Azure solution
- Troubleshoot performance problems in real time
- Rapidly diagnose root cause of performance problems
- Dynamically scale up and scale down their Windows Azure application based on performance metrics

Installing this package will instrument the roles in your Visual Studio solution for monitoring by AppDynamics. There is no traditional Windows wizard-style installation procedure required to use AppDynamics for Windows Azure.
Tags: AppDynamics,apm,performance,monitoring,diagnostics
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

4. Click the projects you want to monitor, then click **OK**.


AzureEmailService.sln - Manage NuGet Packages
Installed packages
Online
All
NuGet official package source
Dev Package Source (http://win-test2:88
Local Environment
Search Results
Updates
Stable Only
Sort by: Relevance
AppDynamics for Windows Azure Cloud Services
Manage the performance of your Windows Azure Cloud Services with AppDynamics.
Install
Select Projects
Install the package in the following projects:
Solution 'AzureEmailService'
MvcWebRole
WorkerRoleA
WorkerRoleB
Click projects to monitor
Click OK
OK
Cancel
Settings
Close

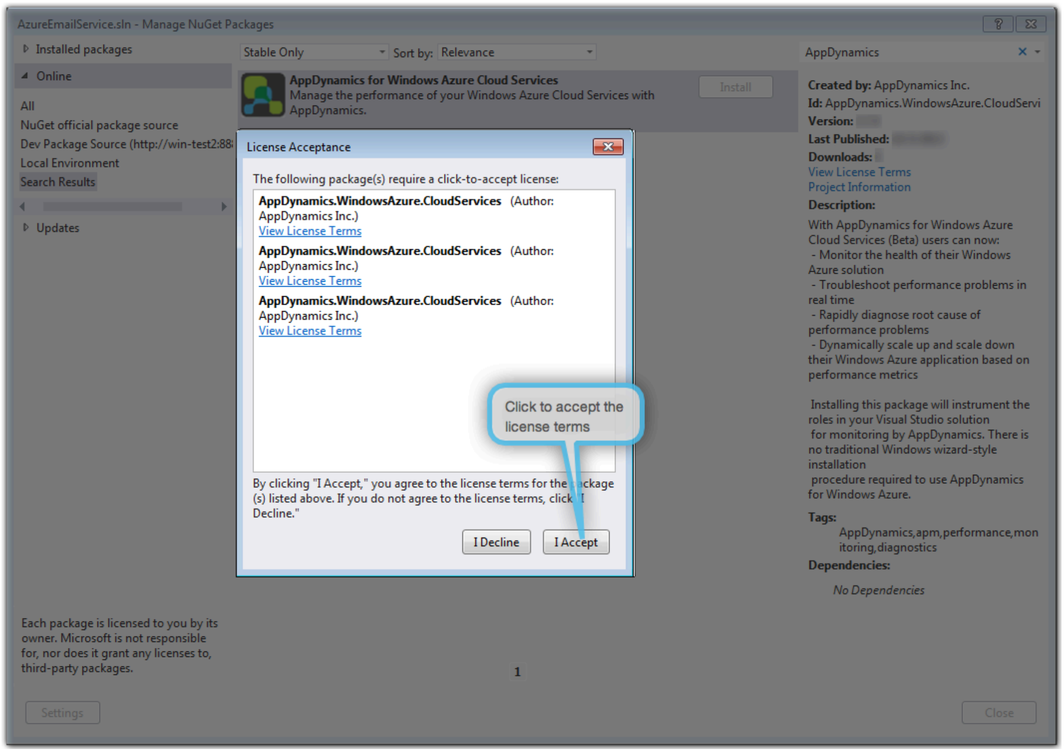
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

AppDynamics

Created by: AppDynamics Inc.
Id: AppDynamics.WindowsAzure.CloudServi
Version: [redacted]
Last Published: [redacted]
Downloads: [redacted]
View License Terms
Project Information
Description:
With AppDynamics for Windows Azure Cloud Services (Beta) users can now:
- Monitor the health of their Windows Azure solution
- Troubleshoot performance problems in real time
- Rapidly diagnose root cause of performance problems
- Dynamically scale up and scale down their Windows Azure application based on performance metrics

Installing this package will instrument the roles in your Visual Studio solution for monitoring by AppDynamics. There is no traditional Windows wizard-style installation procedure required to use AppDynamics for Windows Azure.
Tags: AppDynamics,apm,performance,monitoring,diagnostics
Dependencies:
No Dependencies

5. Click **I Accept** to agree to the license terms.


AzureEmailService.sln - Manage NuGet Packages
Installed packages
Online
All
NuGet official package source
Dev Package Source (http://win-test2:88
Local Environment
Search Results
Updates
Stable Only
Sort by: Relevance
AppDynamics for Windows Azure Cloud Services
Manage the performance of your Windows Azure Cloud Services with AppDynamics.
Install
License Acceptance
Click to accept the license terms
I Decline
I Accept
Settings
Close

The following package(s) require a click-to-accept license:

AppDynamics.WindowsAzure.CloudServices (Author: AppDynamics Inc.)
View License Terms
AppDynamics.WindowsAzure.CloudServices (Author: AppDynamics Inc.)
View License Terms
AppDynamics.WindowsAzure.CloudServices (Author: AppDynamics Inc.)
View License Terms

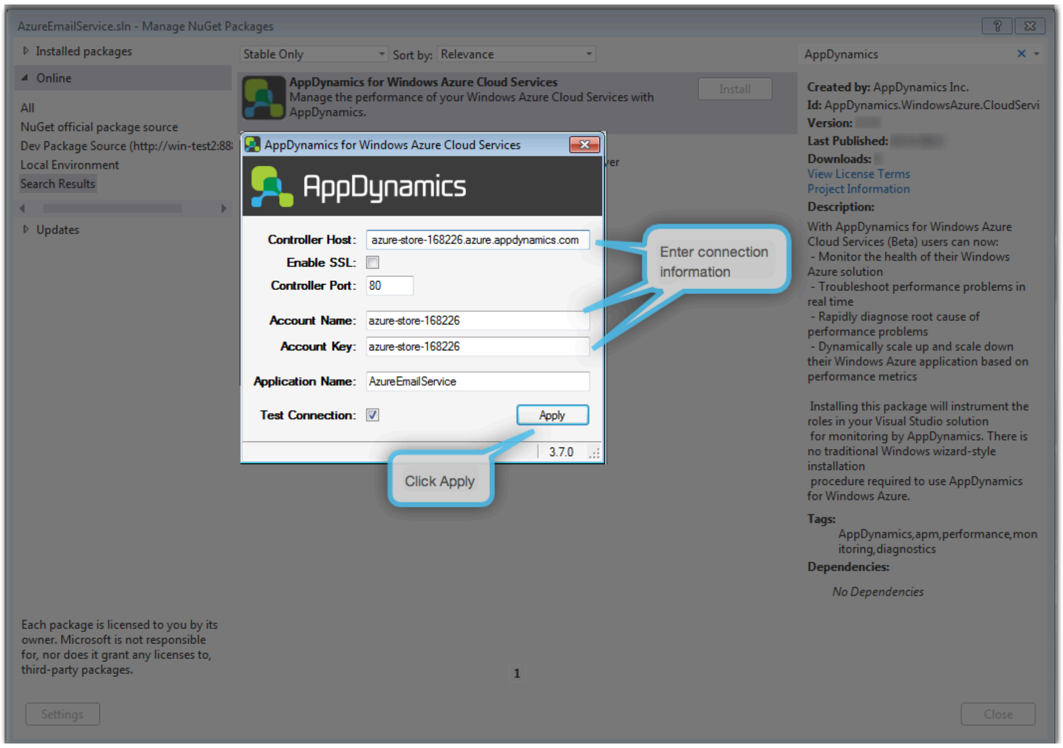
By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

6. Enter the connection information from your Welcome email:

- Controller Host
- Controller Port
- Account Name
- Account Key

We automatically populate **Application Name** with the name of your Windows Azure solution.


AzureEmailService.sln - Manage NuGet Packages
Installed packages
Online
All
NuGet official package source
Dev Package Source (http://win-test2:88
Local Environment
Search Results
Updates
Stable Only
Sort by: Relevance
Install
AppDynamics
AppDynamics for Windows Azure Cloud Services
AppDynamics
Apply
3.7.0
Enter connection information
Click Apply
Settings
Close
1

AppDynamics for Windows Azure Cloud Services
Manage the performance of your Windows Azure Cloud Services with AppDynamics.

Controller Host: azure-store-168226.azure.appdynamics.com
Enable SSL: ☐
Controller Port: 80
Account Name: azure-store-168226
Account Key: azure-store-168226
Application Name: AzureEmailService
Test Connection: ☑

Created by: AppDynamics Inc.
Id: AppDynamics.WindowsAzure.CloudServi
Version: [redacted]
Last Published: [redacted]
Downloads: [redacted]
View License Terms
Project Information
Description:
With AppDynamics for Windows Azure Cloud Services (Beta) users can now:
- Monitor the health of their Windows Azure solution
- Troubleshoot performance problems in real time
- Rapidly diagnose root cause of performance problems
- Dynamically scale up and scale down their Windows Azure application based on performance metrics

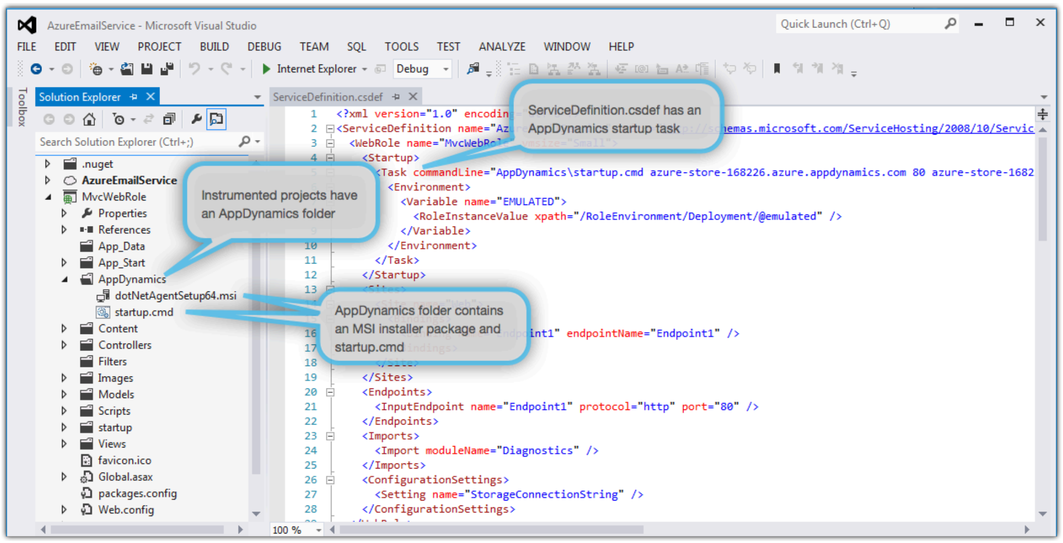
Installing this package will instrument the roles in your Visual Studio solution for monitoring by AppDynamics. There is no traditional Windows wizard-style installation procedure required to use AppDynamics for Windows Azure.
Tags: AppDynamics,apm,performance,monitoring,diagnostics
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

7. **Close** the Manage NuGet Packages window.

8. Verify the following:

- Instrumented projects have an AppDynamics folder
- The AppDynamics folder(s) contain an MSI installer package and a setup.cmd file
- The ServiceDefinition.csdef has an AppDynamics startup task

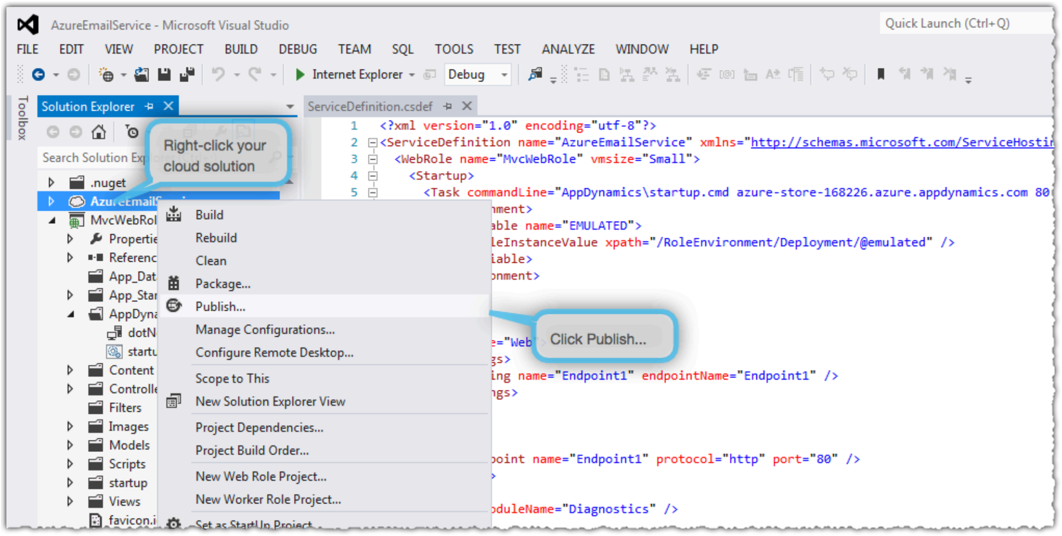


```
<?xml version="1.0" encodin
<ServiceDefinition name="Az                                   hemas.microsoft.com/ServiceHosting/2008/10/Servic
  <WebRole name="MvcWebRo
    <Startup>
      <Task commandLine="AppDynamics\startup.cmd azure-store-168226.azure.appdynamics.com 80 azure-store-1682
        <Environment>
          <Variable name="EMULATED">
            <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
          </Variable>
        </Environment>
      </Task>
    </Startup>

                    int1" endpointName="Endpoint1" />

    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <ConfigurationSettings>
      <Setting name="StorageConnectionString" />
    </ConfigurationSettings>
```

9. Right-click the Windows Azure cloud solution and click **Publish**.



```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="AzureEmailService" xmlns="http://schemas.microsoft.com/ServiceHosti
  <WebRole name="MvcWebRole" vmsize="Small">
    <Startup>
      <Task commandLine="AppDynamics\startup.cmd azure-store-168226.azure.appdynamics.com 80
```

Once you've successfully published your project, you're ready to log on to the Controller and begin monitoring your solution!

Learn More

- AppDynamics Essentials
- AppDynamics for .NET
- Quick Tour of the User Interface Video Tutorial

Manually Install the App Agent for .NET on Windows Azure

- Prerequisites
- Register for an AppDynamics Account

This topic describes how to manually install the App Agent for .NET on Windows Azure. To install the App Agent for .NET on Windows Azure using NuGet, see AppDynamics for Windows Azure with NuGet.

Prerequisites

- Visual Studio 2010 or later
- A Visual Studio solution to monitor
- Windows Azure SDK
- Windows Azure account
- If the Visual Studio Standalone Profile is installed, you must uninstall it and clean up the registry.

Register for an AppDynamics Account

If you haven't registered for an AppDynamics account, see Register for AppDynamics for Windows Azure.

Once you've registered, we will send you a Welcome email with your URL and credentials to connect to the AppDynamics Controller. If you already have AppDynamics credentials from another product, you can sign in using them.

AppDynamics Account Home Page

Your AppDynamics account home page includes:

- A link to the AppDynamics download site where you can download the App Agent for .NET
- Controller URL where you log into your account on the AppDynamics controller hosted service
- AppDynamics credentials: Account, Username, and Access Key
- Number of days left in your Pro trial
- Links to the AppDynamics on-boarding videos and documentation

You can access your AppDynamics account home page at any time by entering its URL in a web browser and signing in with your AppDynamics credentials.

Download the App Agent for .NET

To download the App Agent for .NET from AppDynamics:

1. Navigate to the AppDynamics download site. The URL is in your welcome email and on your AppDynamics account home page.

2. Log in with your AppDynamics account name and access key.

3. Download the file named **dotNetAgentSetup64.msi**. Do not run the file.

4. Download the file named **startup.cmd**.

For convenience we've attached the same file to this document (startup.cmd) and included the contents in step 5 below.

Add the App Agent for .NET to Windows Azure Roles

This step instruments the roles in your Visual Studio solution for monitoring by AppDynamics. There is no traditional Windows wizard-style installation procedure required to use AppDynamics for Windows Azure.

These instructions apply to the AppDynamics Agent for .NET version 3.7.8 or later. If you downloaded an earlier version, please download the latest version of the agent.

1. Either create a new Windows Azure cloud project in Visual Studio or open an existing Windows Azure cloud project.

2. If you created a new project, add the Web role and/or Worker role projects to the solution.

3. To each Web and Worker role project that you want to monitor, add a new folder named **AppDynamics**.
The scripts look for the AppDynamics folder. Using a different name may cause the script to fail.

4. Copy the dotNetAgentSetup64.msi you downloaded to the AppDynamics folder for each Web and Worker role project.

Note that while each *role project* has a single attached agent MSI installer package, each *role instance* in the project requires a separate agent license.

5. Download the startup.cmd and add it to each Web and Worker role project that you want to monitor.

Alternatively, add a text file named startup.cmd and paste the following lines in it:

```
REM Install the AppDynamics agent on Windows Azure
SETLOCAL EnableExtensions

IF {%INTERNAL_APPDYNAMICS_AGENT_INSTALL_REBOOT%}=={true} (
 SETX INTERNAL_APPDYNAMICS_AGENT_INSTALL_REBOOT ""
 GOTO :END
)

REM Bypass the installation if this is emulated environment
IF {%EMULATED%}=={true} GOTO :END

REM Uninstall if .NET Agent already installed
IF {%COR_PROFILER%}=={AppDynamics.AgentProfiler} (
 start /wait msiexec /x {0C633F51-09FE-4AE4-A25F-F6CD167CC46E} /quiet /log
d:\aduninstall.log
 IF NOT %ERRORLEVEL%==0 SHUTDOWN /r /f /c "Reboot after uninstalling the
AppDynamics .NET Agent"
)
ELSE IF DEFINED COR_PROFILER GOTO :END

IF NOT EXIST AppDynamics\dotNetAgentSetup64.msi GOTO :END
```

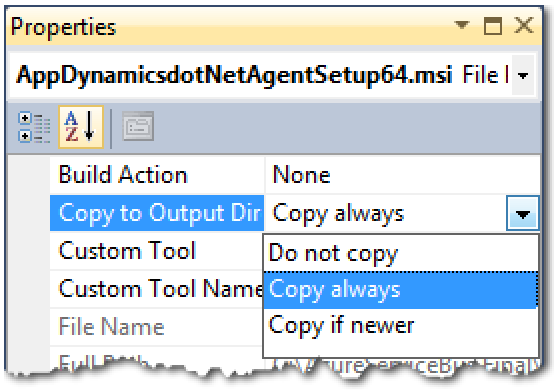
```
SET ControllerHost=%1
SET ControllerPort=%2
SET AccountName=%3
SET AccountAccessKey=%4
SET ControllerApplication=%5
SET ControllerSSLEnabled=%6

IF {%1}=={} (ECHO Syntax error: Missing AD_Agent_ControllerHost parameter
>>d:\adInstall.log) & (GOTO :END)
IF {%2}=={} (ECHO Syntax error: Missing AD_Agent_ControllerPort parameter
>>d:\adInstall.log) & (GOTO :END)
IF {%3}=={} (ECHO Syntax error: Missing AD_Agent_AccountName parameter
>>d:\adInstall.log) & (GOTO :END)
IF {%4}=={} (ECHO Syntax error: Missing AD_Agent_AccountAccessKey parameter
>>d:\adInstall.log) & (GOTO :END)
IF {%5}=={} (SET ControllerApplication=Default)
IF {%6}=={} (SET ControllerSSLEnabled=false)

REM Install the agent
start /wait msiexec /i AppDynamics\dotNetAgentSetup64.msi
AD_Agent_Environment=Azure AD_AzureRoleName=%RoleName%
AD_AzureRoleInstanceID=%RoleInstanceID% AD_Agent_ControllerHost=%ControllerHost%
AD_Agent_ControllerPort=%ControllerPort% AD_Agent_AccountName=%AccountName%
AD_Agent_AccountAccessKey=%AccountAccessKey%
AD_Agent_ControllerApplication=%ControllerApplication%
AD_Agent_ControllerSSLEnabled=%ControllerSSLEnabled% /quiet /log
d:\adInstall.log
IF %ERRORLEVEL%==0 (
 SETX INTERNAL_APPDYNAMICS_AGENT_INSTALL_REBOOT "true"
 REM Reboot the machine after installation in order to restart role CLR and
attach AppDynamics Agent to it
 SHUTDOWN /r /f /c "Reboot after installing the AppDynamics .NET Agent"
)
```

```
:END
EXIT /B %ERRORLEVEL%
```

5. **IMPORTANT:** For each Web and Worker role to that you want to monitor, set the **Copy to Output Directory** property for the dotNetAgentSetup64.msi file and for the startup.cmd file to **Cop y Always**.


Properties
AppDynamicsdotNetAgentSetup64.msi File I

Build Action	None
Copy to Output Dir	Copy always
Custom Tool	Do not copy
Custom Tool Name	Copy always
File Name	Copy if newer

6. In the **ServiceDefinition.csdef** file for the Windows Azure cloud project, add a Startup Task element that invokes startup.cmd with parameters for each WorkerRole and WebRole element.

Add the following lines:

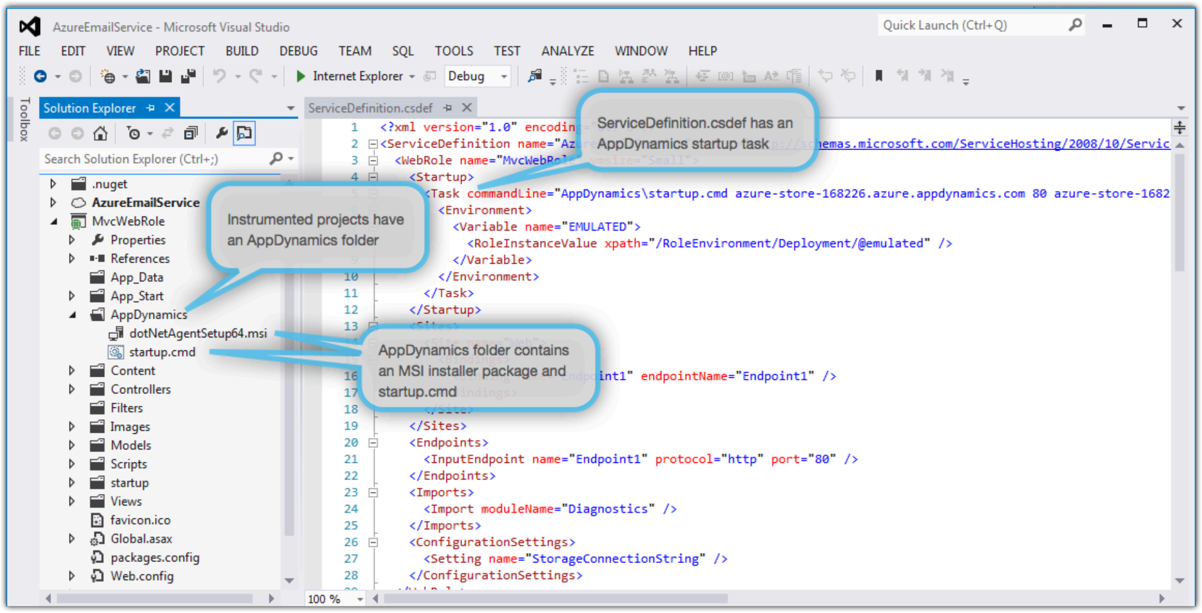
```
<Startup>
      <Task commandLine="AppDynamics\startup.cmd [your_controller_host]
[your_controller_port] [your_account_name] [your_access_key]
[your_application_name] [SSLEnabled]" executionContext="elevated"
taskType="simple">
        <Environment>
          <Variable name="EMULATED">
            <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
          </Variable>
        </Environment>
      </Task>
    </Startup>
```

where:

- **your_controller_host** and **your_controller_port** are the Controller host and port for your AppDynamics account, and **your_account_name** and **your_access_key** are your credentials. The Welcome email we sent when you registered with AppDynamics includes this information. You can also find it on your AppDynamics account home page. See Registe r for AppDynamics for Windows Azure.
- **your_application_name** is the name you choose for your business application. This name identifies the application in the AppDynamics Controller interface.
- **SSLEnabled** is an optional property that defaults to "false". To enable SSL set **your_contro ller_port** to 443 and **SSLEnabled** to "true".

7. Verify the following:

- Instrumented projects have an AppDynamics folder
- The AppDynamics folder(s) contain a dotNetAgentSetup64.msi and a startup.cmd file
- For instrumented projects, you set the **Copy to Output Directory** property for the .msi and .cmd files to **Copy Always**
- The ServiceDefinition.csdef includes an AppDynamics startup task


ServiceDefinition.csdef has an AppDynamics startup task
Instrumented projects have an AppDynamics folder
AppDynamics folder contains an MSI installer package and startup.cmd

```
<?xml version="1.0" encodin
<ServiceDefinition name="Az                                      hemas.microsoft.com/ServiceHosting/2008/10/Servic
  <WebRole name="MvcWebRo
    <Startup>
      <Task commandLine="AppDynamics\startup.cmd azure-store-168226.azure.appdynamics.com 80 azure-store-1682
        <Environment>
          <Variable name="EMULATED">
            <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
          </Variable>
        </Environment>
      </Task>
    </Startup>

                                int1" endpointName="Endpoint1" />


    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
    <ConfigurationSettings>
      <Setting name="StorageConnectionString" />
    </ConfigurationSettings>
```

Publish the AppDynamics-Instrumented Application to Windows Azure

1. In Visual Studio, click the Windows Azure cloud project.

2. Right-click. Click Publish...

Monitor Your Application

Once you've published your application and there is load on it, you can login to the Controller and begin monitoring.

1. Log into the AppDynamics Controller at the URL given in your welcome email and on your AppDynamics account home page.

2. Send some requests to your application so there is some traffic to monitor and wait a few minutes.

3. In the AppDynamics Controller, click your application.

4. Monitor your application.

Learn More

- Features Overview
- Quick Tour of the User Interface Video Tutorial

Register for AppDynamics for Windows Azure

- What is AppDynamics?

Monitor your Azure cloud solutions with the AppDynamics for Windows Azure NuGet package. Sign up for the AppDyanmics add-on in the Windows Azure portal, then enable the AppDynamics agent in your Visual Studio solution.

What is AppDynamics?

AppDynamics is an application performance monitoring solution that helps you:

- Identify problems, such as slow and stalled user requests and errors, in a production environment
- Troubleshoot and isolate the root cause of such problems

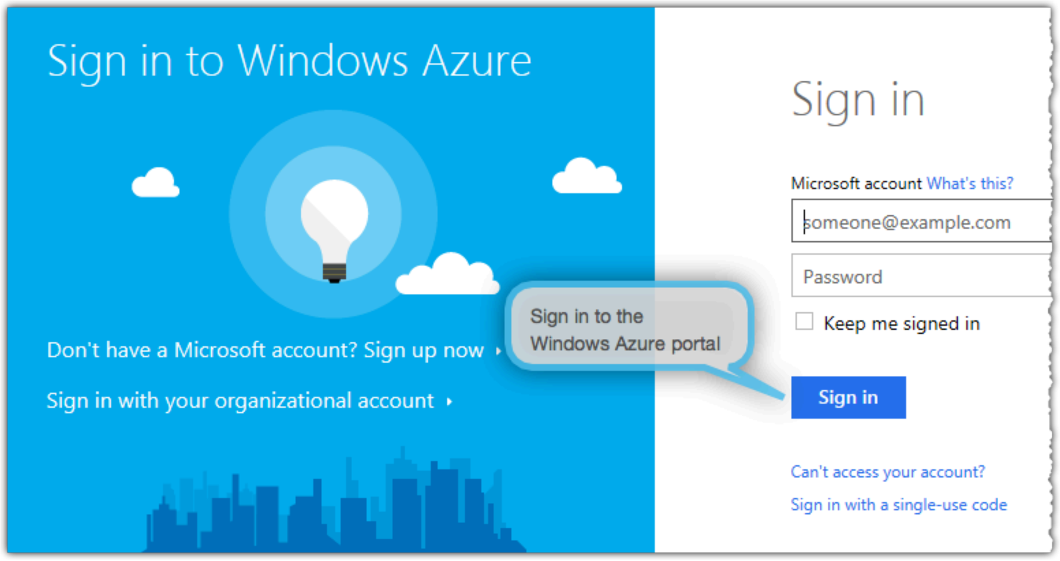
There are two components in AppDynamics:

App Agent: The App Agent for .NET collects data from your servers. You run a separate agent on every role instance that you want to monitor. You install the agent as part of the NuGet package.

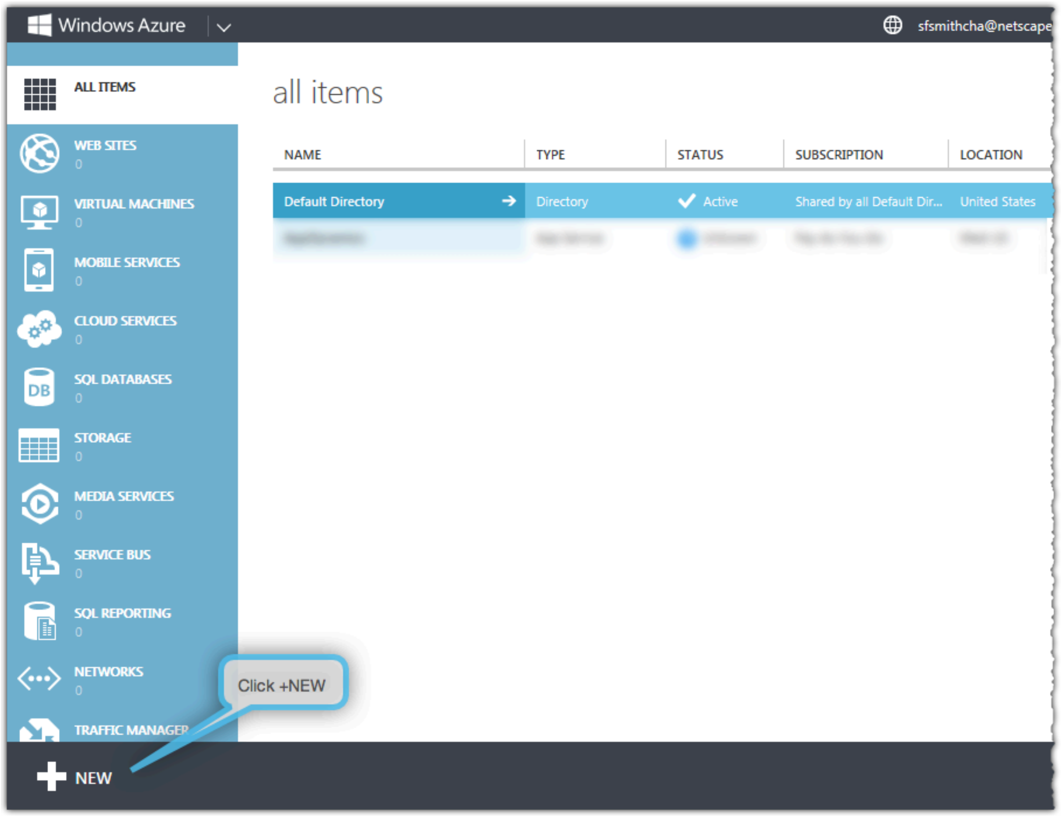
AppDynamics Controller: The agent sends information to an AppDynamics Controller hosted service. Using a browser-based console, you log on to the Controller to monitor, analyze and troubleshoot your application.

Register for an AppDynamics for Windows Azure Account

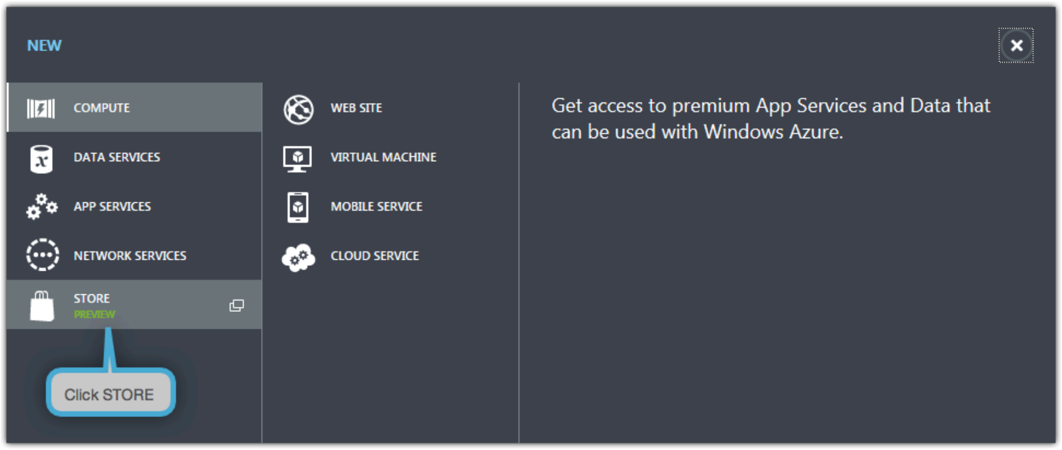
Use the Windows Azure portal to sign up for AppDynamics for Windows Azure.
1. Log on to the Windows Azure portal at https://manage.windowsazure.com.


Sign in to Windows Azure
Don't have a Microsoft account? Sign up now
Sign in with your organizational account
Sign in to the Windows Azure portal
Sign in
Microsoft account What's this?
someone@example.com
Password
Keep me signed in
Sign in
Can't access your account?
Sign in with a single-use code

2. Click **+NEW** at the bottom left corner of the portal.

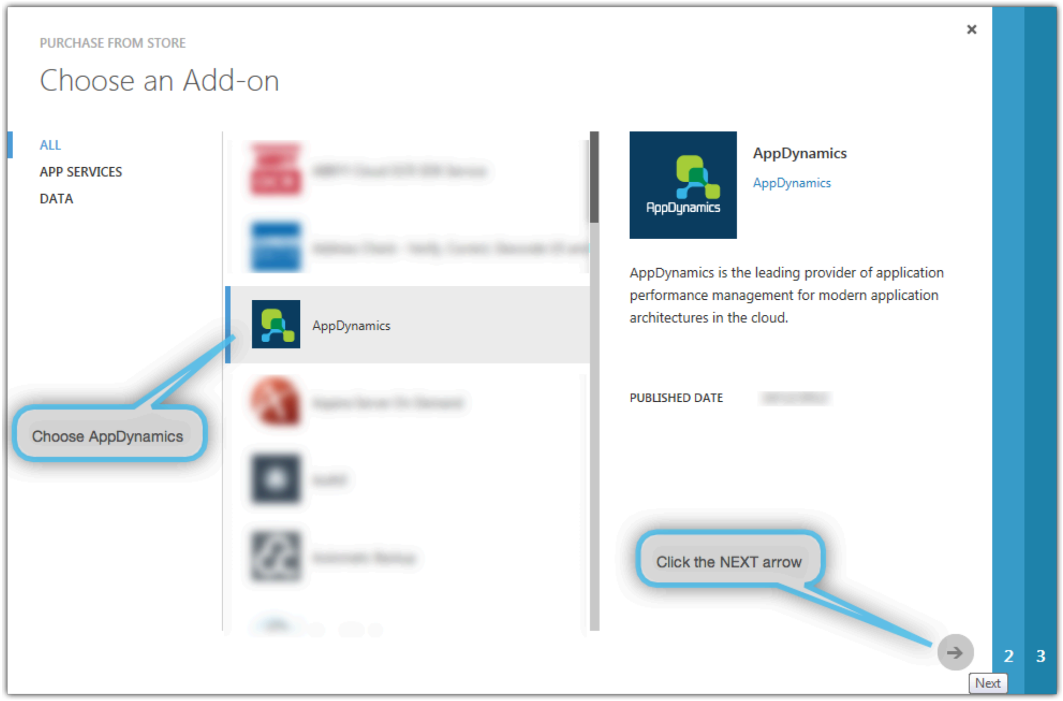


3. In the left menu, click **STORE**.



Get access to premium App Services and Data that can be used with Windows Azure.

4. Click **AppDynamics** in the list of services and click the **NEXT** arrow.


PURCHASE FROM STORE

Choose an Add-on

AppDynamics

AppDynamics

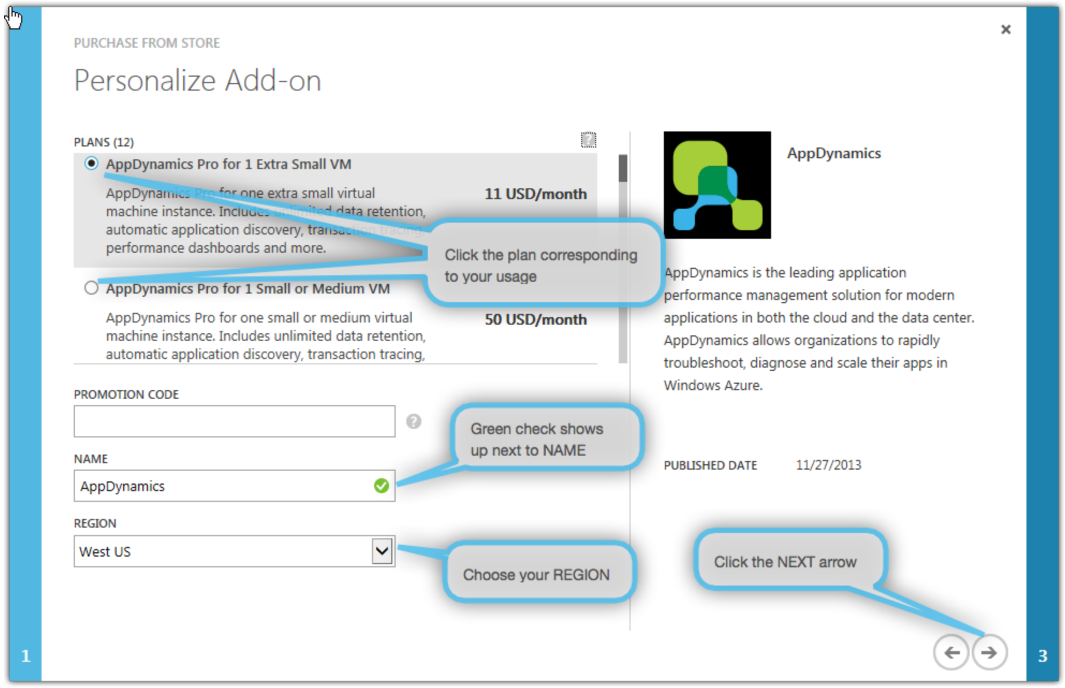
AppDynamics is the leading provider of application performance management for modern application architectures in the cloud.

PUBLISHED DATE

5. Click the plan corresponding to the size (Extra Small, Small, Medium, etc.) and number of virtual machines where you will install AppDynamics monitoring agents.

6. Select your **REGION** and click the **NEXT** arrow.

7. After the form validates the **NAME** field and displays a green check, click the **NEXT** arrow.


PURCHASE FROM STORE

Personalize Add-on

PLANS (12)

● AppDynamics Pro for 1 Extra Small VM — 11 USD/month

AppDynamics Pro for one extra small virtual machine instance. Includes unlimited data retention, automatic application discovery, transaction tracing, performance dashboards and more.

○ AppDynamics Pro for 1 Small or Medium VM — 50 USD/month

AppDynamics Pro for one small or medium virtual machine instance. Includes unlimited data retention, automatic application discovery, transaction tracing,

PROMOTION CODE

NAME: AppDynamics ✓

REGION: West US

AppDynamics

AppDynamics is the leading application performance management solution for modern applications in both the cloud and the data center. AppDynamics allows organizations to rapidly troubleshoot, diagnose and scale their apps in Windows Azure.

PUBLISHED DATE 11/27/2013

Click the plan corresponding to your usage
Green check shows up next to NAME
Choose your REGION
Click the NEXT arrow
1
3

8. Review your purchase and click the **CHECK** to purchase.

You've successfully signed up for AppDynamics for Windows Azure!

Register for an AppDynamics for Windows Azure Account From Windows Azure Marketplace

1. Click **Try Free** or **Sign Up** for AppDynamics on the Windows Azure Marketplace at https://datamarket.azure.com/application/f9949031-b8b9-4da5-b500-c615f3f2a7cd.

If you choose **Sign Up**, you receive a free version of AppDynamics Pro for Windows Azure with full functionality, which downgrades after 30 days to a free version of AppDynamics Lite for Windows Azure with limited functionality. You do not need to provide a credit card for this option. You can upgrade to AppDynamics Pro for Windows Azure at any time.

If you choose **Try Free**, you receive a free version of AppDynamics Pro for Windows Azure with full functionality. You need to provide a credit card for this option. After 30 days your credit account will be charged for continued use of AppDynamics Pro for Windows Azure, unless you cancel your subscription.

You need one agent license for each role instance that you wish to monitor. For example, a site running 2 Web role instances and 2 Worker role instances requires 4 agent licenses.

2. On the registration page, provide your user information, a password, email address, company name, and the name of the application you are monitoring as you will publish it with Windows Azure.

3. Click **Register Now**.

Next Steps

We'll send you a Welcome email with your URL and credentials to connect to the AppDynamics Controller. Then you can use NuGet to add the App Agent for .NET to your Windows Azure solution or install it manually.

- To install the agent using NuGet, see AppDynamics for Windows Azure with NuGet.
- To install the agent manually, see Manually Install the App Agent for .NET on Windows Azure.

Learn More

- AppDynamics for Windows Azure with NuGet
- Manually Install the App Agent for .NET on Windows Azure
- AppDynamics Essentials
- AppDynamics for .NET

Uninstall the App Agent for .NET

This topic describes how to do a complete uninstall of the App Agent for .NET.

⚠ Do not follow these instructions if you are doing an upgrade. If you want to upgrade to a new version see Upgrade the App Agent for .NET.

To completely uninstall the App Agent for .NET

1. Stop IIS, instrumented Windows services, and instrumented standalone applications.
   ⚠ Failing to stop instrumented applications before uninstalling the agent requires you to reboot

the machine to complete the uninstall.

2. Stop the **AppDynamics.Agent.Coordinator** service.

3. In the Control Panel, select Add/Remove Programs. Remove the **AppDynamics .NET Agent**.

4. The uninstall procedure does not remove configuration files. Delete the configuration directory:
   Windows Server 2008 and later: `%ProgramData%\AppDynamics`
   Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics`

5. Verify the uninstall deleted the AppDynamics executables directory. If not, delete it manually:
   Windows Server 2003 and later: %ProgramFiles%\AppDynamics

6. Restart IIS, Windows services, and standalone applications.

Learn More

- Upgrade the App Agent for .NET
- Install the App Agent for .NET
- Agent - Controller Compatibility Matrix
- Release Notes for AppDynamics Pro

Resolve App Agent for .NET Installation and Configuration Issues

This topic covers how to solve installation and configuration problems for the App Agent for .NET.

Resolve Agent-Controller Configuration Issues

To verify that the App Agent for .NET is reporting to the Controller

Use the AppDynamics UI to verify that the agent is able to connect to the Controller.

1. In a browser open:

```
http://<controller-host>:<controller-port>/controller
```

If you can't connect to the controller in Internet Explorer, see To check for misconfigured IE proxy settings.

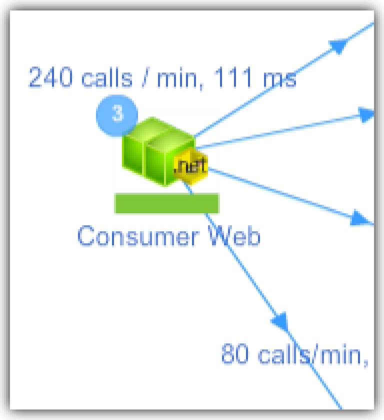
2. Log in to the AppDynamics UI.

3. Select the application to open the Application Dashboard.

4. In the left navigation panel click **Servers -> App Servers** and open the Health tab.

The Health tab lists the tiers, their nodes, and App Agent Status. When an agent successfully reports to the Controller, you see an "up" arrow symbol. For details see Verify App Agent-Controller Communication.

- When deploying multiple agents for the same tier, determine whether you get the correct number of nodes reporting into the same tier.
- After sending a request to your web application, data should appear on the AppDynamics UI. The agents should be displayed in the Application Flow Map of the Application Dashboard.



If no data appears after a few minutes:

- Verify that the Agent is writing its log files:
  Windows Server 2008 and later: `%ProgramData%\AppDynamics\DotNetAgent\Logs\AgentLog.txt`
  Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs\AgentLog.txt`

- If the log file exists, open it and review it for errors.
- If the log file doesn't exist, run the Windows Event Viewer and see the application messages.
- If there are no AppDynamics event messages, look for messages from the .NET Runtime.

To check Internet Explorer proxy settings

Misconfigured proxy settings in Internet Explorer may cause the App Agent for .NET to fail to connect to the controller. If **Test Controller connection** fails on the Controller Configuration window in the AppDynamics Agent Configuration utility, do the following:

1. Verify the Controller host and port settings are correct.

2. In Internet Explorer, open:

```
http://<controller-host>:<controller-port>/controller
```

3. If the connection also fails in Internet Explorer, check the proxy settings. See Change IE Proxy Settings.

4. Correct or remove any incorrect proxy settings.

Checklist for Resolving App Agent for .NET Installation Issues

	Item	Notes
✓	Run the installer as Administrator.	Verify Administrative privileges
✓	Verify that COM+ is enabled.	To verify COM+ Services are enabled
✓	Verify permissions for Agent directory.	To verify that the .NET Agent directory has the correct permissions based on the site’s application pool identity.
✓	Verify that the Agent is compatible with the Controller.	Agent - Controller Compatibility Matrix

✓	Verify the correct settings in the config.xml: **Windows Server 2008 and later**: `%ProgramData%\AppDynamics\DotNetAgent\Config\config.xml` **Windows Server 2003**: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Config\config.xml`	Update the config.xml file to include the App Agent for .NET Configuration Properties .

Resolve App Agent for .NET Issues

Resolving Agent Installation Issues

If the Agent installation is failing, check the following configurations in your environment:

Verify Administrative privileges

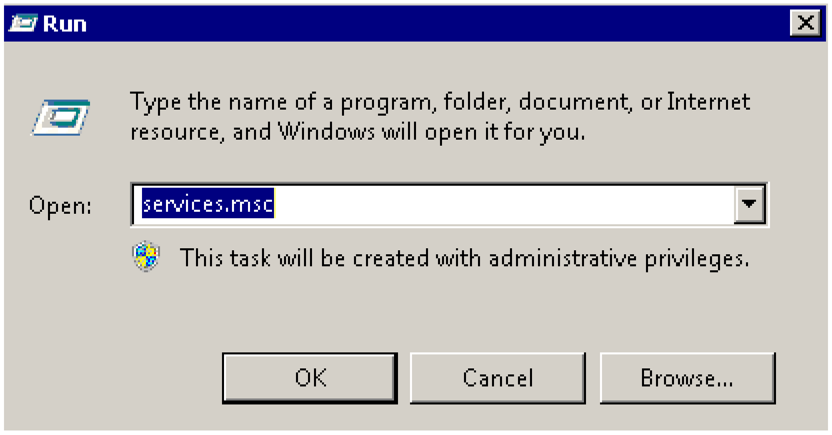
Ensure that you have the administrative privileges when you launch the installer. If the currently user doesn't have sufficient privileges, the installer prompts you for an administrator password.

To verify COM+ Services are enabled

If you receive an error about the Appdynamics.Agent.Coordinator_service, ensure that the COM+ applications in your system are enabled.

1. From the Windows Start Menu, click **Run**.

2. Enter "services.msc" and click on the "OK" button. Windows opens the Services Console.

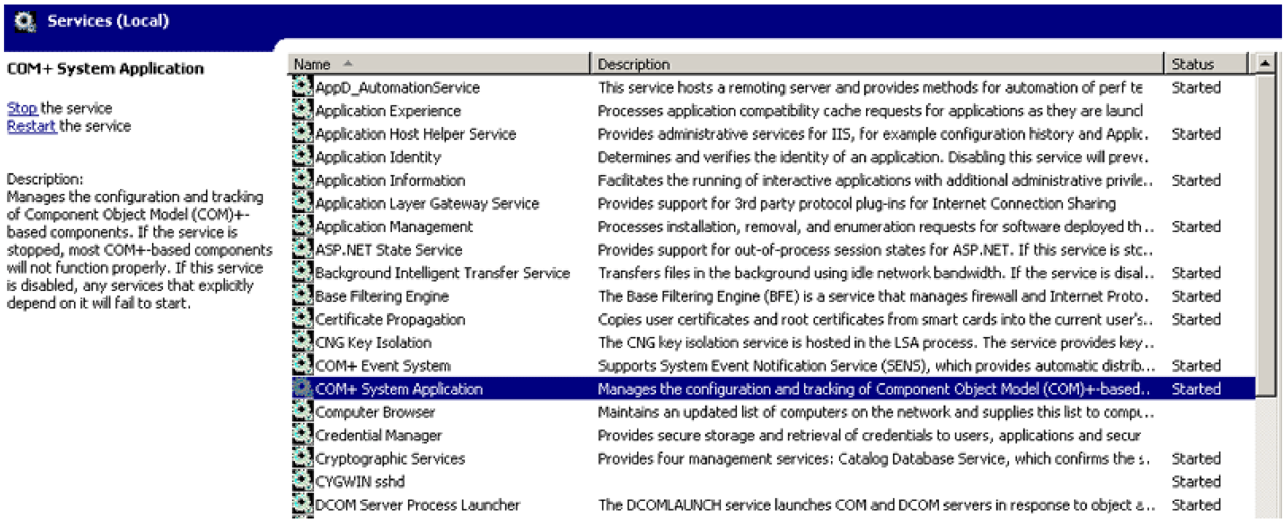

Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: services.msc

This task will be created with administrative privileges.

3. Enable the COM+ services (if not enabled).



Services (Local)

COM+ System Application

Stop the service
Restart the service

Description:
Manages the configuration and tracking of Component Object Model (COM)+-based components. If the service is stopped, most COM+-based components will not function properly. If this service is disabled, any services that explicitly depend on it will fail to start.

Name	Description	Status
AppD_AutomationService	This service hosts a remoting server and provides methods for automation of perf te	Started
Application Experience	Processes application compatibility cache requests for applications as they are launcl	
Application Host Helper Service	Provides administrative services for IIS, for example configuration history and Applic.	Started
Application Identity	Determines and verifies the identity of an application. Disabling this service will preve.	
Application Information	Facilitates the running of interactive applications with additional administrative privile..	Started
Application Layer Gateway Service	Provides support for 3rd party protocol plug-ins for Internet Connection Sharing	
Application Management	Processes installation, removal, and enumeration requests for software deployed th..	Started
ASP.NET State Service	Provides support for out-of-process session states for ASP.NET. If this service is stc..	
Background Intelligent Transfer Service	Transfers files in the background using idle network bandwidth. If the service is disal..	Started
Base Filtering Engine	The Base Filtering Engine (BFE) is a service that manages firewall and Internet Proto.	Started
Certificate Propagation	Copies user certificates and root certificates from smart cards into the current user's..	Started
CNG Key Isolation	The CNG key isolation service is hosted in the LSA process. The service provides key..	
COM+ Event System	Supports System Event Notification Service (SENS), which provides automatic distrib...	Started
COM+ System Application	Manages the configuration and tracking of Component Object Model (COM)+-based..	Started
Computer Browser	Maintains an updated list of computers on the network and supplies this list to compu..	
Credential Manager	Provides secure storage and retrieval of credentials to users, applications and secur	
Cryptographic Services	Provides four management services: Catalog Database Service, which confirms the s.	Started
CYGWIN sshd		Started
DCOM Server Process Launcher	The DCOMLAUNCH service launches COM and DCOM servers in response to object a..	Started

To generate a log for agent installation failures

If installer fails, use the command line utility to launch the installer.

```
msiexec /i $Path_to_the_MSI_File /l*v verbose.log
```

A verbose log for the .NET Agent is created at the same location where you saved the installer file. Send this log to the AppDynamics Support Team.

To correct failed installation caused by other APM products

The .NET Agent installation may fail if there are other Application Performance Management (APM) products installed in the same managed environment. Remove the associated "Environment" subkey for certain services for the installed APM products.

To remove associated "Environment" subkey for W2SVC and WAS services in the registry:

1. Run Regedit or regedt32.
2. In regedit.exe, locate the following registry keys:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\W3SVC
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\WAS
3. Expand the keys.
4. Modify the **Environment** subkey to delete the following values:

```
COMPLUS_ProfAPI_ProfilerCompatibilitySetting=EnableV2Profiler
    COR_ENABLE_PROFILING=1
    COR_PROFILER= {a GUID}
```

5. Restart the services. For more details see How to restart the W2SVC and WAS services?.

Resolve Configuration Errors

Verify the configuration in the config.xml file

- Ensure that you have correctly configured the config.xml file for the App Agent for .NET. For more detail, see App Agent for .NET Configuration Properties.
- If you made manual edits to the config.xml, check the AgentLog.txt and WarnLog.txt for errors. Invalid XML shows in the log as follows:

```
2014-03-13 10:49:18.7199 1232 dllhost 1 1 Error ConfigurationManager Error
reading the configuration file
```

Resolve Log Issues

The App Agent for .NET writes logs to the following directories:

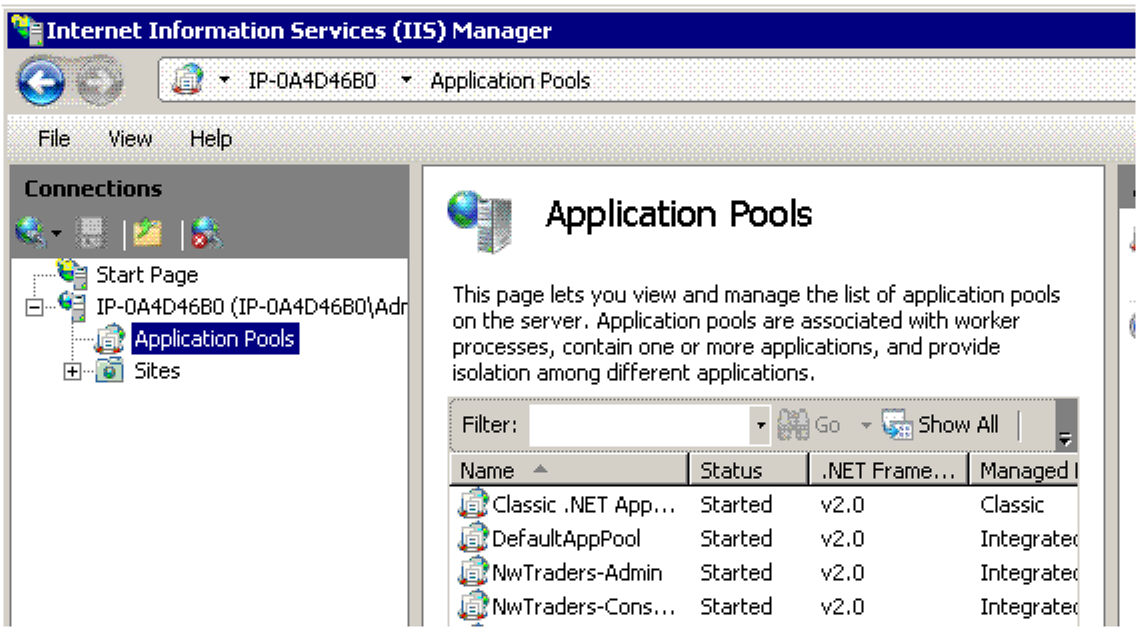
Windows Server 2008 and later: `%ProgramData%\AppDynamics\DotNetAgent\Logs`
Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs`

The agent will not generate logs if the agent directory does not have sufficient permissions permissions.

⚠ **IMPORTANT:** If the installation doesn't create the AppDynamics directory, contact AppDynamics Support Team.

To verify that the .NET Agent directory has the correct permissions

1. Click **IIS -> Application pools**.
IIS displays the list of application pools for your machine.

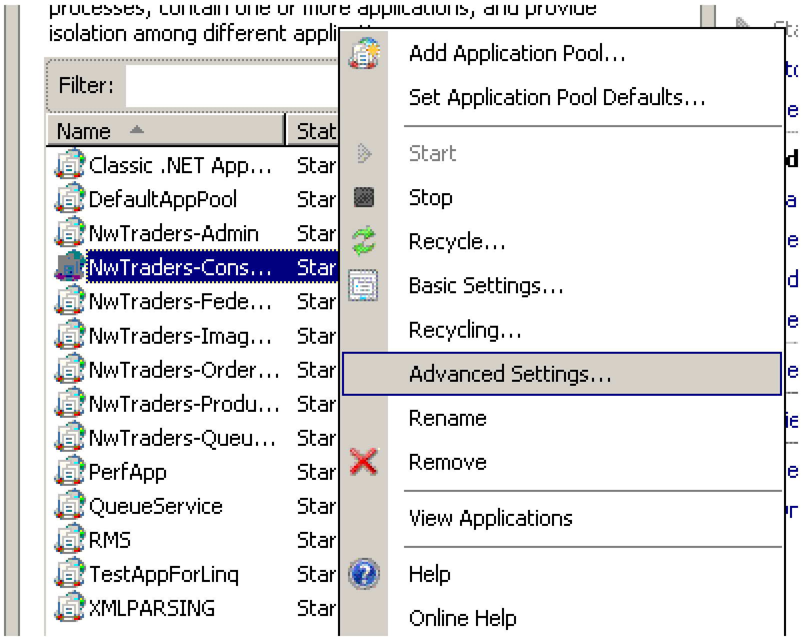

Internet Information Services (IIS) Manager
IP-0A4D46B0 ▸ Application Pools
File View Help
Connections
Start Page
IP-0A4D46B0 (IP-0A4D46B0\Adr
Application Pools
Sites
Application Pools
Filter:
Go
Show All

This page lets you view and manage the list of application pools on the server. Application pools are associated with worker processes, contain one or more applications, and provide isolation among different applications.

Name	Status	.NET Frame...	Managed I
Classic .NET App...	Started	v2.0	Classic
DefaultAppPool	Started	v2.0	Integratec
NwTraders-Admin	Started	v2.0	Integratec
NwTraders-Cons...	Started	v2.0	Integratec

2. Right-click on a particular application pool.

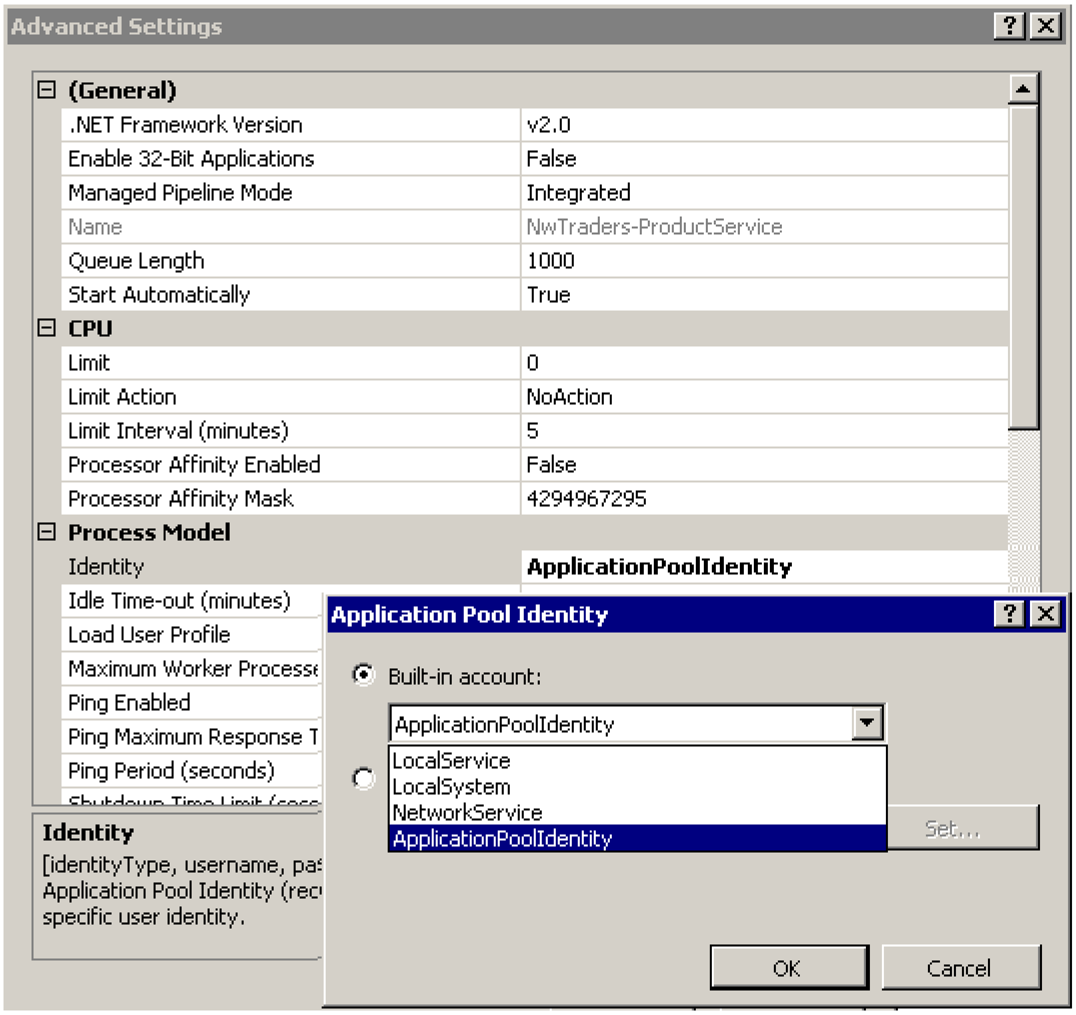
3. Click **Advanced Settings**.



processes, contain one or more applications, and provide isolation among different appli

Name	Stat
Classic .NET App...	Star
DefaultAppPool	Star
NwTraders-Admin	Star
NwTraders-Cons...	Star
NwTraders-Fede...	Star
NwTraders-Imag...	Star
NwTraders-Order...	Star
NwTraders-Produ...	Star
NwTraders-Queu...	Star
PerfApp	Star
QueueService	Star
RMS	Star
TestAppForLinq	Star
XMLPARSING	Star

IIS displays the Application Pool Identity for that application.



Advanced Settings

(General)

.NET Framework Version	v2.0
Enable 32-Bit Applications	False
Managed Pipeline Mode	Integrated
Name	NwTraders-ProductService
Queue Length	1000
Start Automatically	True

CPU

Limit	0
Limit Action	NoAction
Limit Interval (minutes)	5
Processor Affinity Enabled	False
Processor Affinity Mask	4294967295

Process Model

Identity	ApplicationPoolIdentity
Idle Time-out (minutes)	
Load User Profile	
Maximum Worker Processe	
Ping Enabled	
Ping Maximum Response T	
Ping Period (seconds)	

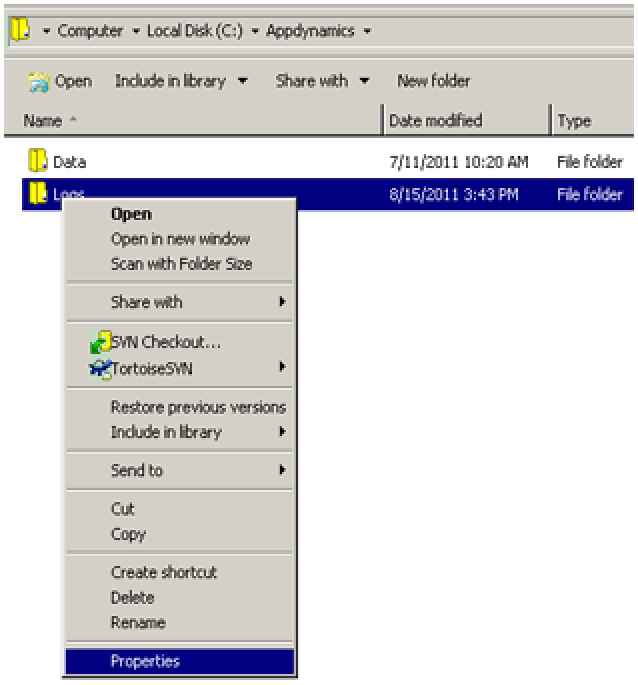
Identity
[identityType, username, pa
Application Pool Identity (rec
specific user identity.

Application Pool Identity

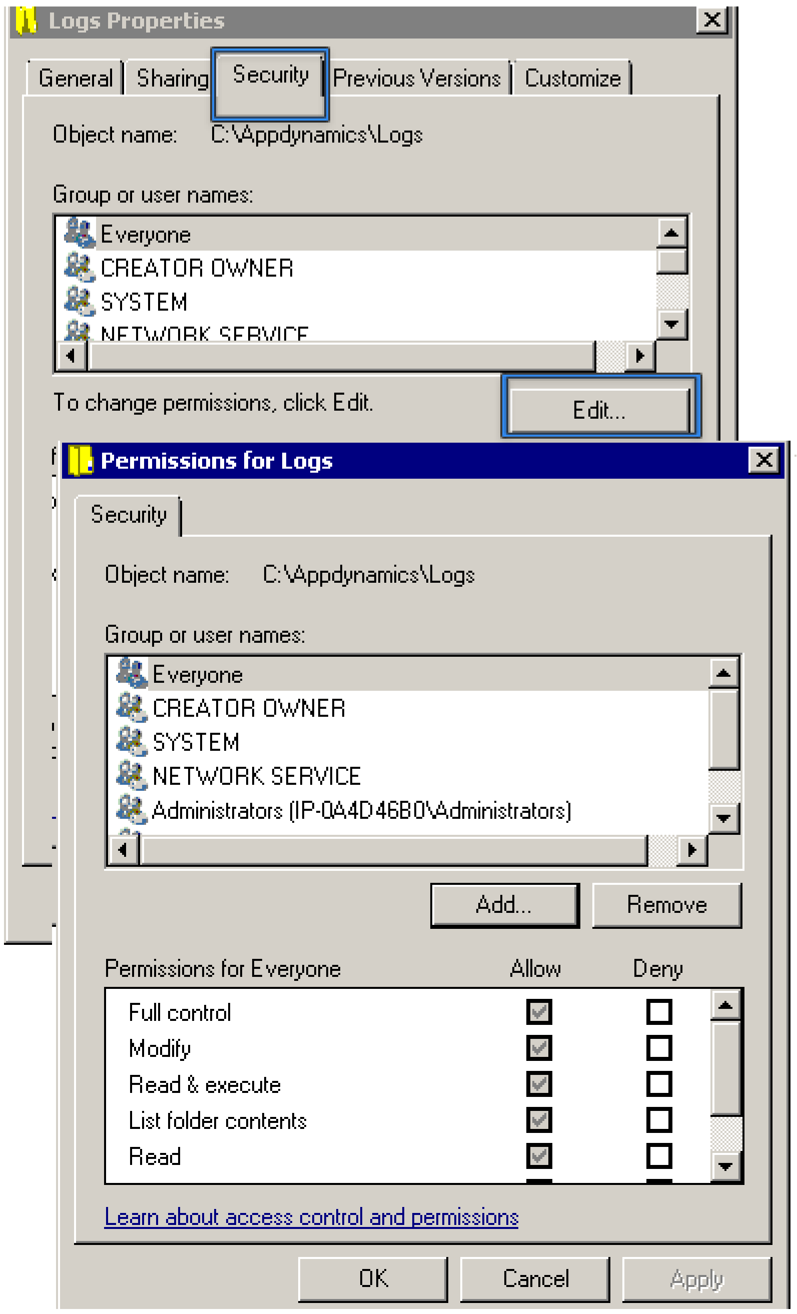
Built-in account: ApplicationPoolIdentity

4. Ensure that your Agent Directory also has the same permissions as your site's application pools.

- Navigate to AppDynamics .NET App Server Agent directory location.
- Right-click on the "logs" directory for the App Server Agent and select **Properties**.



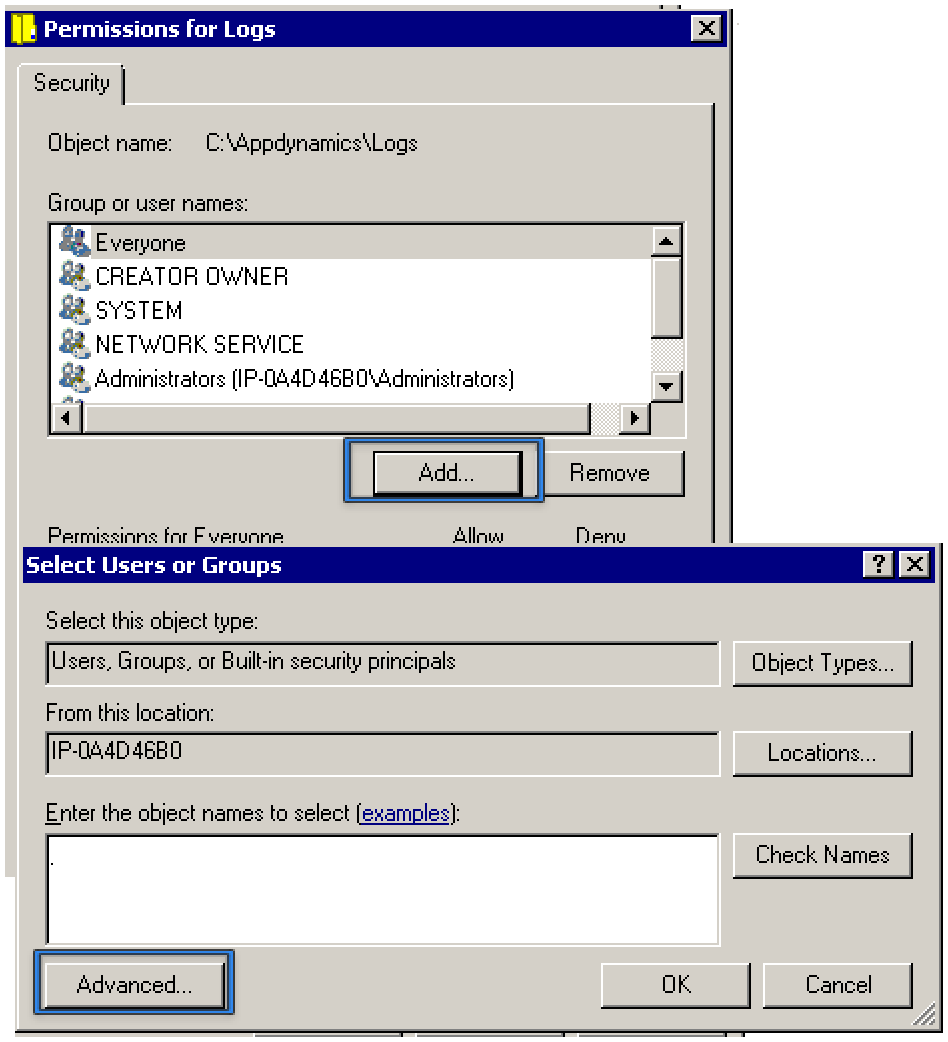
- Click the **Security** tab and verify that the same Application Pool Identity is specified for the .NET Agent directory.



Permissions for Everyone	Allow	Deny
Full control	✓	☐
Modify	✓	☐
Read & execute	✓	☐
List folder contents	✓	☐
Read	✓	☐

If the Agent's logs directory does not have the required permissions:

1. In the Security tab, click **Edit**.
2. Click **Add** to add new permissions to the Agent directory.
3. Click **Advanced**.


Permissions for Logs

Security

Object name: C:\Appdynamics\Logs

Group or user names:

Everyone
CREATOR OWNER
SYSTEM
NETWORK SERVICE
Administrators (IP-0A4D46B0\Administrators)

Permissions for Everyone Allow Deny

Select Users or Groups

Select this object type:
Users, Groups, or Built-in security principals

From this location:
IP-0A4D46B0

Enter the object names to select (examples):

4. Click **Find Now** to find all the users, groups, or built-in security principals on your machine.

5. Select the required group (see the information given below for "**Allowed groups**") from the list and click **OK**.

6. Provide the read and write permissions for the selected user/group/security principal to the Agent directory and click **OK**.

7. Click **Apply**.

Allowed groups for different IIS versions

For IIS v6.x, following settings are applicable for Application Pool Identities:

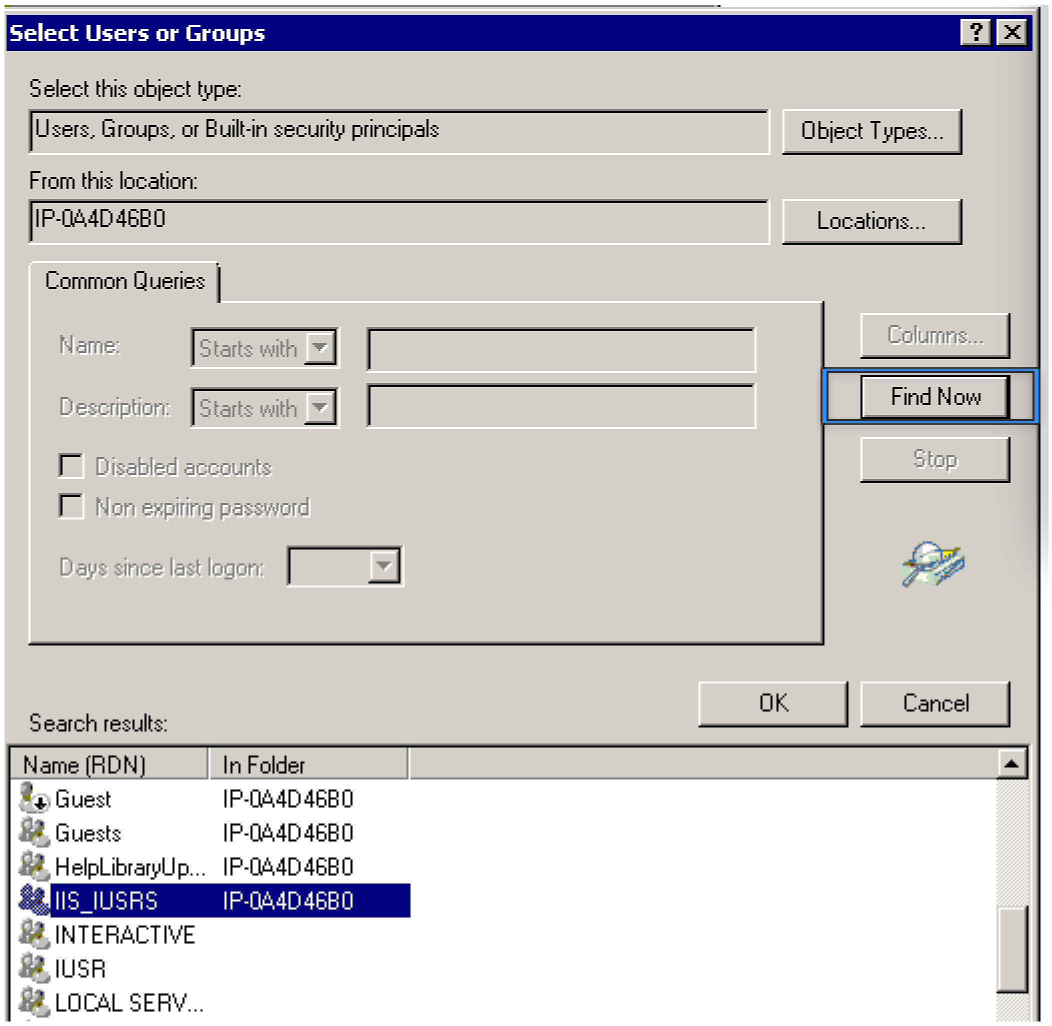
Application Pool Identity	Permission Level
LocalService	LOCAL SERVICE
LocalSystem	SYSTEM
NetworkService	NETWORK SERVICE

Custom Account	Provide the exact name of the account.

For IIS v7.0 and later, following settings are applicable for Application Pool Identities:

Application Pool Identity	Permission Level
LocalService	LOCAL SERVICE
LocalSystem	SYSTEM
NetworkService	NETWORK SERVICE
ApplicationPoolIdentity	Provide the group level permissions for IIS_IUSRS Group (see the screenshot given below).
Custom Account	Provide the exact name of the account.

For example, if your application has the identity "ApplicationPoolIdentity", you must provide the permissions for "IIS_IUSRS" group to your Agent's directory.



Select this object type: Users, Groups, or Built-in security principals

From this location: IP-0A4D46B0

Search results:

Name (RDN)	In Folder
Guest	IP-0A4D46B0
Guests	IP-0A4D46B0
HelpLibraryUp...	IP-0A4D46B0
IIS_IUSRS	IP-0A4D46B0
INTERACTIVE	
IUSR	
LOCAL SERV...	

Learn More

- Install the App Agent for .NET

Configure AppDynamics for .NET

- Enable Thread Correlation for .NET
- Enable Correlation for .NET Remoting
- Configure Backend Detection for .NET
- Configure Business Transaction Detection for .NET
- Configure Application Domain Monitoring
- Instrument the DefaultDomain for Standalone Applications
- Getter Chains in .NET Configurations
- Enable Monitoring for Windows Performance Counters
- Configure the .NET Machine Agent
- Enable Instrumentation for WCF Data Services

Enable Thread Correlation for .NET

- To enable thread correlation for .NET applications

Learn More

This topic describes how to enable thread correlation for .NET applications.

AppDynamics supports instrumentation for Thread.Start and ThreadPool.QueueUserWorkItem on the Common Language Runtime (CLR) 2.x and CLR 4.x.

For versions 3.8 - 3.8.4, the App Agent for .NET instruments ThreadPool.QueueUserWorkItem CLR 4 (`ThreadCorrelationThreadPoolCLR4Instrumentor`) by default. *New in 3.8.5*, you must enable the async-tracking node property.

To enable thread correlation for .NET applications

Configure all instrumentation settings for the App Agent for .NET in the config.xml file. See Where to Configure App Agent Properties.

1. Launch a text editor as administrator.

2. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.

3. Copy the code block below to a child element of the Machine Agent element. (See Machine Agent Element):

```
<instrumentation>
        <instrumentor name="ThreadCorrelationThreadPoolCLR2Instrumentor"
enabled="true"/>
        <instrumentor name="ThreadStartCLR2Instrumentor" enabled="true"/>
        <instrumentor name="ThreadStartCLR4Instrumentor" enabled="true"/>
    </instrumentation>
```

For 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">
...
 <machine-agent>
    <!--Enable thread correlation-->
    <instrumentation>
        <instrumentor name="ThreadCorrelationThreadPoolCLR2Instrumentor"
enabled="true"/>
        <instrumentor name="ThreadStartCLR2Instrumentor" enabled="true"/>
        <instrumentor name="ThreadStartCLR4Instrumentor" enabled="true"/>
    </instrumentation>
  </machine-agent>
...
</appdynamics-agent>
```

The configuration syntax is **enabled="true"**.

4. Save the config.xml file.

5. New in 3.8.5, enable the async-tracking node property.

6. Restart the AppDyanmics.Agent.Coordinator Service.

7. Restart instrumented applications for your changes to take effect.

Learn More

- Configure Backend Detection for .NET
- Monitor Remote Services

Enable Correlation for .NET Remoting

Developers use .NET remoting to build distributed applications that share objects across processes or across application domains running in the same process. AppDynamics doesn't enable correlation for .NET remoting functions by default.

Instrumenting Applications That Use .NET Remoting

You can configure the App Agent for .NET to discover .NET remoting entry and exit points.

To enable correlation for .NET remoting

1. Launch a text editor as administrator.

2. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.
3. Copy the code block below to a child element of the Machine Agent element. (See Machine Agent Element):

```
<instrumentation>
        <instrumentor name="RemotingMscorlibEntryInstrumentor"
enabled="true"/>
        <instrumentor name="RemotingExitInstrumentor"
enabled="true"/>
    </instrumentation>
```

   For 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">
...
 <machine-agent>
    <!--Enable correlation for .NET remoting-->
    <instrumentation>
        <instrumentor name="RemotingMscorlibEntryInstrumentor"
enabled="true"/>
        <instrumentor name="RemotingExitInstrumentor"
enabled="true"/>
    </instrumentation>
  </machine-agent>
...
</appdynamics-agent>
```

4. Save the config.xml file.
5. Restart the AppDyanmics.Agent.Coordinator Service.
6. Restart instrumented applications for your changes to take effect.

If the agent doesn't discover the entry points after configuration, specify an agent trigger.

To specify an agent trigger

.NET remoting entry point functions execute in low-level .NET libraries that may not trigger automatic agent instrumentation. *New in 3.8.1*, if the agent doesn't discover the .NET remoting entry points after configuration you can specify a function that triggers the agent to begin instrumentation.

1. Identify a function to trigger the agent to begin instrumentation. The function can be any function that executes as part of the application process.

   For example, consider the following code for a MovieTicket remoting object. In this case, use the function GetTicketStatus to trigger the agent.

```
using System;
namespace MovieGoer
{
    public class MovieTicket : MarshalByRefObject
    {
        public MovieTicket()
        {
        }
        public string GetTicketStatus(string stringToPrint)
        {
            return String.Format("Enquiry for {0} -- Sending back
status: {1}", stringToPrint, "Ticket Confirmed");
        }
    }
}
```

2. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.
3. Update the Instrumentation element to include the AgentTriggerInstrumentor. (See Machine Agent Element):

```
<instrumentation>
  <instrumentor name="AgentTriggerInstrumentor" enabled="true"
args="" />
        <instrumentor name="RemotingMscorlibEntryInstrumentor"
enabled="true"/>
        <instrumentor name="RemotingExitInstrumentor"
enabled="true"/>
    </instrumentation>
```

4. Set the AgentTriggerInstrumentor `args` value to the name of the trigger function from step 1.

   For 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">
...
 <machine-agent>
    <!--Enable correlation for .NET remoting-->
    <instrumentation>
        <instrumentor name="AgentTriggerInstrumentor" enabled="true"
args="MovieGoer.MovieTicket.GetTicketStatus" />
        <instrumentor name="RemotingMscorlibEntryInstrumentor"
enabled="true"/>
        <instrumentor name="RemotingExitInstrumentor"
enabled="true"/>
    </instrumentation>
  </machine-agent>
...
</appdynamics-agent>
```

5. Save the config.xml file.
6. Restart instrumented applications for your changes to take effect.

Learn More

- Configure Backend Detection for .NET
- Monitor Remote Services

Configure Backend Detection for .NET

To review general information about monitoring databases and remote services (collectively known as backends) and for an overview of backend configuration see Backend Monitoring.

Types of Exit Points

Each automatically discovered backend type has a default discovery rule and a set of configurable properties. See the following:

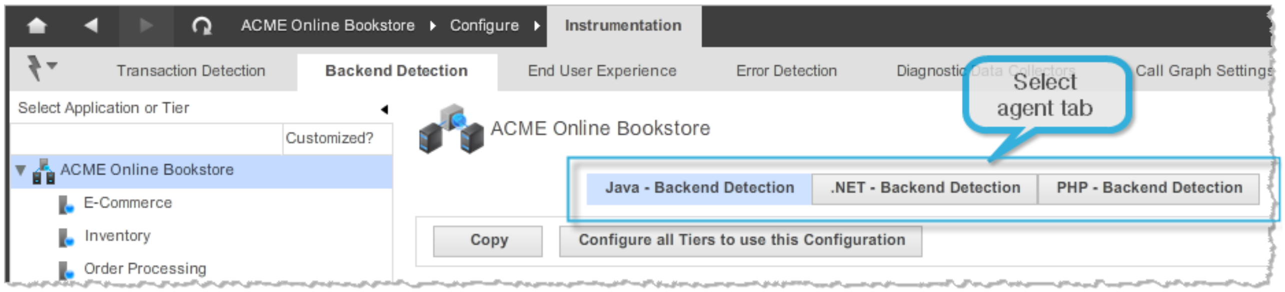
- WCF Exit Points for .NET
- Message Queue Exit Points for .NET
- ADO.NET Exit Points
- HTTP Exit Points for .NET
- Web Services Exit Points for .NET
- Configure Custom Exit Points for .NET

For information on All Other Traffic transactions, see Backend Monitoring.

View the Discovery Rules

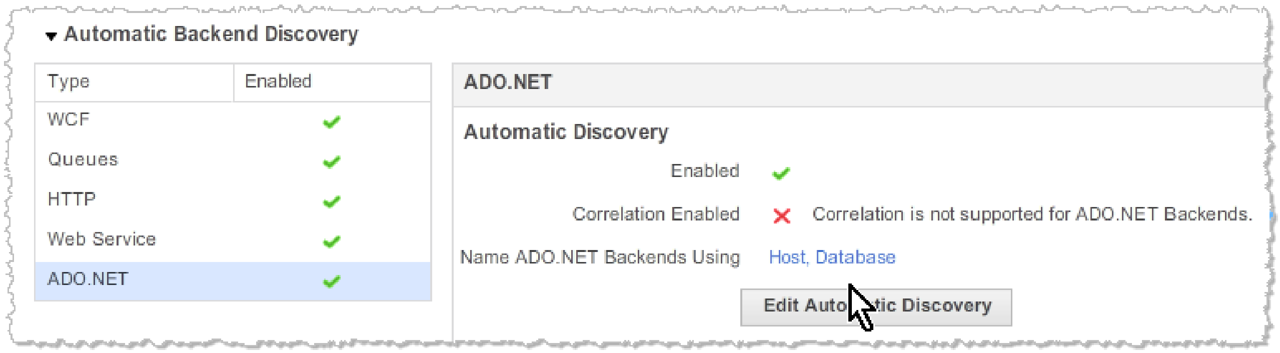
To view the discovery rules for an automatically discovered backend, access the backend configuration screen using these steps:

1. Select the application.
2. In the left navigation pane, click **Configure -> Instrumentation**.
3. Select the **Backend Detection** tab.
4. Select the application and the tab corresponding to your agent platform (Java, .NET, PHP).



The Automatic Backend Discovery default configurations for that agent are listed.

5. In the Automatic Backend Discovery list, click the backend type to view.
A summary of the configuration appears on the right. For example, the following figure shows that JMS backends are auto-discovered using the Destination, Destination Type, and Vendor.



Automatic Backend Discovery

Type	Enabled
WCF	✓
Queues	✓
HTTP	✓
Web Service	✓
ADO.NET	✓

ADO.NET

Automatic Discovery

Enabled	✓
Correlation Enabled	✗ Correlation is not supported for ADO.NET Backends.
Name ADO.NET Backends Using	Host, Database

Revise Backend Discovery Rules

If the default settings don't give you exactly what you need, you can refine the configuration in the following ways:

- Change the default discovery rules:
  - Enable or disable one or more of the properties
  - Use one or more specified segments of a property
  - Run a regular expression on a property
  - Execute a method on a property
- Add new discovery rules
- Add custom exit points

The precise configurations vary according to the backend type. These general features are configurable:

- Discovery Enabled - You can enable and disable automatic discovery for the backend type. Undiscovered backends are not monitored.
- Correlation Enabled - You can enable and disable correlation. Correlation enables AppDynamics to tag, trace, and learn about application calls to and through the backend to other remote services or tiers. For example, if a call is made from Tier1 -> Backend1 -> Tier2, Tier2 knows about the transaction flow because the agent "tags" the outbound call from Backend1 to identify it as related to the same transaction that called Backend1 from Tier1. If you do not care about activity downstream from the backend, you may want to disable correlation.
- Backend Naming - You can configure how backends are named.

Change the Default Discovery Rules

When you need to revise the default set of discovery rules, in many cases, you can achieve the visibility you need by making adjustments to the default automatic discovery rules. For some scenarios, you might want to disable some or all of the default rules and create custom rules for detecting all your backends. AppDynamics provides flexibility for configuring backend detection.

For example, detection of HTTP backends is enabled by default. In Java environments, HTTP backends are identified by the host and port and correlation with the application is enabled. To change the discovery rule for HTTP backends in some way, such as disabling correlation, omitting a property from the detected name, or using only certain segments of a property in the name, you edit the HTTP automatic discovery rule.

Review the rules for each exit point to determine the best course of action for your application if the default discovery rules do not give the results you need for your monitoring strategy.

To change default backend automatic discovery rules

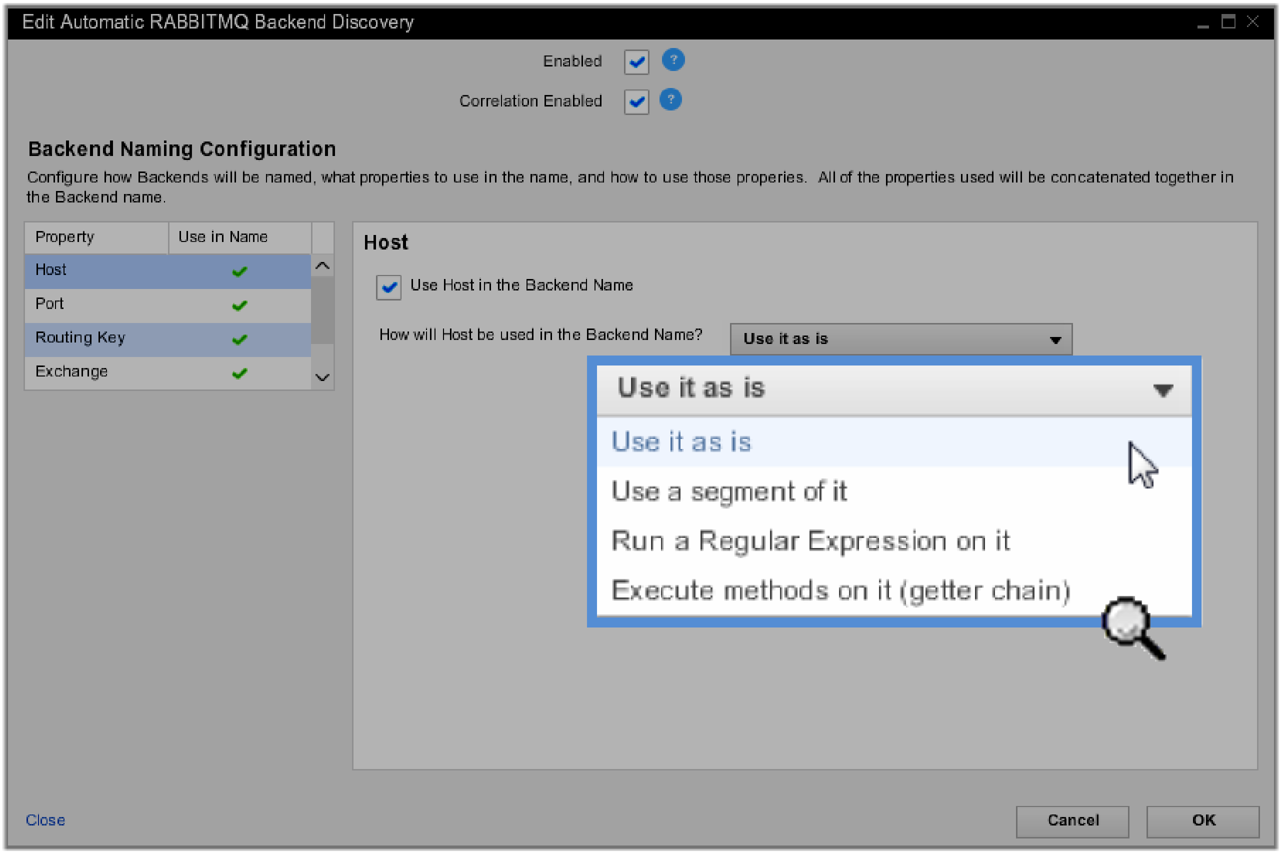
1. From the left-hand navigation panel, select **Configure -> Instrumentation**. Then select the **Backend Detection** tab and the application or tier you want to configure.

2. In the Automatic Backend Discovery list, select the backend type to modify.
The rule summary appears in the Automatic Discovery panel on the right.

3. Click **Edit Automatic Discovery**.
The Edit Automatic Backend Discovery Rule window appears.

4. For each property that you want to configure:

- Select the property in the property list.
- Check the property check box to use the property for detection; clear the check box to omit it.
- If you are using the property, choose how the property is used from the drop-down list.



Edit Automatic RABBITMQ Backend Discovery

Enabled ✓

Correlation Enabled ✓

Backend Naming Configuration

Configure how Backends will be named, what properties to use in the name, and how to use those properies. All of the properties used will be concatenated together in the Backend name.

Property	Use in Name
Host	✓
Port	✓
Routing Key	✓
Exchange	✓

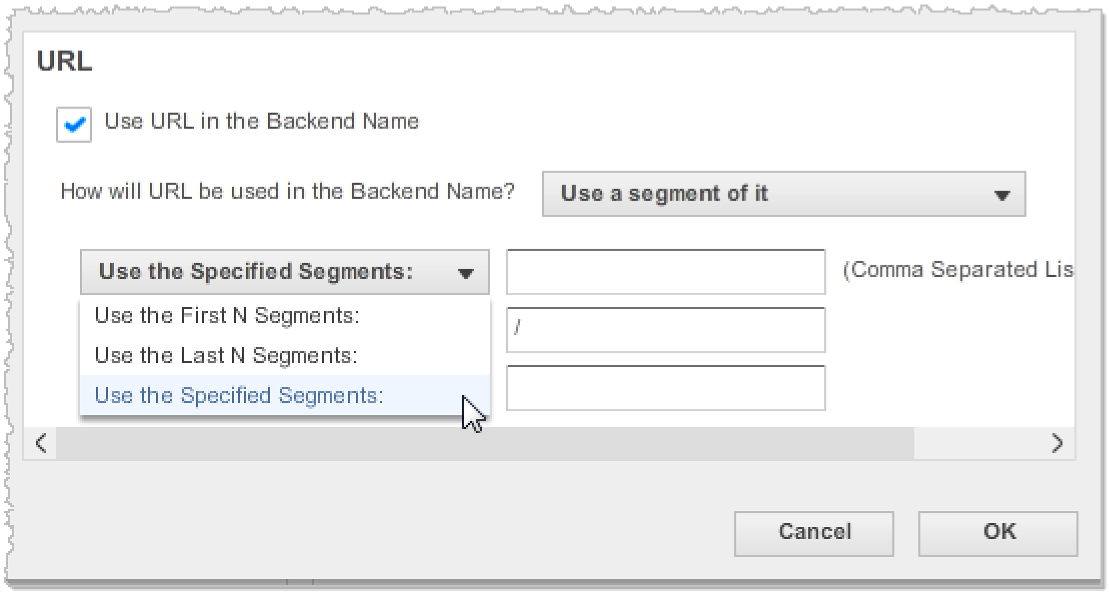
Host

✓ Use Host in the Backend Name

How will Host be used in the Backend Name? Use it as is

Use it as is
Use it as is
Use a segment of it
Run a Regular Expression on it
Execute methods on it (getter chain)
Close
Cancel
OK

- If you have a complex property, such as the URL, destination, or a query string, and you want to eliminate some parts of it or need some additional manipulation you can use an option from the second drop-down list such as **Run a Regular Expression on it** or **Execute methods on it (getter chain)**. Each option has associated configuration parameters. For example, you have options for manipulating the segments of the **URL**.


URL
Use URL in the Backend Name
How will URL be used in the Backend Name?
Use a segment of it
Use the Specified Segments:
(Comma Separated Lis
Use the First N Segments:
Use the Last N Segments:
Use the Specified Segments:
/
Cancel
OK

5. Check **Enabled** to enable the rule; clear the check box to disable it.

6. Check **Correlation Enabled** to enable correlation.

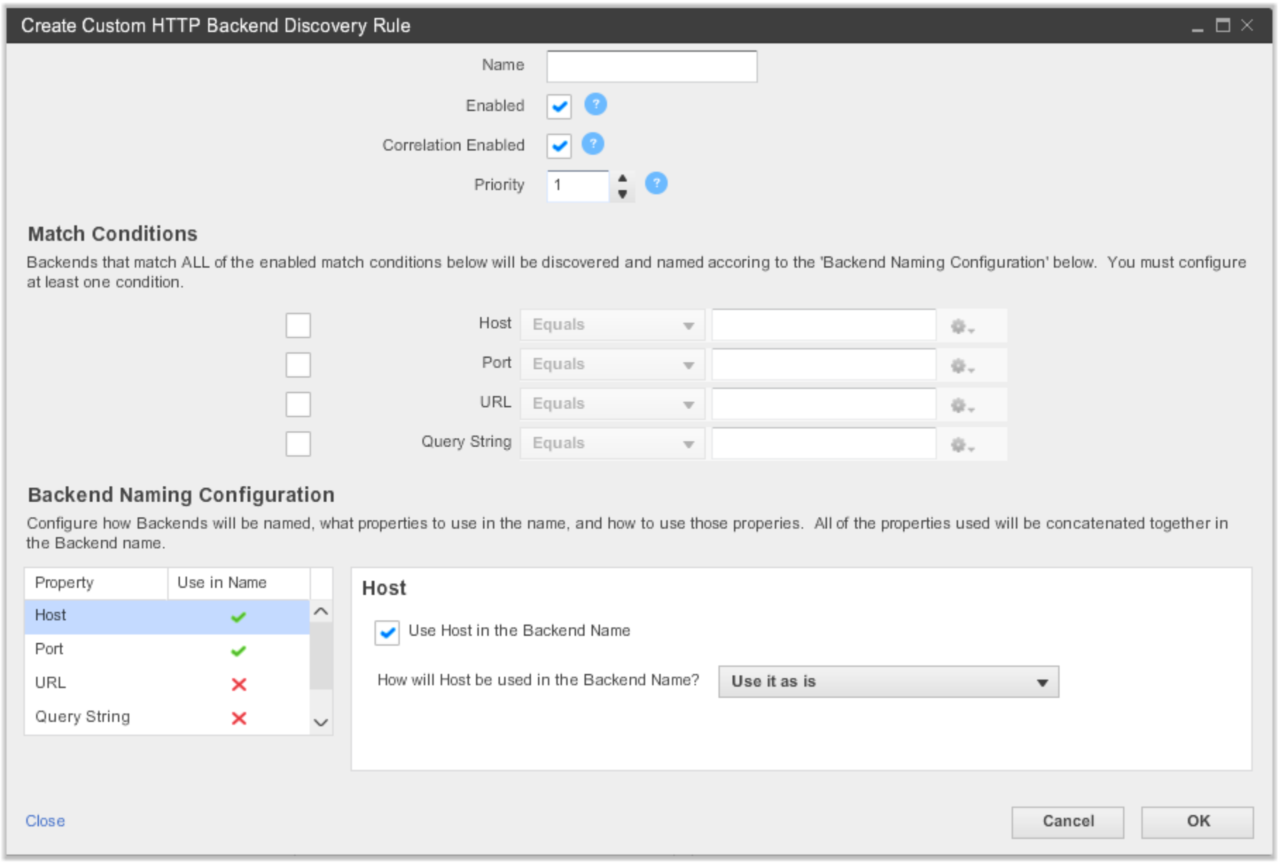
7. Click **OK**.

Add Backend Discovery Rules

AppDynamics provides the additional flexibility to create new custom discovery rules for the automatically discovered backend types. Custom rules include the following settings:

- **Name** for the custom rule.
- **Priority** used to set precedence for custom rules.
- **Match Conditions** used to identify which backends are subject to the custom naming rules.
- **Backend Naming Configuration** used to name the backends matching the match conditions.

The window for adding custom discovery rules looks like this:



Create Custom HTTP Backend Discovery Rule

Name	
Enabled	✓
Correlation Enabled	✓
Priority	1

Match Conditions

Backends that match ALL of the enabled match conditions below will be discovered and named accoring to the 'Backend Naming Configuration' below. You must configure at least one condition.

☐	Host	Equals	
☐	Port	Equals	
☐	URL	Equals	
☐	Query String	Equals	

Backend Naming Configuration

Configure how Backends will be named, what properties to use in the name, and how to use those properies. All of the properties used will be concatenated together in the Backend name.

Property	Use in Name
Host	✓
Port	✓
URL	✗
Query String	✗

Host

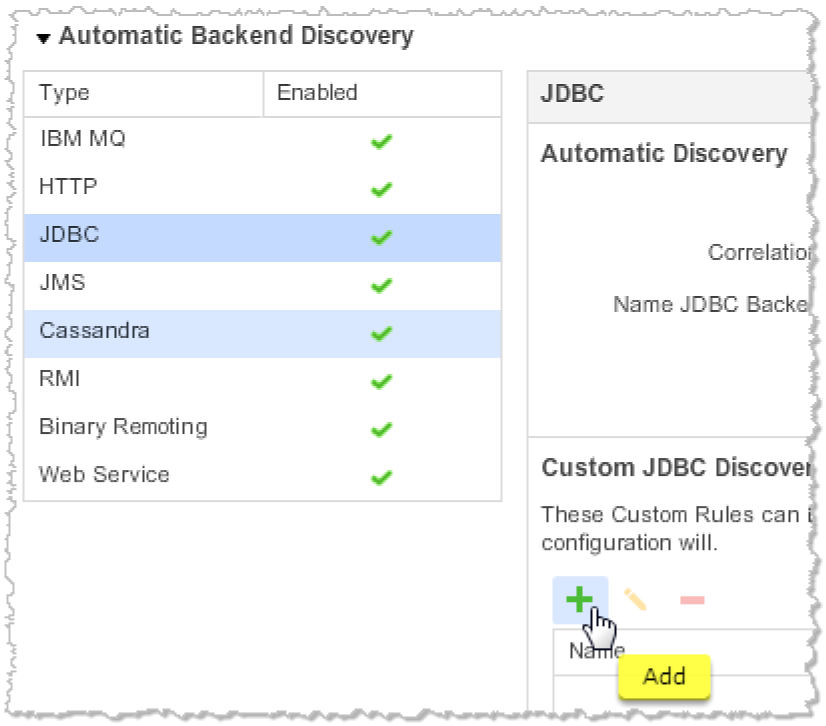
✓ Use Host in the Backend Name

How will Host be used in the Backend Name? Use it as is

To create a custom discovery rule for an automatically discovered backend type, use these steps:

1. In the Automatic Backend Discovery list, select the backend type.
The Custom Discovery Rules editor appears in the right panel below the Automatic Discovery panel.

2. Click **Add** (the + icon) to create a new rule or select an existing rule from the list and click the edit icon to modify one.



Type	Enabled
IBM MQ	✓
HTTP	✓
JDBC	✓
JMS	✓
Cassandra	✓
RMI	✓
Binary Remoting	✓
Web Service	✓

These Custom Rules can i
configuration will.

3. Enter a name for the custom rule.

4. Confirm the settings for **Enabled** and **Correlation Enabled** (if applicable).

4. Enter the priority for the custom rule compared to other custom rules for this backend type. The higher the number, the higher the priority. A value of 0 (zero) indicates that the default rule should be used.

5. In the next section, configure the match conditions.
Match conditions are used to identify which backends should use the custom rule. Backends that do not meet all the defined match conditions are discovered according to the default rule.

7. In the next section, configure the naming for the backends matching the rule. The naming configuration must include the property used by the match condition.

8. Save the configuration.
See specific exit points for examples.

Add Custom Exit Points

When your application has backends that are not automatically discovered, you can enable discovery using custom exit points. To do this, you need to know the class and method used to identify the backend. See Configure Custom Exit Points for Java.

Propagate Changes to Other Tiers or Applications

When you have made changes to the backend detection rules, you may want to propagate your changes to other tiers or applications.

To copy an entire backend detection configuration to all tiers

1. Access the backend detection window. See View the Discovery Rules.
2. In the left panel select the application or tier whose configuration you want to copy.
3. Click **Configure all Tiers to use this Configuration**.

To copy an entire backend detection configuration to another tier or application

1. Access the backend detection window. See View the Discovery Rules.
2. In the left panel select the application or tier whose configuration you want to copy.
3. Click **Copy**.
4. In the Application/Tier browser, choose the application or tier to copy the configuration to.
5. Click **OK**.

Learn More

- Backend Monitoring
- Monitor Databases
- Monitor Remote Services

WCF Exit Points for .NET

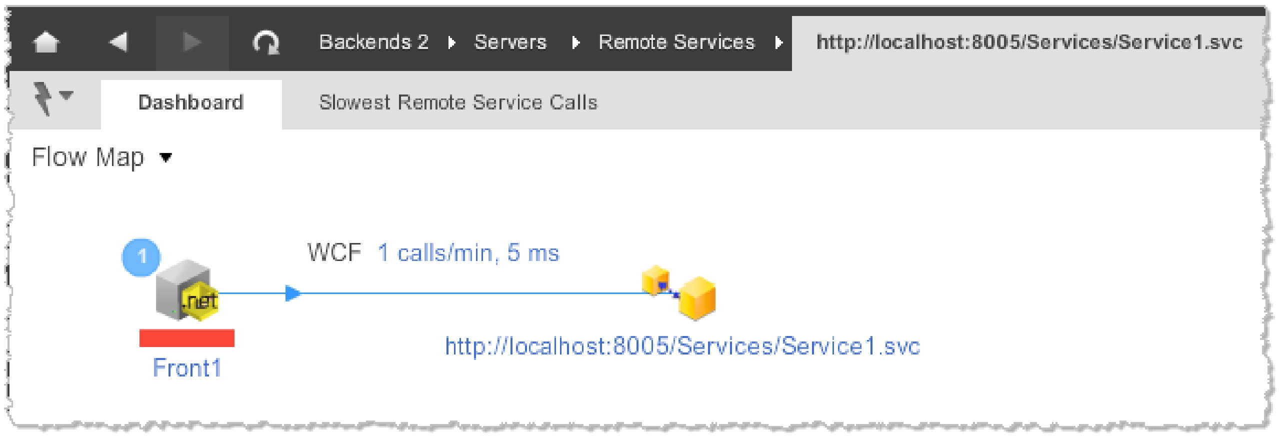
- Auto-Discovery and Default Naming
- Configurable Properties
- Changing the Default WCF Automatic Discovery and Naming
- Learn More

This topic explains WCF exit point configuration. To review general information about monitoring databases and remote services (collectively known as backends) and for an overview of backend configuration see Backend Monitoring.

Auto-Discovery and Default Naming

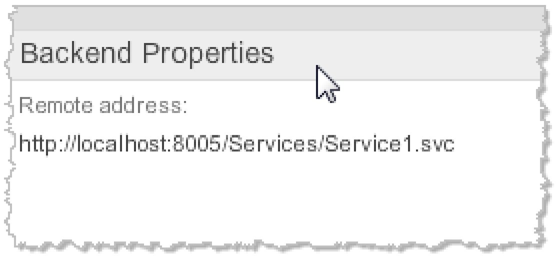
By default, AppDynamics automatically detects and identifies WCF exit points (backends) when an application uses the WCF client libary.

The default WCF automatic discovery rule uses the remote address property. From the enabled properties AppDynamics derives a display name using the remote address, for example:


Backends 2
Servers
Remote Services
http://localhost:8005/Services/Service1.svc
Dashboard
Slowest Remote Service Calls
Flow Map
1
.net
WCF 1 calls/min, 5 ms
http://localhost:8005/Services/Service1.svc
Front1

By default, AppDynamics groups multiple backends of the same type together on the application flow map.

The backend properties for the WCF backend can be viewed on the Remote Services dashboard.



Remote address:
http://localhost:8005/Services/Service1.svc

Configurable Properties

Configurable Properties	Property Used by Default in Detection and Naming	Description
Remote Address	Yes	URL minus the query, fragment and user information (name and password)
Operation Contract	No	WCF operation name
URL	No	full URL
Host	No	host portion of URL
Port	No	port number if present in the URL, otherwise protocol default
SOAP Action	No	In the case that a web service is called, the SOAP action

Changing the Default WCF Automatic Discovery and Naming

Depending on what you need to monitor, you can change the default configuration to use other properties.

For procedures, see Configure Backend Detection for .NET.

Learn More

- Backend Monitoring
- Monitor Remote Services
- Configure Custom Exit Points

Message Queue Exit Points for .NET

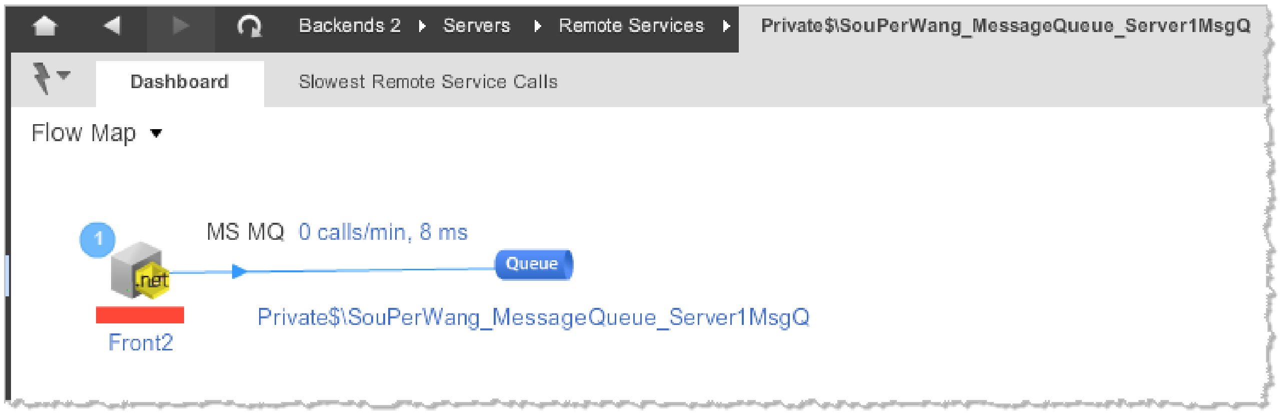
- Auto-Discovery and Default Naming
- Queues Configurable Properties
- Changing the Default Queues Automatic Discovery and Naming
- Learn More

This topic explains exit point configuration for message queue exit points. To review general information about monitoring databases and remote services (collectively known as backends) and for an overview of backend configuration see Backend Monitoring.

Auto-Discovery and Default Naming

By default, AppDynamics automatically detects and identifies many message queue exit points. For a list of the supported message-oriented middleware products, see Supported Remote Services for the App Agent for .NET.

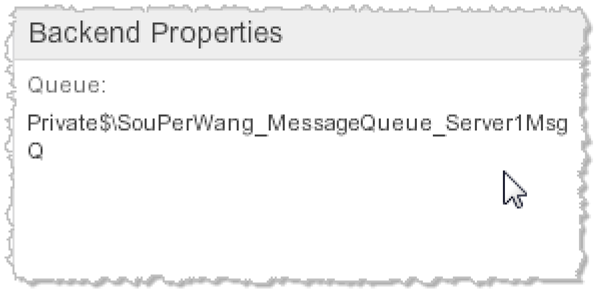
The default queue automatic discovery rule uses the destination property. From the enabled properties AppDynamics derives a display name as shown here:



Backends 2
Servers
Remote Services
Private$\SouPerWang_MessageQueue_Server1MsgQ
Dashboard
Slowest Remote Service Calls
Flow Map
1
.net
MS MQ 0 calls/min, 8 ms
Queue
Front2
Private$\SouPerWang_MessageQueue_Server1MsgQ

By default, AppDynamics groups multiple backends of the same type together on the application flow map.

The backend properties for the queue can be viewed on the Remote Server dashboard.


Backend Properties

Queue:

Private$\SouPerWang_MessageQueue_Server1MsgQ

Queues Configurable Properties

In general, the properties listed below are used for queue exit points. However, each message-oriented product is different and there may be variations in the properties or their names.

Configurable Properties	Property Used by Default in Detection and Naming	Description
Host	No	Queue server name
Destination	Yes	Name of topic or queue
Destination Type	No	queue or topic
Vendor	No	Vendor from the client library

Note
For MSMQ, correlation for downstream calls is not supported.
For .NET remoting, additional configuration is needed to get downstream correlation. See Enable Correlation for .NET Remoting.

Changing the Default Queues Automatic Discovery and Naming

Depending on what you need to monitor, you can change the default configuration by disabling one or more properties.
For configuration procedures, see Configure Backend Detection for .NET.

Learn More

- Backend Monitoring
- Monitor Remote Services
- Enable Correlation for .NET Remoting
- Configure Custom Exit Points

Monitor RabbitMQ Backends for .NET

    - HandleBasicDeliver Method
- Learn More

The App Agent for .NET (agent) automatically discovers RabbitMQ remote services. This topic covers the methods we instrument for RabbitMQ backends and the backend naming convention.

Remote Service Detection

AppDynamics auto-detects RabbitMQ backends based upon calls from instrumented tiers. RabbitMQ exit points are methods that publish or push messages to a queue. RabbitMQ entry points are methods that listen or poll for new messages in the queue.

To see RabbitMQ in the list of backends, in the left navigation pane click **Servers->Remote Services**.

Exit Points

The agent discovers a RabbitMQ backend exit point when your application sends a message to the queue using the `BasicPublish()` method.

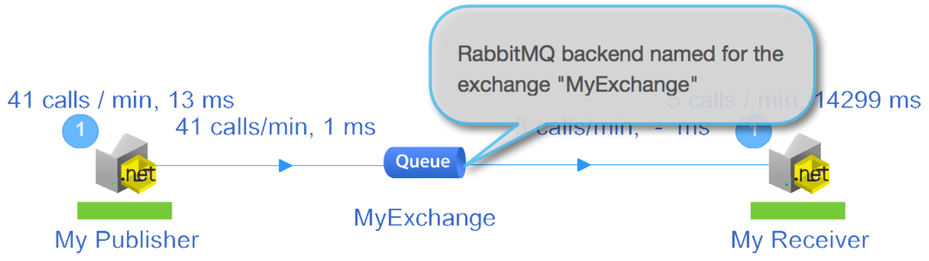
Backend Naming

By default, the agent names the RabbitMQ backend for the exchange parameter of the `BasicPublish()` method.

For example:

```
model.BasicPublish("MyExchange", "", false, false,
    basicProperties, Encoding.UTF8.GetBytes(message));
```

In this case the agent names the queue **MyExchange**.



You can refine the backend name to include some or all segments of the routing key. To configure RabbitMQ naming you must be familiar with your implementation RabbitMQ exchanges and routing keys. See RabbitMQ Exchanges and Exchange Types.

To refine backend naming

Register the **rmqsegments** node property. For instructions on how to set a node property, see App Agent Node Properties.

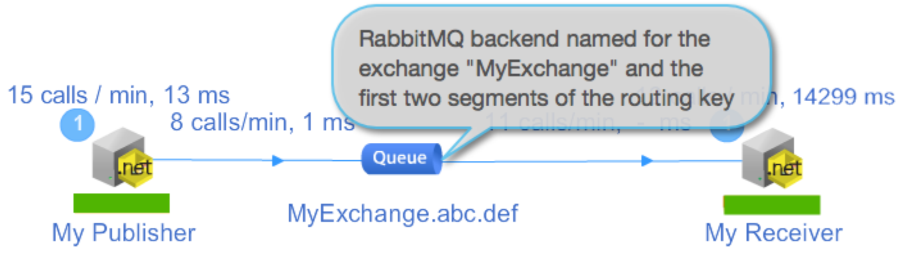
Name: rmqsegments

Description: "Configure RabbitMQ naming to include routing key segments."
Type: Integer
Value: <integer>
The routing key is a string. The agent treats dot-separated (".") substrings of the routing key as segments. Set the value to an integer that represents the number of routing key segments to include in the name.

In the following example the routing key is **abc.def.ghi**. Set the rmqsegments value to "2" to name the queue **MyExchange.abc.def**.

```
model.BasicPublish("MyExchange", "abc.def.ghi", false, false,
    basicProperties, Encoding.UTF8.GetBytes(message));
```

After you save the node property, the Controller sends the configuration to the agent. After some time the RabbitMQ backend shows up with the new name.


RabbitMQ backend named for the exchange "MyExchange" and the first two segments of the routing key
15 calls / min, 13 ms
8 calls/min, 1 ms
min, 14299 ms
Queue
My Publisher
MyExchange.abc.def
My Receiver

Entry Points

The agent discovers RabbitMQ backend entry point when your application polls the the queue. AppDynamics auto-detects RabbitMQ based upon the following patterns:

- BasicGet
- HandleBasicDeliver

BasicGet Method

The agent detects the pattern below where the application periodically polls the message queue using the `BasicGet()` method. The call timing is limited to time spent inside the `while` loop. The timer only starts when the BasicGet result returns a value at line 4. The timer ends when the next BasicGet executes. In this example, the application polls every five seconds, so the execution time equals the time in the `if` loop plus five seconds.

```
while (true)
    {
       var result = channel.BasicGet("MyExchange", true);
       if (result != null)
             {
                 var body = result.Body;
                 var message = Encoding.UTF8.GetString(body);
                 Console.WriteLine("Received: {0}.", message);
              }

                    Thread.Sleep(5000);
     }
```

HandleBasicDeliver Method

The agent detects the `HandleBasicDeliver()` method for custom implementations of the `IBasicConsumer` interface. In this case the call timing reflects the execution time for the HandleBasicDeliver method.

Learn More

RabbitMQ Exchanges and Exchange Types

RabbitMQ Monitoring Extension

App Agent Node Properties

App Agent Node Properties Reference by Type

ADO.NET Exit Points

This topic explains ADO.NET exit point configuration. To review general information about monitoring databases and remote services (collectively known as backends) and for an overview of backend configuration see Backend Monitoring.

Auto-Discovery and Default Naming

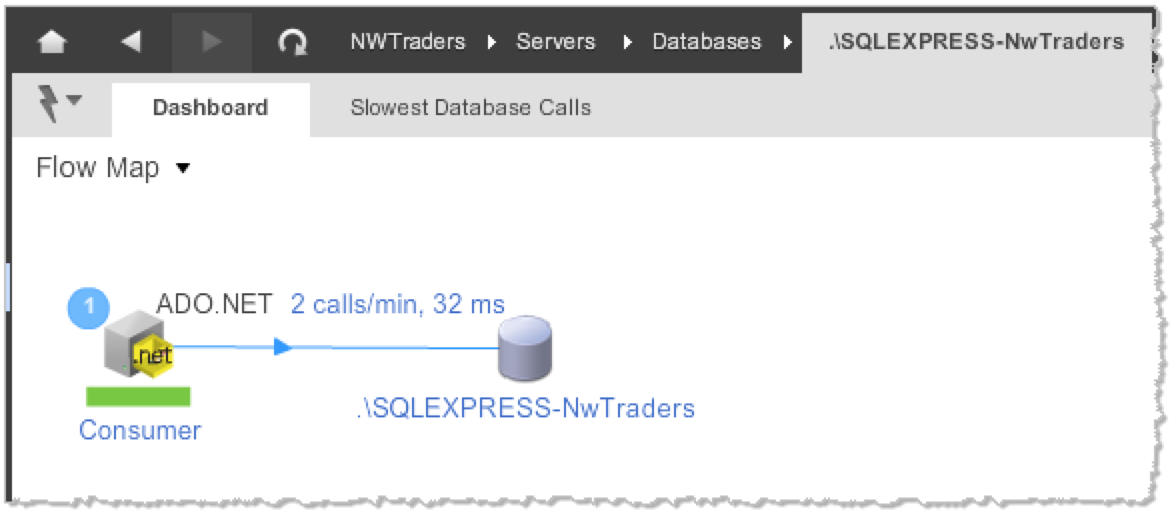
ADO.NET data providers implementing standard Microsoft interfaces are automatically discovered as backends. For a complete list, see Supported ADO.NET Clients for the .NET Agent.

Because the ADO.NET API is interface-based, by default AppDynamics instruments all ADO.NET database providers that implement these interfaces.

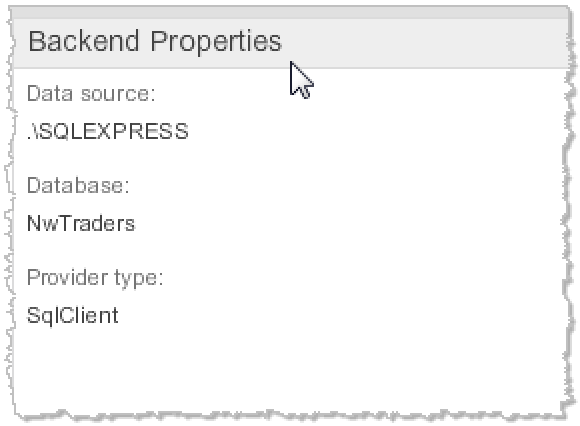
For database identification, AppDynamics uses information from the ADO.NET connection string.

The connection string specifies the server address and schema or the local file name. Most connection strings are formatted according to well-known rules that can be parsed and distilled to a database name. However, because there is no standard on the connection string, it is up to the ADO.NET provider implementer to choose the format.

For some providers, AppDynamics may fail to parse the connection string. In these cases, the .NET agent uses the complete connection string minus any user password. The property is labeled **ADO.NET connection string** and the value shows the connection string minus any user password.



The backend properties can be viewed on the **Database Dashboard**.



Backend Properties

Data source:
.\SQLEXPRESS

Database:
NwTraders

Provider type:
SqlClient

Configurable Properties

You can enable or disable the use of the following properties for ADO.NET exit points.

Configurable Properties	Property Used by Default in Detection and Naming	Description
Host	Yes	Data source or database server. Labeled **Data Source** on the Backend Dashboard.

Database	Yes	Database
Vendor	No	Type of the client-side ADO.NET library. Labeled **Provider type** on the Backend Dashboard
Connection String	No	Full connection string with password filtered out
Port	No	Port number. This is seldom used to identify ADO.NET databases.

Changing the Default ADO.NET Automatic Discovery and Naming

There may be times when you want to change the default configuration depending on exactly what you need to monitor. For example, when you have multiple databases on the same server, you may need to revise the automatic discovery rule. Doing this enables you to more effectively monitor the key performance indicators (KPIs) that are of most interest to you.
For configuration procedures, see Configure Backend Detection for .NET.

Examples

Different ADO.NET Providers

If you have different ADO.NET providers connecting to a single Oracle database, you might be using both the Microsoft implementation of ADO.NET and the Oracle client, ODP.NET. In this case, you could enable the Vendor property to split the single Oracle backend into two separate backends.

Learn More

- Backend Monitoring
- Monitor Remote Services
- Configure Custom Exit Points

Resolve Unknown0 Database Backend Name

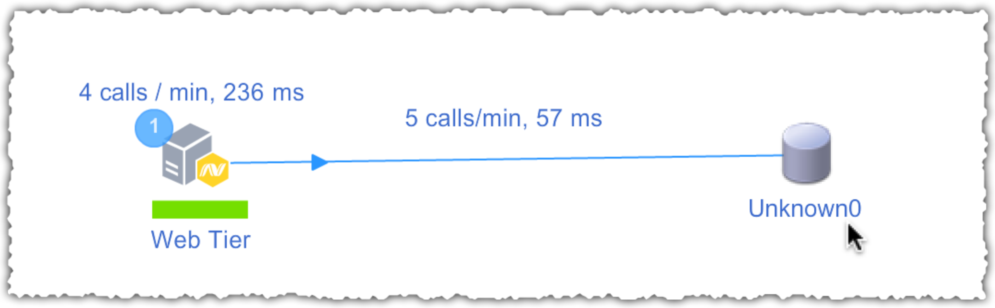
- Database Backends Named "Unknown0"
    - To enable database naming for your backend

Certain database connection strings cause ODP.NET database backends to appear with the label "Unknown0". This topic provides instructions on how to configure the App Agent for .NET to detect the database name. For general information on database backend naming, see ADO.NET Exit Points.

Database Backends Named "Unknown0"

The default database backend detection mechanism for the App Agent for .NET sometimes identifies an ODP.NET database as "Unknown0" based upon the connection string. If this is the case with your business application, you may observe one of the following:

- The backend displays the label "Unknown0" in the flow map:



- The AgentLog.txt references an incompatible connection string:

```
2014-03-26 11:05:08.8272 4716 w3wp 2 10 Warn ADOUtils ADO.NET backend:
Unknown0 is using incompatible connection string: User Id=MYUSER;Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=
54.235.245.21)(PORT=1521))(CONNECT_DATA=(SID=XE)))
```

  For locations of the agent logs, see App Agent for .NET log files.

The following is an example connection string that produces the "Unknown0" name:

```
string CONNECTION_STRING = "User Id=MYUSER;Password=welcome1;Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=
54.235.245.21)(PORT=1521))(CONNECT_DATA=(SID=XE)));";
```

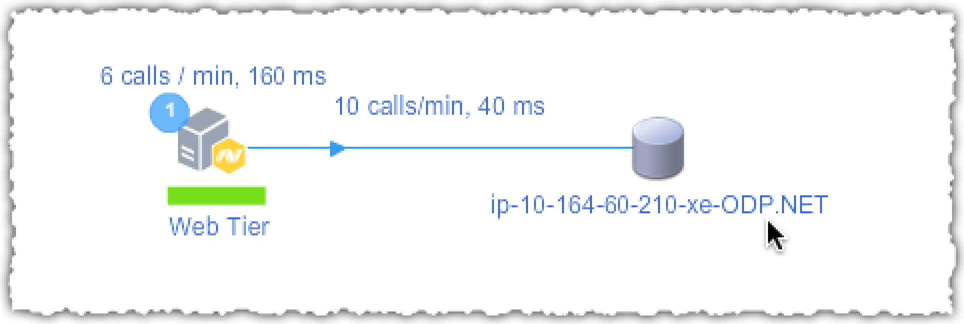
To enable database naming for your backend

1. Register the **ado-new-resolvers** node property.

   For instructions on how to set a node property, see App Agent Node Properties.
   Name: ado-new-resolvers
   Description: Enable database detection and naming for ODP.NET backends labeled "Unknown0".
   Type: Boolean
   Value: True

   > The default value is False.

   After configuration the agent rediscovers the backend with the correct name.



2. Optionally delete the old database backend.

HTTP Exit Points for .NET

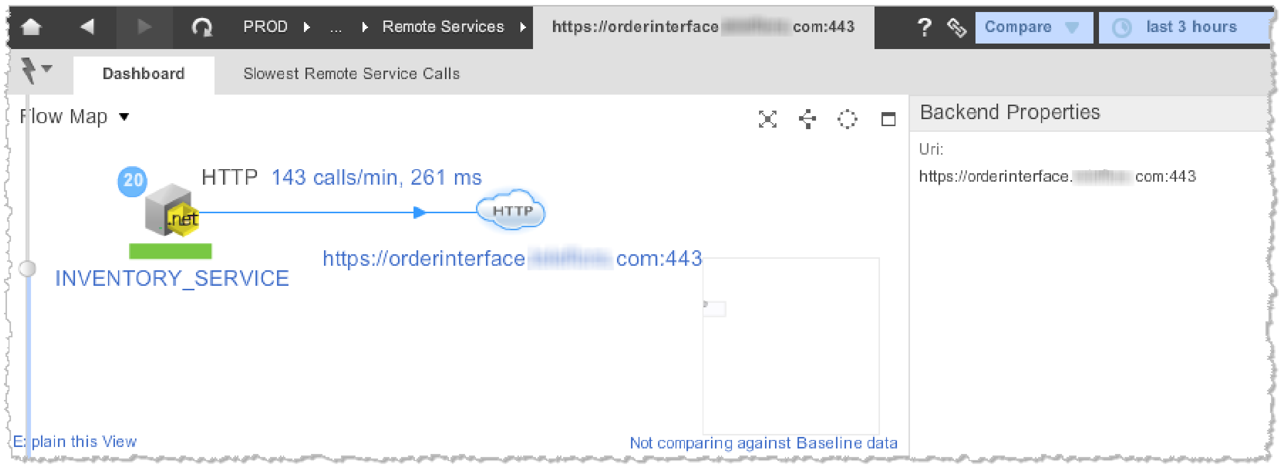
- Automatic Discovery and Default Naming
- HTTP Configurable Properties
- Changing the Default HTTP Automatic Discovery and Naming
- Learn More

This topic explains HTTP exit point configuration for .NET. To review general information about monitoring databases and remote services (collectively known as backends) and for an overview of backend configuration see Backend Monitoring.

Automatic Discovery and Default Naming

By default, AppDynamics automatically detects and identifies HTTP exit points (backends) that use the Microsoft HTTP client.

The default HTTP automatic discovery rule uses the URL property. From the enabled properties AppDynamics derives a display name using the URL, for example:


PROD
...
Remote Services
https://orderinterface com:443
Compare
last 3 hours
Dashboard
Slowest Remote Service Calls
Flow Map
HTTP 143 calls/min, 261 ms
20
.net
HTTP
INVENTORY_SERVICE
https://orderinterface com:443
Explain this View
Not comparing against Baseline data
Backend Properties
Uri:
https://orderinterface. com:443

By default, AppDynamics groups backends of the same type together on the application flow map.

HTTP Configurable Properties

You can enable or disable the use of the following properties for HTTP exit points.

Type	Configurable Properties	Property Used by Default in Detection and Naming
HTTP	Host	No
	Port	No
	URL	Yes
	Query String	No

Changing the Default HTTP Automatic Discovery and Naming

Depending on exactly what you need to monitor, there may be times when you want to change the default HTTP configuration. Review HTTP Exit Points (Java) for examples.
For configuration procedures, see Configure Backend Detection for .NET.

Learn More

- Backend Monitoring
- Monitor Remote Services
- Configure Custom Exit Points

Web Services Exit Points for .NET

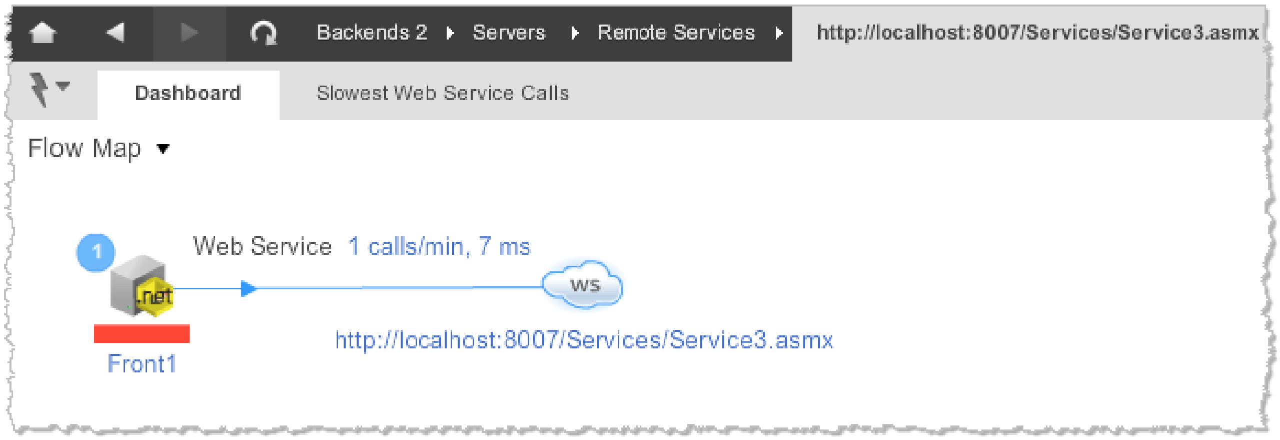
- Auto-Discovery and Default Naming
- Configurable Properties
- Changing the Default WCF Automatic Discovery and Naming
- Learn More

This topic explains Web Services exit point configuration for .NET. To review general information about monitoring databases and remote services (collectively known as backends) and for an overview of backend configuration see Backend Monitoring.

Auto-Discovery and Default Naming

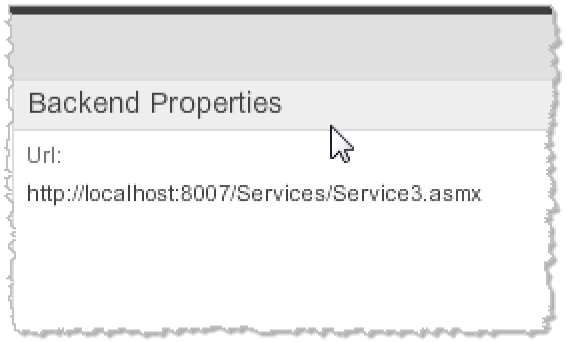
By default, AppDynamics automatically detects and identifies web services exit points (backends) when an application uses the Microsoft Web Services client library.

The default web services automatic discovery rule uses the URL property. From the enabled properties AppDynamics derives a display name using the URL, for example:


Backends 2
Servers
Remote Services
http://localhost:8007/Services/Service3.asmx
Dashboard
Slowest Web Service Calls
Flow Map
1
Web Service 1 calls/min, 7 ms
.net
ws
http://localhost:8007/Services/Service3.asmx
Front1

By default, AppDynamics groups multiple backends of the same type together on the application flow map.

The backend properties for the web services backend can be viewed on the Remote Services dashboard.


Backend Properties

Url:

http://localhost:8007/Services/Service3.asmx

Configurable Properties

Configurable Properties	**Property Used by Default in Detection and Naming**
Service	No
URL	Yes
Operation	No
Soap Action	No

Changing the Default WCF Automatic Discovery and Naming

Depending on exactly what you need to monitor, there may be times when you want to change the default configuration. In most cases, you can generate the right name and the correct number of backends to monitor by editing the automatic discovery rule.
For configuration procedures, see Configure Backend Detection for .NET.

Learn More

- Backend Monitoring
- Monitor Remote Services
- Configure Custom Exit Points

Configure Custom Exit Points for .NET

AppDynamics provides default automatic discovery for commonly-used backends. If a backend used in your environment is not discovered:

Compare the list of default backends.

- If the backend is listed, but the agent doesn't discover it, modify the default configuration OR
- If the backend isn't listed, then configure a custom exit point according to these instructions.

Default Backends Discovered by the Agent for .NET

For a list of supported backends see Remote Service Detection.

For instructions on modifying default backend discovery see Configure Backend Detection for .NET.

Configure Custom Exit Points for .NET Backends

Use custom exit points to identify backend types that are not automatically detected, such as file systems, mainframes, and so on. For example, you can define a custom exit point to monitor the file system read method. After you have defined a custom exit point, the backend appears on the flow map with the type-associated icon you selected when you configured the custom exit point.

You define a custom exit point by specifying the class and method used to identify the backend. If the method is overloaded, you need to add the parameters to identify the method uniquely.

You can restrict the method invocations for which you want AppDynamics to collect metrics by specifying match conditions for the method. The match conditions can be based on a parameter or the invoked object.

You can also optionally split the exit point based on a method parameter, the return value, or the invoked object.

You can also configure custom metrics and transaction snapshot data to collect for the backend.

To create a custom exit point

1. From the left navigation pane, click **Configure -> Instrumentation** and select the **Backend Detection** tab.

2. Select the application or tier for which you are configuring the custom exit point.

3. Ensure **Use Custom Configuration for this Tier** is selected.
Backend detection configuration is applied on a hierarchical inheritance model. See Hierarchical Configuration Model.

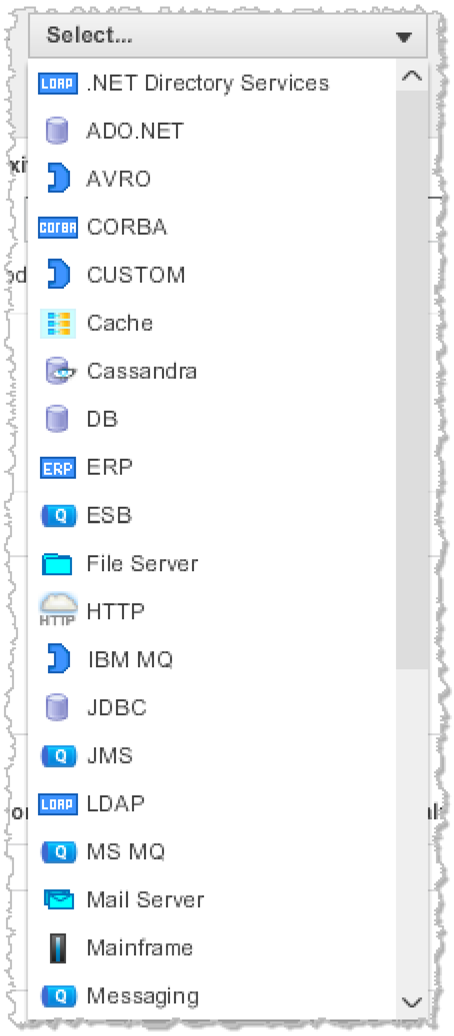
4. Scroll down to **Custom Exit Points** and click **Add** (the + icon).

5. In the **Create Custom Exit Point** window, click the **Identification** tab if it is not selected.

6. Enter a name for the exit point. This is the name that identifies the backend.

7. Select the type of backend from the **Type** drop-down menu.

This field controls the icon and name that appears on the flow maps and dashboards. Some of the values are shown in this screen shot:



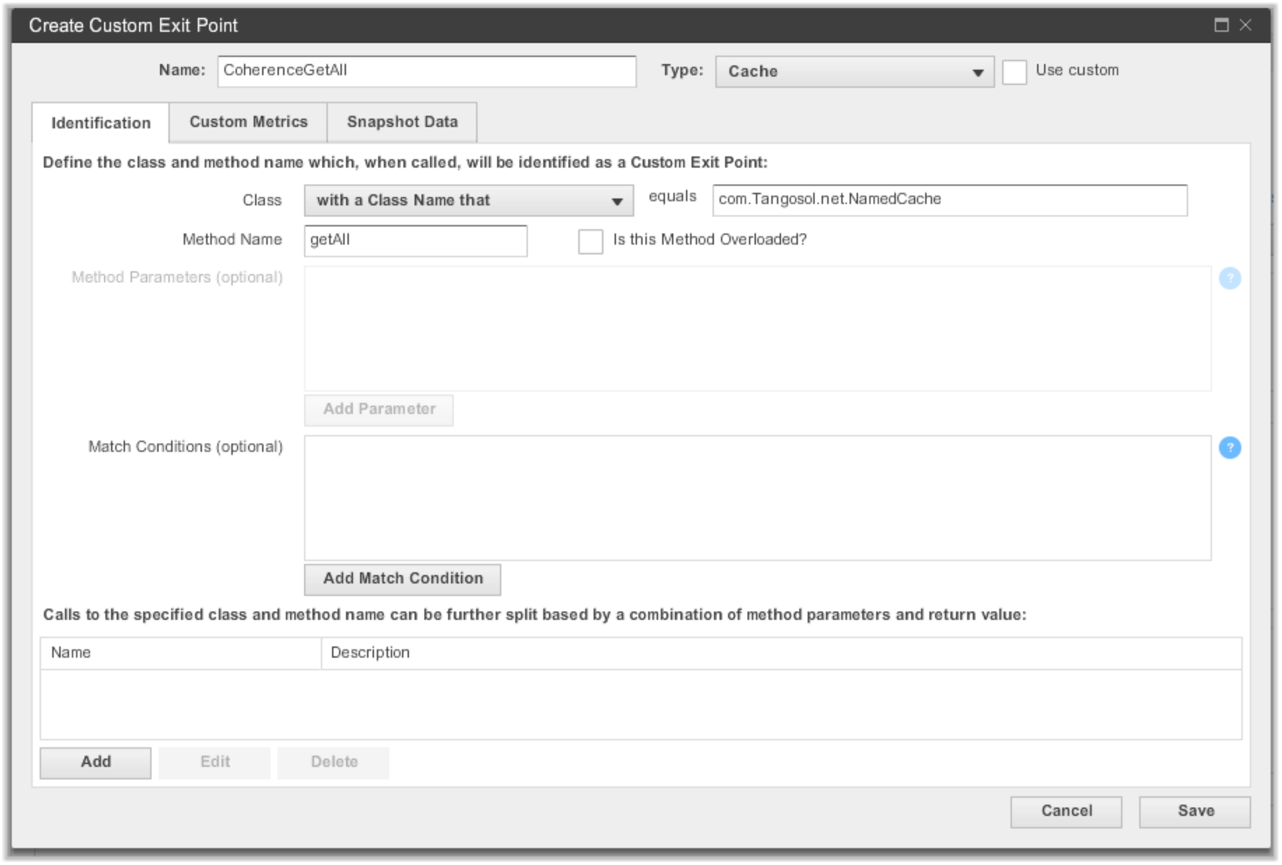
If the type is not listed, you can check **Use Custom** and enter a string to be used as the name on the dashboards.

8. Configure the class and method name that identify the custom exit point.
If the method is overloaded, check the Overloaded check box and add the parameters.

9. If you want to restrict metric collection based on a set of criteria that are evaluated at runtime, click **Add Match Condition** and define the match condition(s).
For example, you may want to collect data only if the value of a specific method parameter contains a certain value.

10. Click **Save**.

The following screenshot shows a custom exit point for a Cache type backend. This exit point is defined on the getAll() method of the specified class. The exit point appears in flow maps as an unresolved backend named CoherenceGetAll.



Name: CoherenceGetAll
Type: Cache
Use custom

Define the class and method name which, when called, will be identified as a Custom Exit Point:

Class: with a Class Name that equals com.Tangosol.net.NamedCache

Method Name: getAll

Is this Method Overloaded?

Method Parameters (optional)

Match Conditions (optional)

Calls to the specified class and method name can be further split based by a combination of method parameters and return value:

Name	Description

To split an exit point

1. In the Backend Detection configuration window, click **Add**.

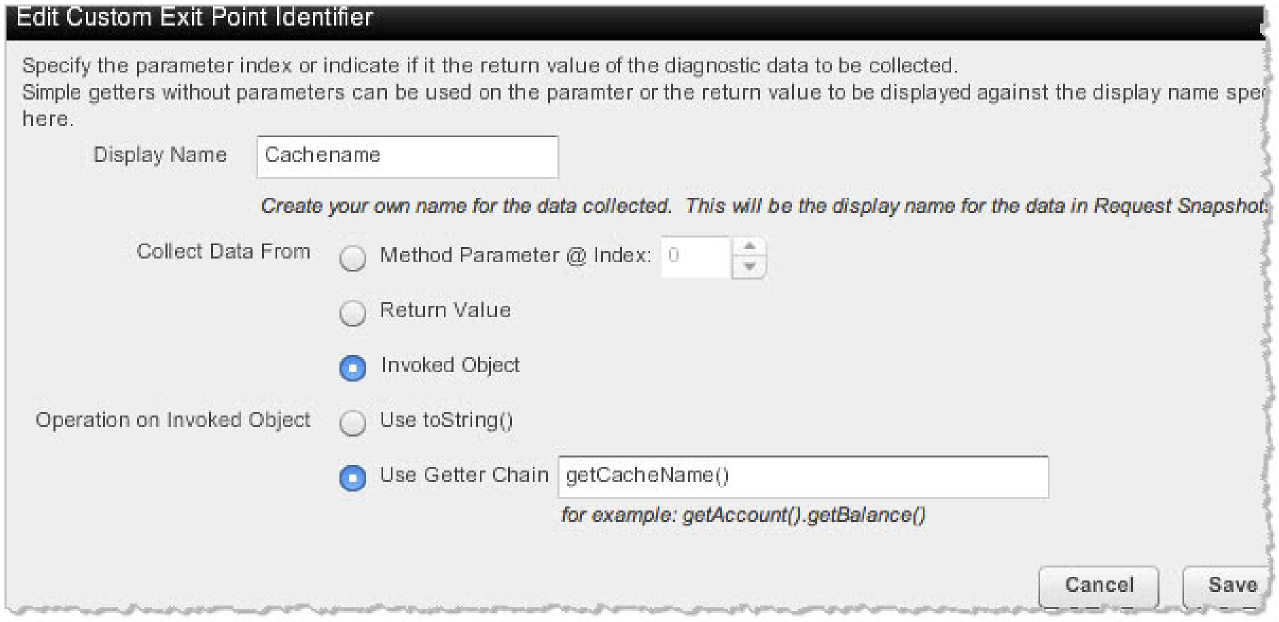
2. Enter a display name for the split exit point.

3. Specify the source of the data (parameter, return value, or invoked object).

4. Specify the operation to invoke on the source of the data: **Use toString()** or **Use Getter Chain** (for complex objects).

5. Click **Save**.

The following example shows a split configuration of the previously created CoherenceGetAll exit point based on the getCacheName() method of the invoked object.


Edit Custom Exit Point Identifier

Specify the parameter index or indicate if it the return value of the diagnostic data to be collected.
Simple getters without parameters can be used on the paramter or the return value to be displayed against the display name spe
here.

Display Name: Cachename

Create your own name for the data collected. This will be the display name for the data in Request Snapshot

Collect Data From: Method Parameter @ Index: 0 / Return Value / [✓ Invoked Object]

Operation on Invoked Object: Use toString() / [✓ Use Getter Chain] getCacheName()

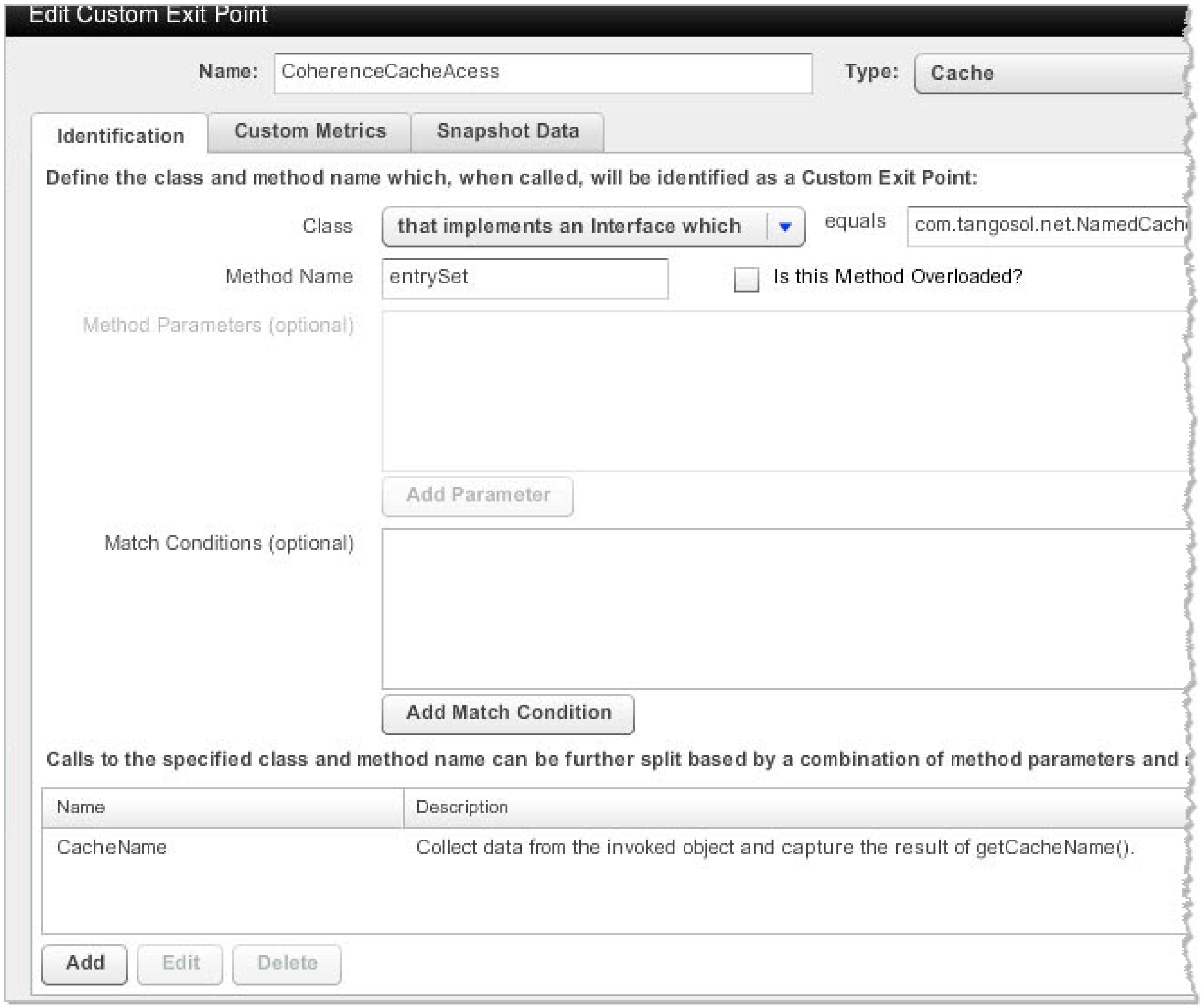
for example: getAccount().getBalance()

To group an exit point

You can group methods as a single exit point if the methods point to the same key.

For example, ACME Online has an exit point for NamedCache.getAll. This exit point has a split configuration of getCacheName() on the invoked object as illustrated in the previous screen shot.

Suppose we also define an exit point for NamedCache.entrySet. This is another exit point, but it has the split configuration that has getCacheName() method of the invoked object.


Edit Custom Exit Point

Name: CoherenceCacheAcess
Type: Cache

Identification | Custom Metrics | Snapshot Data

Define the class and method name which, when called, will be identified as a Custom Exit Point:

Class: that implements an Interface which equals com.tangosol.net.NamedCach
Method Name: entrySet
Is this Method Overloaded? [ ]
Method Parameters (optional)
Add Parameter
Match Conditions (optional)
Add Match Condition

Calls to the specified class and method name can be further split based by a combination of method parameters and

Name	Description
CacheName	Collect data from the invoked object and capture the result of getCacheName().

Add | Edit | Delete

If the getAll() and the entrySet() methods point to the same cache name, they will point to the same backend.

Matching name-value pairs identify the back-end. In this case, only one key, the cache name, has to match. So, here both exit points have the same name for the cache and they resolve to the same backend.

To define custom metrics for a custom exit point

Custom metrics are collected in addition to the standard metrics.

The result of the data collected from the method invocation must be an integer value, which is either averaged or added per minute, depending on your data roll-up selection.

To configure custom business metrics that can be generated from the Java method invocation:

1. Click the **Custom Metrics** tab.

2. Click **Add**.

3. In the **Add Custom Metric** window type a name for the metric.

4. Select **Collect Data From** to specify the source of the metric data.

5. Select **Operation on Method Parameter** to specify how the metric data is processed.

6. Select how the data should be rolled up (average or sum) from the Data Rollup drop-down menu.

7. Click **Create Custom Metric**.

To define transaction snapshot data collected

1. Click the **Snapshot Data** tab.

2. Click **Add**.

3. In the Add Snapshot Data window, enter a display name for the snapshot data.

4. Select the Collect Data From radio button to specify the source of the snapshot data.

5. Select the Operation on Method Parameter to specify how the snapshot data is processed.

5. Click **Save**.

Learn More

- Backend Monitoring
- Monitor Remote Services
- Monitor Databases

Configure Business Transaction Detection for .NET

This topic introduces .NET entry points and the way that they are used in business transaction detection. If the auto-discovered entry points don't include all your critical business transactions, use custom match rules to customize entry point discovery.

.NET Entry Points

AppDynamics detects entry points in the following places:

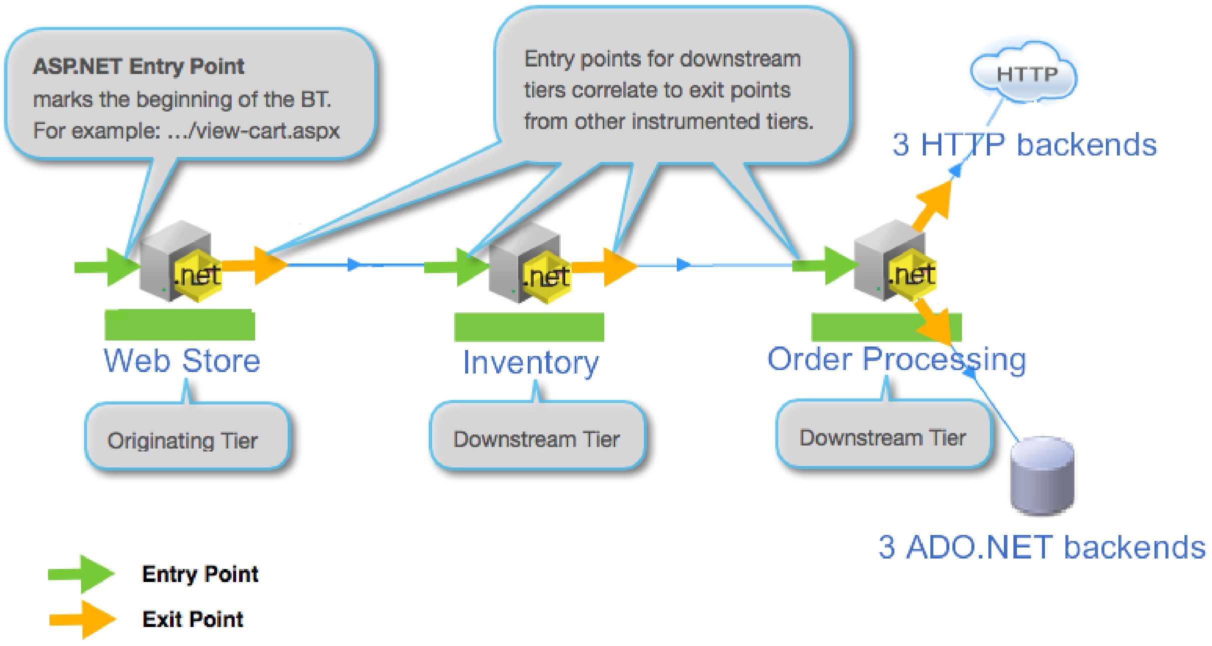
- On **originating tiers**, the method or operation that marks the beginning of a business transaction is an entry point. In most cases, this type of entry point maps to a user request or action such as "View/Cart". Entry points on originating tiers define the business transaction name.

- On **downstream tiers**, entry points correlate to incoming http calls, web service requests, and other communications from instrumented tiers.

The App Agent for .NET (agent) automatically detects entry points for the frameworks listed as automatically discovered business transactions. See Supported Environments and Versions (.NET).

If the agent detects an entry point on the originating tier, it registers a business transaction with the

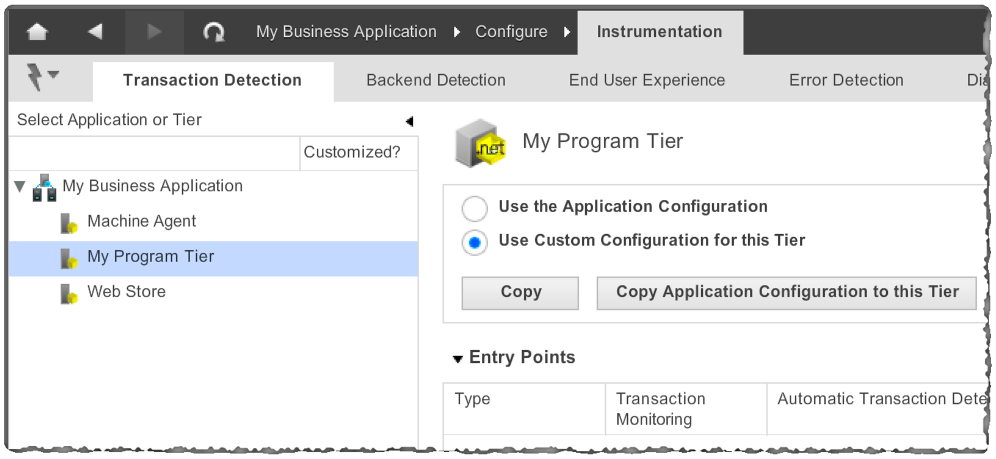
Controller. The Controller displays the business transaction in the Business Transactions List. If an entry point correlates to an upstream exit point, such as an exiting service call, the agent includes it as part of the existing business transaction.



If you need to monitor transactions not automatically discovered by the App Agent for .NET, you can customize entry point detection. See Organizing Traffic as Business Transactions to learn how to plan your business transactions. After you have identified the critical business transactions for your application, create custom match rules for them to enable transaction detection.

To create custom match rules for .NET entry points

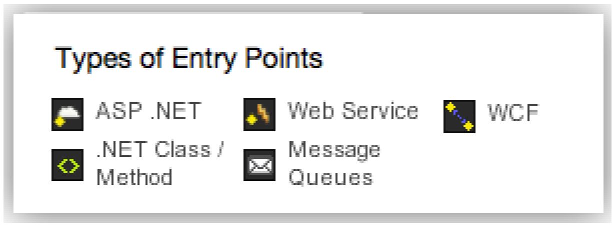
1. Click **Configure > Instrumentation > Transaction Detection**.
2. Click the **.NET - Transaction Detection** tab.
3. From the **Select Application or Tier** list at the left, click either:
    - an application, to configure transaction detection for all tiers in a business application.
    - a tier, to configure transaction detection at the tier level. At the tier level click **Use Custom Configuration for this Tier**. AppDynamics copies the application configuration to the tier level so that you can modify it for the tier.



( ) Use the Application Configuration
(●) Use Custom Configuration for this Tier

4. Use the **Custom Match Rules** pane to add and remove business transaction match rules.
For details on types of .NET entry points and how to setup
custom match rules, see:

- POCO (.NET Class/Method) Entry Points
- ASP.NET Entry Points



Learn More

- Web Entry Points
- Configure Business Transaction Detection
- Organizing Traffic as Business Transactions

POCO Entry Points

- Defining a POCO Entry Point
- Discovery of POCO Transactions
        - To specify a POCO custom match rule
    - POCO Transaction as a Background Task
- Learn More

Some applications use frameworks other than ones the App Agent for .NET auto-detects. This is frequently the case with Windows services and standalone applications. AppDynamics lets you specify entry points using custom match rules for Plain Old CLR Objects (POCOs). Once you've defined POCOs, we measure performance data for POCO transactions the same as for other transactions.

Define the custom match rule on the .NET class/method that is the most appropriate entry point for the business transaction (BT). Someone who is familiar with your application code should help make this determination.

Defining a POCO Entry Point

On an originating tier, a POCO entry point is the method that starts the BT. If the POCO entry point is on a downstream tier, it may correlate to an upstream exit point. When defining a POCO entry point, it is important to choose a method that begins and ends every time the BT executes. For more on entry points, see Business Transaction Monitoring.

Good candidates for POCO entry points include the following:

- A timer in a Windows service that executes a database call to check for new records to process. For example, an order processing system that periodically checks for new orders in the system.
- A loop in a standalone application that batch processes records via a web service call. For example, an expense reporting system that loops through approved expenses to submit them for reimbursement.
- A method in a web application that executes every time a client connects. For example, consider the method execution sequence:

```
using System.Net;
using System.Net.Sockets;
using System.Threading.Tasks;

namespace MyService
{
class MyProgram
{
    static void Main(string[] args)
    {
        TcpListener myList = new TcpListener(IPAddress.Parse("127.0.0.1"),
8000);
        using (Socket s = myList.AcceptSocket())
            Task.Factory.StartNew(DoWork, s);
    }
    static void DoWork<Socket>(Socket s)
    {
        // ...
    }
}
}
```

The `AcceptSocket()` method is the blocking method that accepts a job and invokes it. The `MyProgram.DoWork()` method is the unit of work because it executes every time a client calls the business transaction and it finishes at the same time as the business transaction. This makes `DoWork()` a good POCO entry point.

Discovery of POCO Transactions

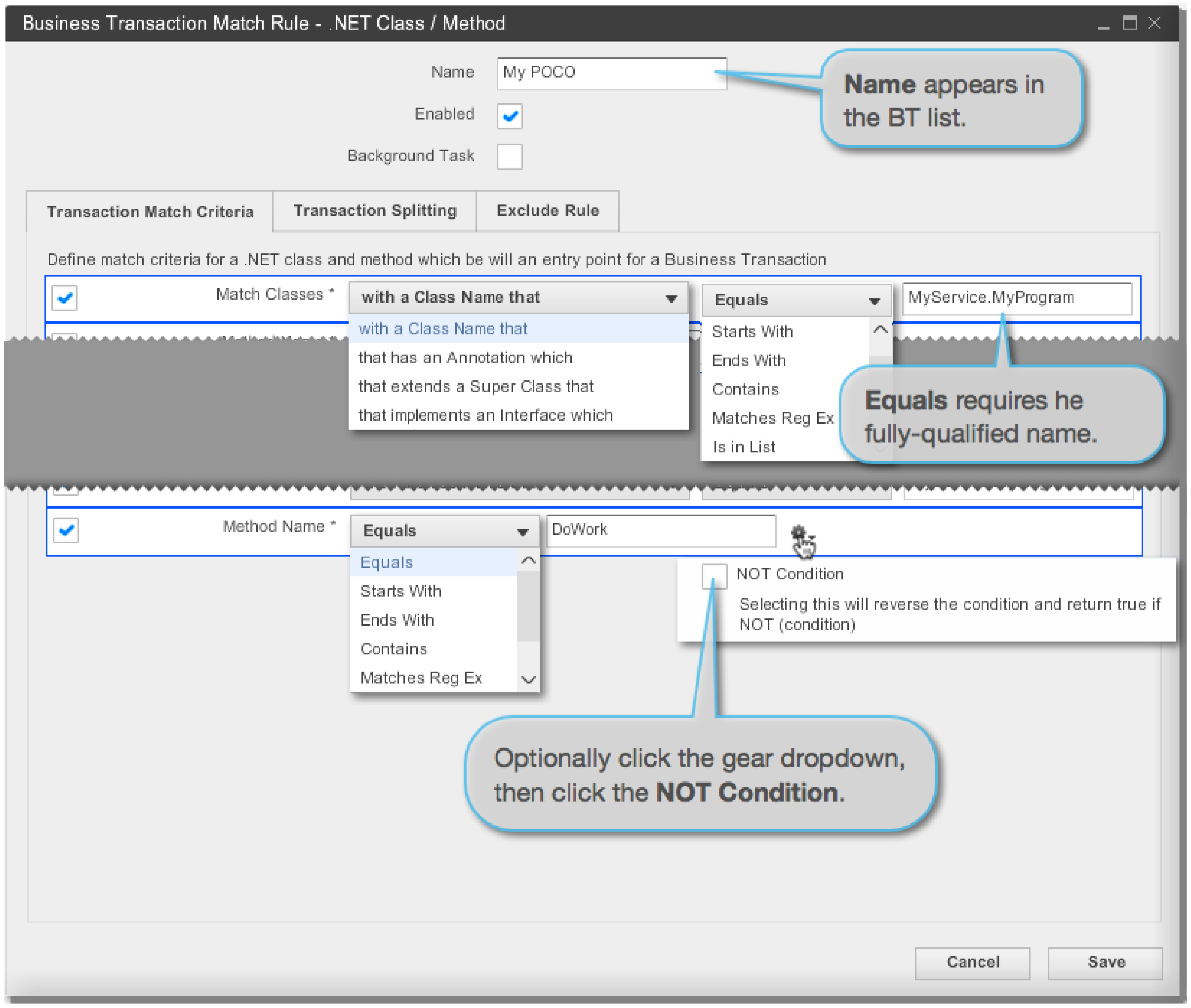
By default, once you configure a custom match rule for a POCO entry point, the App Agent for .NET detects and displays it in the Business Transactions List. AppDynamics names the BT for the name of the custom match rule. For more information, see Configure Business Transaction Detection.

To specify a POCO custom match rule

To set up a POCO entry point, define a custom match rule for a .NET Class/Method.

For steps to access the **Custom Match Rules** pane, see To create custom match rules for .NET entry points.

1. In the **Custom Match Rules** pane, click the plus symbol (**+**) to add an entry point.
2. Click **.NET Class/Method** in the dropdown list, then click **Next**.
3. Name the **New Business Transaction Match Rule**.
   - AppDynamics uses the rule **Name** to name the BT.
   - The Controller enables the rule by default. Disable it later if needed.
   - The POCO is a foreground task, to configure it as a background task, see POCO Transaction as a Background Task.
4. In the **Transaction Match Criteria** tab, specify the **Match Classes** criteria.
5. Specify the **Method Name** match criteria.


Business Transaction Match Rule - .NET Class / Method
Transaction Match Criteria
Transaction Splitting
Exclude Rule
with a Class Name that
that has an Annotation which
that extends a Super Class that
that implements an Interface which
Equals
Starts With
Ends With
Contains
Matches Reg Ex
Is in List
Name appears in the BT list.
Equals requires he fully-qualified name.
Optionally click the gear dropdown, then click the NOT Condition.
Cancel
Save

Name: My POCO
Enabled: ✓
Background Task: ☐

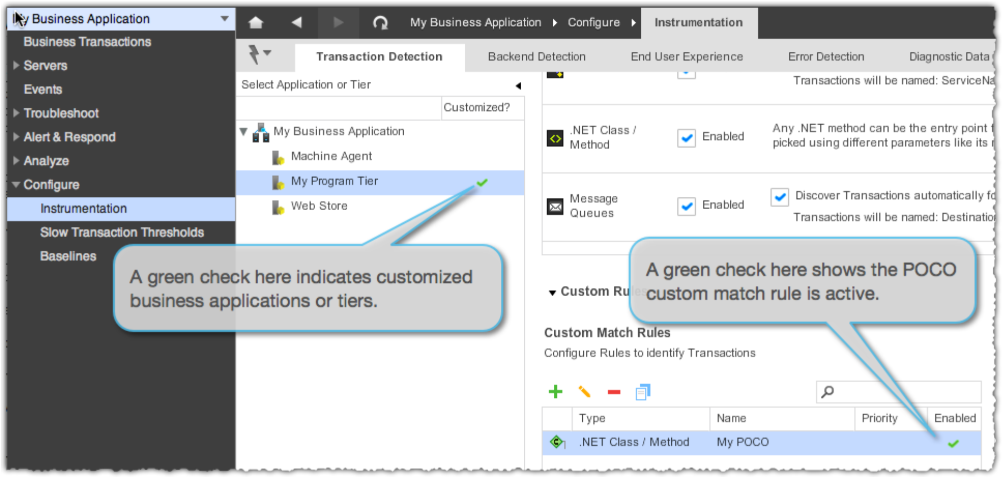
Define match criteria for a .NET class and method which be will an entry point for a Business Transaction

Match Classes *: with a Class Name that — Equals — MyService.MyProgram

Method Name *: Equals — DoWork

NOT Condition
Selecting this will reverse the condition and return true if NOT (condition)

6. Click **Save**. If you are configuring at the application level, click **Configure all Tiers to use this Configuration**.
7. Click **OK** to the notification message **Instrumentation changes require restart**.
   After you save the rule, it appears in the **Custom Match Rule** list. The business application or tier you customized displays a green check in the **Select Application or Tier** pane.



A green check here indicates customized business applications or tiers.

A green check here shows the POCO custom match rule is active.

Transactions will be named: ServiceNa

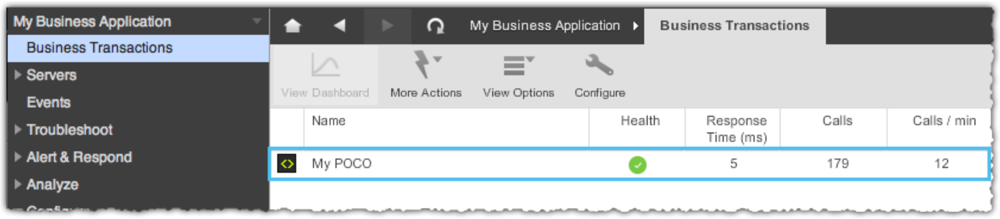
Any .NET method can be the entry point f picked using different parameters like its

Discover Transactions automatically fo Transactions will be named: Destination

Type	Name	Priority	Enabled
.NET Class / Method	My POCO		✓

8. Wait one minute and restart the CLR/application.

   The next time the POCO method executes, the agent detects it and registers the entry point. If the entry point is on an originating tier, the Controller displays it as a business transaction in the Business Transactions List.



Name	Health	Response Time (ms)	Calls	Calls / min
My POCO	●	5	179	12

   The agent identifies all the qualifying transactions using the custom match rule. In some situations you may need to further refine the discovery rules. Use the splitting options on the **T ransaction Splitting** tab.

POCO Transaction as a Background Task

Click the **Background Task** check box in the **Business Transaction Match Rule** window to indicate that a POCO transaction runs in the background.

When a request runs as a background task, AppDynamics reports only Business Transaction metrics for the request. It does not aggregate response time and call metrics at the tier and application levels for background tasks. This prevents background tasks from distorting baselines for the business application. Also, you can set a separate set of thresholds for background tasks. See Background Task Monitoring.

Learn More

Background Task Monitoring
Configure Business Transaction Detection
Business Transactions List

ASP.NET Entry Points

AppDynamics automatically detects entry points for client requests to ASP.NET applications. If the request occurs on an originating tier, the method or operation marks the beginning of a business transaction and defines the transaction name. In most cases, this type of entry point maps to a user request or action such as "Cart/Checkout". AppDynamics allows you to configure transaction naming based upon the ASP.NET request.

Default Automatic Naming for ASP.NET Transactions

By default, the AppDynamics auto-detection naming scheme identifies all ASP.NET transactions using the first two segments of the URI.

For example, the following URI represents the checkout operation in an online store:

```
http://mydotnetsite.com/Cart/Checkout
```

AppDynamics automatically names the transaction:

```
/Cart/Checkout
```

For another example, the following URI represents a funds transfer operation in an online bank:

```
http://webbank.mybank.com/Account/Transferfunds/NorthernCalifornia
```

AppDynamics automatically names the transaction:

```
/Account/Transferfunds
```

Customize the Automatic Naming Scheme

The AppDynamics auto-detected transaction names might not be optimal for your users. You can configure the naming scheme as follows:

- Identify transactions using URI segments
- Identify transactions using headers, cookies, and other parts of HTTP requests

To modify automatic naming

1. Click **Configure > Instrumentation > Transaction Detection**.

2. Click the **.NET - Transaction Detection** tab.

3. From the **Select Application or Tier** list at the left, click either:
   - an application to configure transaction detection for all tiers in a business application.
   - a tier. At the tier level click **Use Custom Configuration for this Tier**. AppDynamics copies the application configuration to the tier level so that you can modify it for the tier.

4. If necessary, click **Enabled** under Transaction Monitoring and **Discover Transactions automatically for ASP.NET requests**.
   You can configure naming with Discover Transactions automatically for ASP.NET requests disabled, but the agent doesn't discover ASP.NET transactions.

5. Click **Configure Naming** for the ASP.NET type in the in the Entry Points panel.

6. Change the naming scheme in the ASP.NET Transaction Naming Configuration window and click **Save**.

The following sections provide examples to help you decide how to configure the naming scheme.

Identify Transactions Using URI Segments

AppDynamics offers the following options to automatically name ASP.NET transactions based upon the URI:

- Use all, first, or last URI segments
- Use specific URI segments

To name transactions using all, first, or last URI segments

Consider the following URL that represents the checkout operation in an online store:

```
http://mydotnetsite.com/Web/Store/Checkout
```

The first two segments of the URI don't provide a significant name for the business transaction:

```
/Web/Store
```

Identify a more meaningful name using one of the following options:

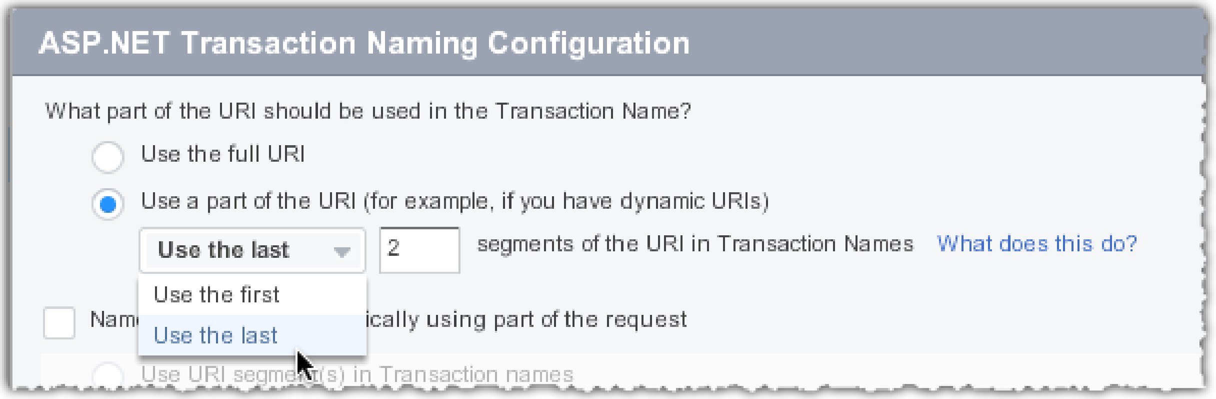
- Click **Use the full URI** to identify the transaction by all URI segments. For example:

```
/Web/Store/Checkout
```

- Click **Use the first** or **Use the last** n segments to use two contiguous segments at the beginning or end of the URI, where n is the number of segments.

  For example, to identify the checkout transaction using the last two segments of the URI:

  ```
  /Store/Checkout
  ```



ASP.NET Transaction Naming Configuration

What part of the URI should be used in the Transaction Name?

Use the full URI

Use a part of the URI (for example, if you have dynamic URIs)

Use the last: 2 segments of the URI in Transaction Names What does this do?

Use the first
Use the last
Nam
ically using part of the request
Use URI segment(s) in Transaction names

- If you need more flexibility, such as using non-contiguous segments in the name, click **Name Transactions dynamically using part of the requests** and specify the segments with the **Use URI segments in Transaction names** option.

To use specific URI segments in transaction names

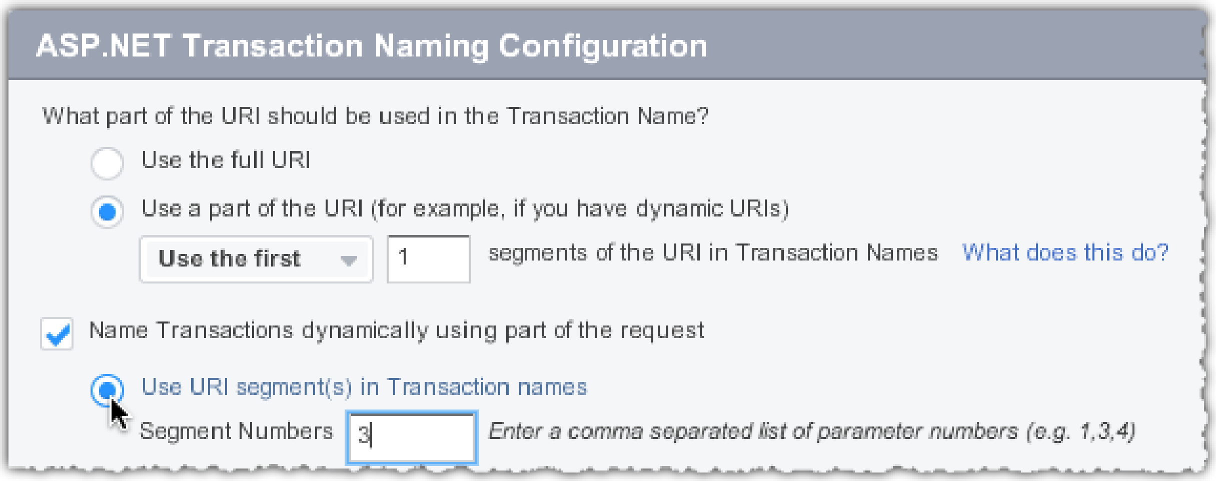
You can choose specific URI segments to use in the transaction name. This enables you to skip URI segments or use non-contiguous segments in the naming scheme.

1. Click **Use a part of the URI**.
2. Enter the number of first or last segments to use.
3. Click **Name Transactions dynamically using part of the request.**
4. Click **Use URI segment(s) in Transaction names**.
5. Enter the segment numbers separated by commas.

   For example the following URL represents the checkout transaction requested by a customer with ID 1234:

   ```
   http://mydotnetsite.com/Store/cust1234/Checkout
   ```

   The checkout transaction is the same regardless of the customer, so it makes sense to name the transaction based upon the first and third segments of the URI.


ASP.NET Transaction Naming Configuration

What part of the URI should be used in the Transaction Name?

( ) Use the full URI

(●) Use a part of the URI (for example, if you have dynamic URIs)

Use the first | 1 | segments of the URI in Transaction Names What does this do?

[✓] Name Transactions dynamically using part of the request

(●) Use URI segment(s) in Transaction names

Segment Numbers | 3 | *Enter a comma separated list of parameter numbers (e.g. 1,3,4)*

AppDynamics names the transaction:

`/Store/Checkout`

Identify Transactions Using Headers, Cookies, and Other Parts of HTTP Requests

You can also name ASP.NET transactions using parameters, headers, cookies, and other parts of HTTP requests.

To identify all your ASP.NET transactions using particular parts of the HTTP request, use the Name **Transactions dynamically using part of the request** option.

Carefully consider your naming configuration choices. If you use a value such as the request originating address and you have many clients accessing your application, you may see the All Other Traffic Business Transaction.

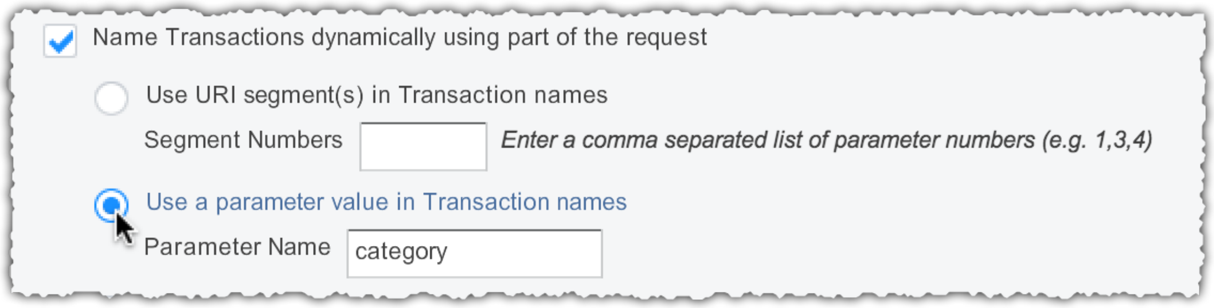
To use HTTP parameter values in transaction names

1. Set the URI identification option.

2. Click **Use a parameter value in Transaction names** and enter the **Parameter Name**.

   For example, consider the following URL:

   `http://mydotnetsite.com/Store/Inventory?category=electronics`


[✓] Name Transactions dynamically using part of the request

( ) Use URI segment(s) in Transaction names

Segment Numbers | | *Enter a comma separated list of parameter numbers (e.g. 1,3,4)*

(●) Use a parameter value in Transaction names

Parameter Name | category

   AppDynamics names the transaction to include the category parameter value:

   `/Store/Inventory.electronics`

To use a header value in transaction names

1. Set the URI identification option.

2. Click **Use header value in transaction names** and enter a **Header Name.**
   For example, consider a site that uses the custom header "Version", AppDynamics names transactions with the header value as follows:

   ```
   /Store/Inventory.v2.5
   ```

To use a cookie value in transaction names

1. Set the URI identification option.

2. Click **Use a cookie value in Transaction names** and enter the **Cookie Name**.
   For example, a website tracks a user's loyalty status in a cookie. Set the Cookie Name to "loyalty". AppDynamics names transactions for the loyalty cookie value:

   ```
   /Store/Inventory.Status=Gold
   ```

To use a session attribute value in transaction names

1. Set the URI identification option.

2. Click **Use a session attribute in Transaction names** and enter the **Session Attribute Key**
   .
   For example, a website stores a customer's region in the session property. Set the Session Attribute name to "region". AppDynamics names transactions for the region session attribute value:

   ```
    /Store/Inventory.NorthAmerica
   ```

To use the request method in Transaction names

1. Set the URI identification option.

2. Click **Use the request method (GET/POST/PUT) in Transaction names**.
   AppDynamics names transactions for the request method. For example:

   ```
   /Store/Inventory.GET
   ```

To use the request host in Transaction names

1. Set the URI identification option.

2. Click **Use the request host in Transaction names**.
   AppDynamics names transactions for the ip address of the request host. For example:

   ```
   /Store/Inventory.192.0.2.0
   ```

To use the request originating address in Transaction names

1. Set the URI identification option.

2. Click **Use the request originating address in Transaction names**.
   AppDynamics names transactions for the ip address of the request client. For example:

   ```
   /Store/Inventory.192.0.2.10
   ```

To use a custom expression on the HttpRequest

New in 3.8.4, custom expressions enable you to name transactions using getter chain(s) for HttpRequest properties and methods.

1. Set the URI identification option.
2. Click **Apply a custom expression on HttpRequest and use the result in Transaction Names**.
3. Enter your custom expression getter chain as follows:
   - Enclose getter chain(s) inside braces: `${}` .
   - Use getter chain syntax.
   - Use any HttpRequest request attributes or methods.

   For example, consider this URL:

   ```
   http://mystore.example.com/Store/Inventory-Furniture
   ```

   The following custom expression uses two getter chains:
   - The first getter chain fetches the URL, splits it on the dash character ("-"), and uses the second string in the array.
   - The second getter chain fetches the `HttpRequest.UserAgent` property.
   - The literal dash character "-" separates the two getter chains.

   ```
   ${Url.ToString().Split(Char[]/-).[2]}-${UserAgent}
   ```

   The result is the following business transaction name:

   ```
   Furniture-Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like
   Gecko
   ```

Custom Match Rules for ASP.NET Transactions

Custom match rules provide greater flexibility for transaction naming. When you define a match rule, AppDynamics uses the rule name for the business transaction name.

For steps to access the **Custom Match Rules** pane, see To create custom match rules for. NET entry points.

To create an ASP.NET custom match rule

1. In the **Custom Match Rules** pane, click the plus symbol (**+**) to add an entry point.

2. Click **ASP.NET** in the dropdown list. Click **Next**.

3. Name the **New Business Transaction Match Rule**.
    - AppDynamics uses the rule **Name** to name the BT.
    - The Controller enables the rule by default. Disable it later if needed.
    - Set the **Priority** for the match rule. AppDynamics applies higher priority rules first.
4. Set one or more of the following match criteria. When AppDynamics detects a requests matching your specified criteria, it identifies the request using your custom name.

    Method: Match on the HTTP request method, GET, POST, PUT or DELETE.

    WIth automatic discovery for ASP.NET transactions enabled, configuring the match on GET or POST causes the the agent to discover both GET and POST requests. If you only want either GET or POST requests for the transaction, consider the following options:

    - Disable automatic discovery for ASP.NET transactions.
    - Create an exclude rule for the method you don’t want: GET or POST.

    URI: Set the conditions to match for the URI.
    - For rules on regular expressions for .NET, see .NET Framework Regular Expressions.
    - Optionally click the gear icon to set a NOT condition.
    - You must set an URI match condition in order to use transaction splitting.

    HTTP Parameter: Match on HTTP parameter existence or a specific HTTP parameter value.

    Header: Match on a specific HTTP header's (parameter's) existence or a specific HTTP header value.

    Hostname: Match on the server host name. Optionally click the gear icon to set a NOT condition.

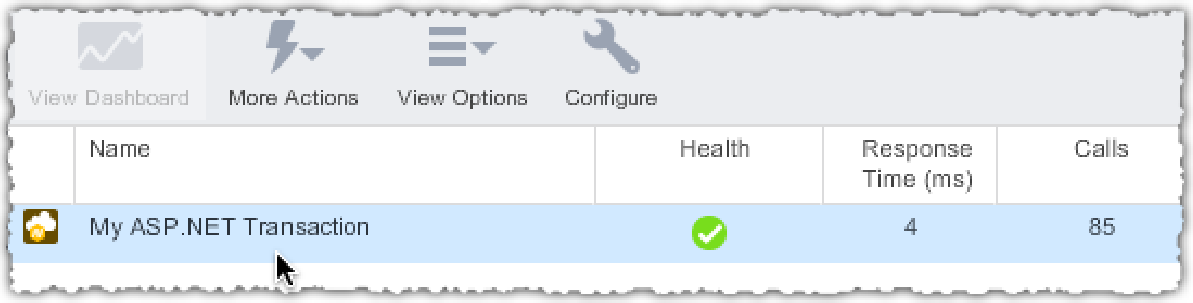
    Port: Match on the server port number. Optionally click the gear icon to set a NOT condition.

    Class Name: Match on the ASP.NET class name. Optionally click the gear icon to set a NOT condition.

    Cookie: Match on cookie existence or a specific a specific cookie value.

5. Click **Save**.

    The rule appears in the **Custom Match Rule** list. The business application or tier you customized displays a green check in the **Select Application or Tier** pane.
    After the agent receives the updated configuration, it discovers the new business transaction and displays it in the Business Transactions List.



	Name	Health	Response Time (ms)	Calls
	My ASP.NET Transaction	✓	4	85

To split custom ASP.NET transactions

AppDynamics lets you further refine ASP.NET custom transaction names using transaction splitting. See Transaction Splitting for Dynamic Discovery.

1. Create a custom match rule. To use transaction splitting, you must specify URI match criteria.

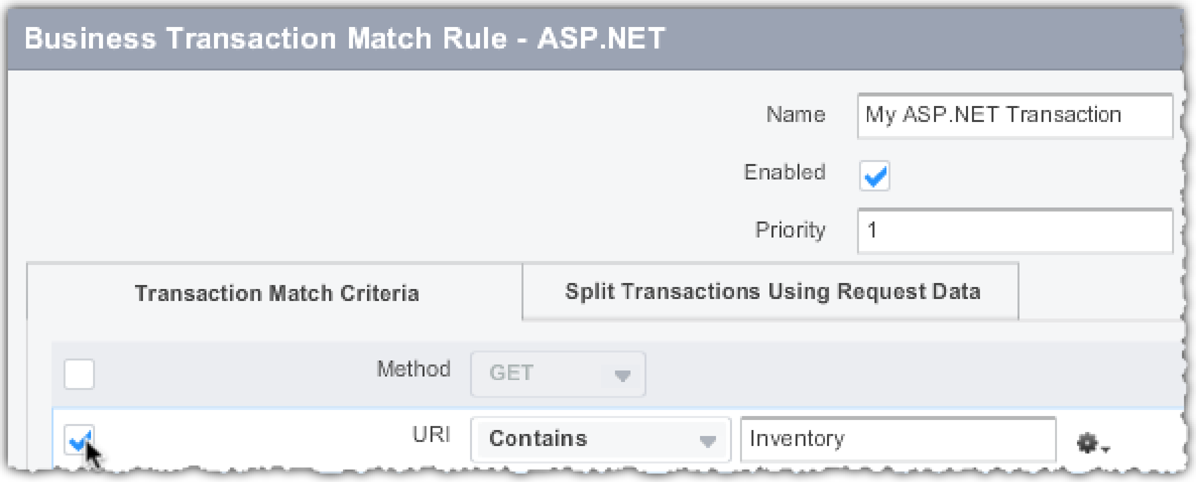
2. Click **Split Transactions Using Request Data**.

3. Click the splitting option to use.

   The transaction splitting options work the same as the methods described in the previous sections:
   Identify transactions using URI segments
   Identify Transactions Using Headers, Cookies, and Other Parts of HTTP Requests

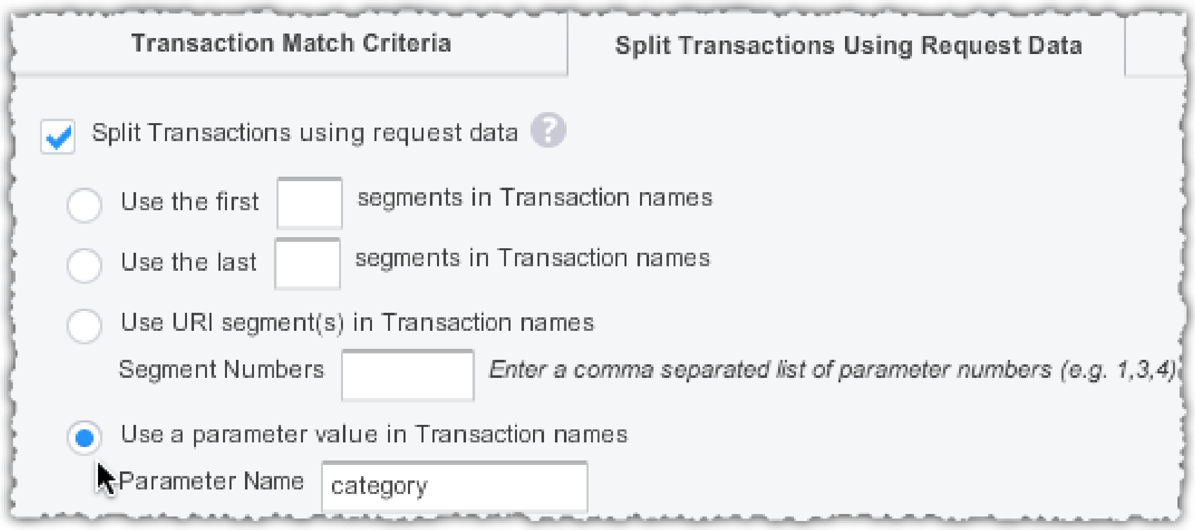
   For example, consider the following URL:
   `http://mydotnetsite.com/Store/Inventory?category=electronics`

   Configure the custom match rule to match on the "URI contains Inventory".


Business Transaction Match Rule - ASP.NET
Transaction Match Criteria
Split Transactions Using Request Data

Name	My ASP.NET Transaction
Enabled	✓
Priority	1
Method	GET
✓ URI	Contains Inventory

Split the transaction on the category parameter.



Transaction Match Criteria | Split Transactions Using Request Data

[✓] Split Transactions using request data

( ) Use the first [ ] segments in Transaction names

( ) Use the last [ ] segments in Transaction names

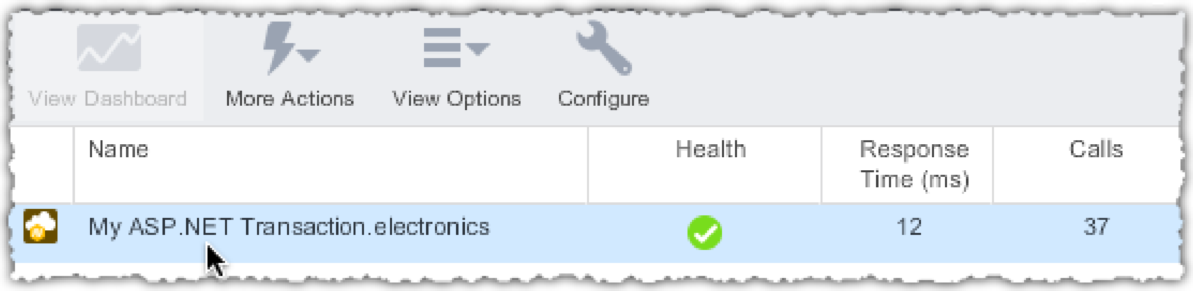
( ) Use URI segment(s) in Transaction names

Segment Numbers [ ] *Enter a comma separated list of parameter numbers (e.g. 1,3,4)*

(●) Use a parameter value in Transaction names

Parameter Name: category

4. Click **Save**.
   After the agent receives the updated configuration, it discovers the new business transaction and displays it in the Business Transactions List.



Name	Health	Response Time (ms)	Calls
My ASP.NET Transaction.electronics	✓	12	37

Learn More

Configure Business Transaction Detection for .NET

Web Entry Points

Configure Business Transaction Detection

Import and Export Transaction Detection Configuration for .NET

- Import and Export Auto-Detected Entry Point Configurations
  - Export guidelines
  - Import guidelines
  - To import or export the configurations for all the auto-detected entry-points to or from an application
  - To import or export the configuration for a single auto-detected entry point type to or from an application
    - Entry point type names
  - To import or export the configurations for all the auto-detected entry-points to or from

New in 3.8.4, AppDynamics lets you migrate transaction detection configurations for .NET entry points from one application to another using a REST API. Use the API to copy transaction detection configurations rather than manually re-configuring multiple applications in the Controller.

Import and Export Auto-Detected Entry Point Configurations

You can import and export all your entry point configurations or one entry point configuration in a single request. Lists of multiple entry point names are not supported. Import to or export from application-level configuration and tier-level configuration as follows:

- auto-detected entry point configurations
- custom match rules
- exclude rules

Export guidelines

Follow these rules when exporting entry point configurations:

- Use the HTTP GET method.
- Encode the URI using UTF-8 URL encoding.
- The base URI is the source application and, optionally, the source tier for your configuration export.

```
http://<controller host>:<controller
port>/controller/transactiondetection/<application
name>/<optional tier name>
```

- The Controller exports the configurations to an XML file. If necessary, you can edit the XML file before you import it.
  For example, if you export the all the auto-detected entry points but don't want to import them all, delete the ones you do not want from the file before import.

Import guidelines

Follow these rules when importing entry point configurations:

- Export the auto-detected entry point configuration, custom match rule, or exclude rule from the Controller.

ℹ You may manually create the XML, but that option is more difficult.

- Use the HTTP POST method.
- Encode the URI using UTF-8 URL encoding.
- The base URI is the destination application and, optionally, the destination tier for your configuration import.

```
http://<controller host>:<controller
port>/controller/transactiondetection/<application
name>/<optional tier name>
```

- Include the XML configuration as a file attachment to the request.
- To overwrite an existing configuration with the same name see Overwrite Parameter.
- A successful import request returns HTTP status code 200.

To import or export the configurations for all the auto-detected entry-points to or from an application

`http://<controller host>:<controller port>/controller/transactiondetection/<application name>/auto`

exports all the auto-detected entry point types for all agents: ASP.NET, Java, PHP, NodeJS, etc.

For example:

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/auto
```

produces the output in auto_all.xml.

To import or export the configuration for a single auto-detected entry point type to or from an application

`http://<controller-host>:<controller-port>/controller/transactiondetection/<application name>/auto/<entry point type name>`

Entry point types names are case insensitive.

In this example the customized ASP.NET naming configuration uses one segment of the URI and the "destination" parameter, the API

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/auto/aspdotnet
```

produces the output in auto_aspdotnet.xml.

Entry point type names

When you import or export a specific entry point type, the REST API requires the following entry point type names:

ASP.NET: `AspDotNet`
Web Service: `DotNetWebService`

WCF: `WCF`
.NET Class / Method: `POCO`
Message Queue: `DotNetJms`
.NET Remoting: `DotNetRemoting`

To import or export the configurations for all the auto-detected entry-points to or from a tier

```
http://<controller-host>:<controller-port>/controller/transactiondetecti
on/<application name>/<tier name>/auto
```

For example:

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/TravelSearch+Server/auto/
```

The tier output is the same as for an application. See auto_all.xml.

To import or export the configuration for a single auto-detected entry point type to or from a tier

```
http://<controller host>:<controller
port>/controller/transactiondetection/<application name>/<tier
name>/auto/<entry point type name>
```

For example if you disabled Web Services for the TravelSearch Server tier, the API

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/TravelSearch+Server/auto/DotNetWebService
```

produces the output in auto_webservice.xml.

Import and Export Custom Match Rules

To import or export a single custom match rule to or from an application

```
http://<controller host>:<controller
port>/controller/transactiondetection/<application name>/custom/<entry
point type name>/<custom rule name>
```

For example to export a POCO, the API

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/custom/POCO/My+POCO
```

produces the output in custom_poco.xml.

To import or export a single custom match rule to or from a tier

```
http://<controller host>:<controller
port>/controller/transactiondetection/<application name>/<tier name>/cus
```

`tom/<entry point type name>/<custom rule name>`

For example:

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/TravelSearch+Server/custom/AspDotNet/My+ASP.NET+Tra
nsaction
```

produces the output in custom_aspdotnet.xml.

Import and Export Exclude Rules

To import or export a single exclude rule to or from an application

`http://<controller host>:<controller port>/controller/transactiondetection/<application name>/exclude/<entry point type name>/<exclude rule name>`

For example, to export an exclude rule for a specific service, the API

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/exclude/DotNetWebService/My+Web+Service+Exclude+Rule
```

produces the output in exclude_webservice.xml.

To import or export a single exclude rule to or from a tier

`http://<controller host>:<controller port>/controller/transactiondetection/<application name>/<tier name>/exclude/<entry point type name>/<exclude rule name>`

For example:

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/TravelSearch+Server/exclude/DotNetWebService/My+Web
+Service+Exclude+Rule
```

The tier output is the same as for an application. See exclude_webservice.xml.

Overwrite Parameter

Use the overwrite parameter to overwrite a configuration of the same name. Without this parameter, if the import encounters a configuration for a component of the same name, the request will fail.

For example, to import a configuration for a POCO custom match rule named "My POCO" to an application that has an existing "My POCO" custom match rule use:

```
http://appdcontroller.example.com/controller/transactiondetection/Ho
wdy+World+Travel/custom/POCO/My+POCO?overwrite=true
```

The default is overwrite=false.

Learn More

- Configure Business Transaction Detection
- Use the AppDynamics REST API

Identify MVC Transactions by Controller and Action

*Y*ou can configure the agent to identify transactions by MVC Controller/Action instead of the default naming by URI. For general information on organizing and naming business transactions, see Organizing Traffic as Business Transactions.

Identifying MVC Transactions by Controller and Action

By default the App Agent for .NET (agent) identifies ASP.NET MVC business transactions by the request URL or server URI.

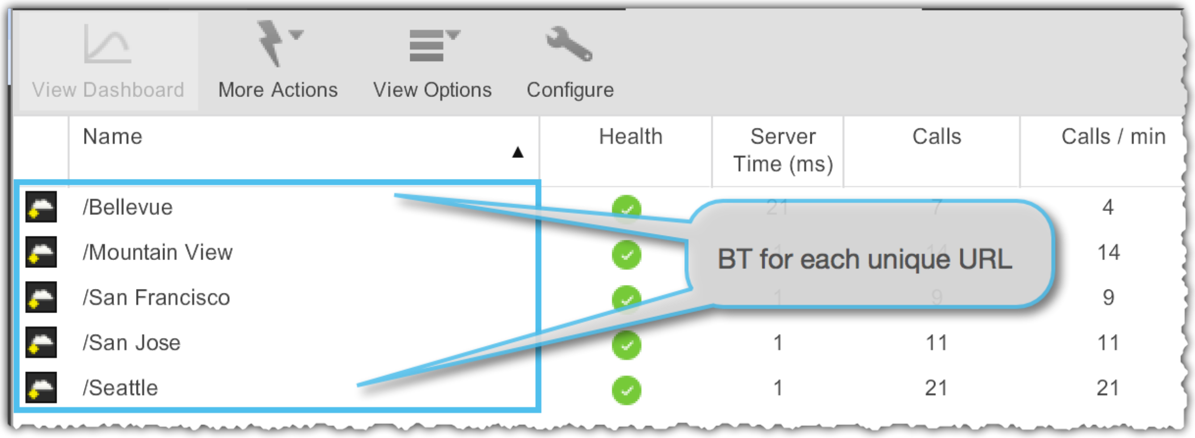
In cases where an application accepts numerous client URLs for a single MVC controller and action, naming business transactions with the client URL can cause several issues including the following:

- The number of business transactions exceeds the limits. See Business Transaction Limits.
- Most requests wind up in "All other traffic". See All Other Traffic Business Transaction.
- Requests per minute per business transaction is inconsistent.

For example, consider a MVC application that lists store locations. City-specific URLs list the locations for a city:

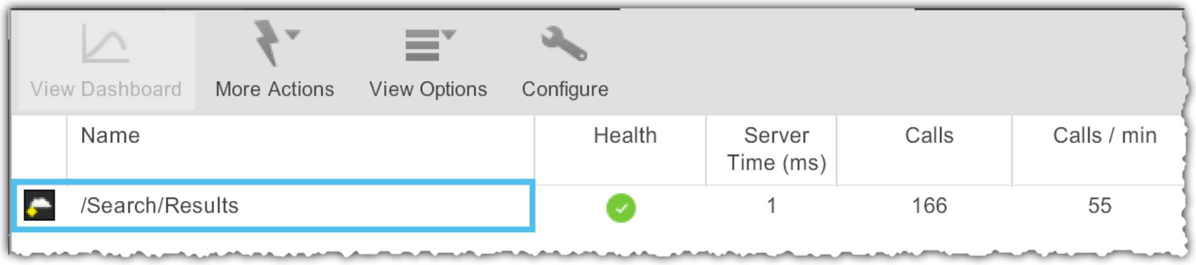
http://myapp.mycompany.com/Bellevue

The business transaction name for this URL defaults to **/Bellevue**. Each request for a unique city generates a business transactions. None of the URIs contain common elements that you can use to configure business transaction names.



	Name	Health	Server Time (ms)	Calls	Calls / min
	/Bellevue	✓	21	7	4
	/Mountain View	✓	1	14	14
	/San Francisco	✓	1	9	9
	/San Jose	✓	1	11	11
	/Seattle	✓	1	21	21

In the web application, all city location searches, such as /Bellevue, map to the *Results* action of the *Search* controller. After you configure the agent to name transactions by controller and action, the agent identifies the business transaction as **Search/Results**.



	Name	Health	Server Time (ms)	Calls	Calls / min
	/Search/Results	✓	1	166	55

The Search/Results business transaction combines search requests for all cities into one transaction.

To identify ASP.NET MVC transactions by Controller/Action

To configure the agent to identify MVC transactions as Controller/Action, register the **aspdotnet-mvc-naming-controlleraction** node property. The node property works for MVC 3, MVC 4 and WebAPI transactions.

For instructions on how to set a node property, see App Agent Node Properties.

Name: aspdotnet-mvc-naming-controlleraction
Description: Identfy ASP.NET MVC Business Transactions as Controller/Action.
Type: Boolean
Value: True

ℹ The default value is **False**.

After the agent registers traffic with a business transaction named for the Controller/Action and after traffic to the old business transactions named for the client URL diminishes, delete the old business transactions.

Business Transaction Naming Convention

New in 3.8.4, if you use Areas to organize your MVC application, the agent includes the Area name in the business transaction name:

```
/Area name/Controller name/Action name
```

For example, if your travel application has separate Areas for hotel, airfare, and car rentals:

/Hotel/Search/Results

Otherwise the agent names the transaction as follows:

```
/Controller name/Action name
```

Learn More

Business Transaction Limit
All Other Traffic Business Transaction
App Agent Node Properties
Configure Business Transaction Detection
ASP.NET Routing

Configure Application Domain Monitoring

*Y*ou can configure monitoring for ASP.NET applications with multiple Application Domains (AppDomains). This topic assumes you have a working knowledge of AppDomains and that you are familiar with the AppDomain implementation in your application.

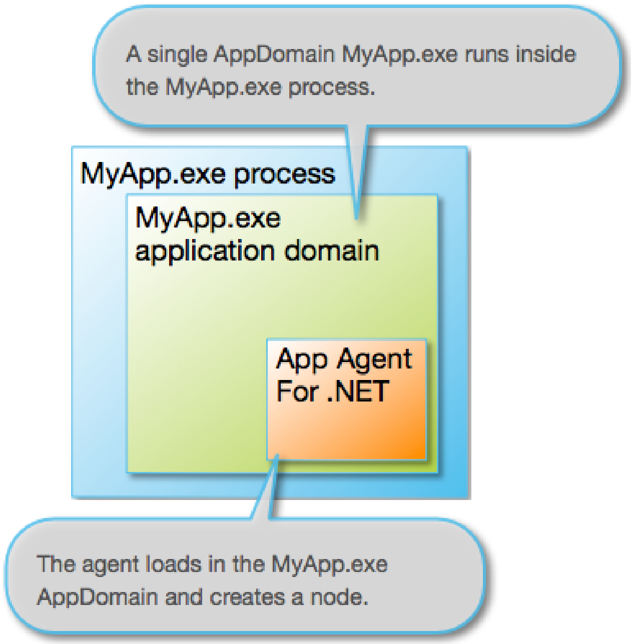
This topic does not cover the System Domain, Shared Domain, or DefaultDomain AppDomains the CLR instantiates before it executes the managed code. If your standalone application runs in the DefaultDomain, see Instrument the DefaultDomain for Standalone Applications.

Overview

Windows uses processes to manage security and performance isolation between running applications. Process isolation ensures one application's running code doesn't interfere with another application. However, for applications that share data, making calls between Windows processes can introduce complications and performance issues. AppDomains enable developers to create several applications that run inside a single process but maintain application isolation.

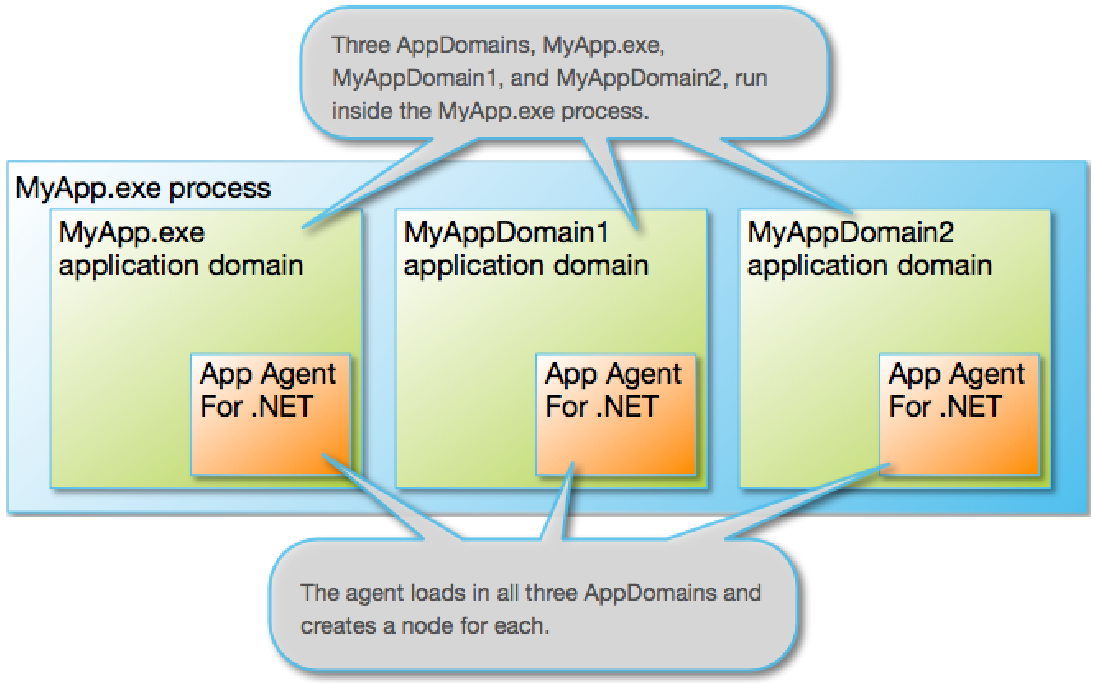
Single Application Domain

In the case of a single application running inside its own process, the runtime host typically manages the AppDomain. The application executable and the AppDomain have the same name. The App Agent for .NET (agent) installs itself inside the single AppDomain and creates a node for the application.


A single AppDomain MyApp.exe runs inside the MyApp.exe process.
MyApp.exe process
MyApp.exe application domain
App Agent For .NET
The agent loads in the MyApp.exe AppDomain and creates a node.

Multiple Application Domains

When developers include multiple AppDomains in an application, all the AppDomains run inside a single process. The application executable may have the same name as one AppDomain, but there are other, uniquely named AppDomains. By default, the agent installs itself inside all the AppDomains and creates nodes for them.


Three AppDomains, MyApp.exe, MyAppDomain1, and MyAppDomain2, run inside the MyApp.exe process.
MyApp.exe process
MyApp.exe application domain
MyAppDomain1 application domain
MyAppDomain2 application domain
App Agent For .NET
App Agent For .NET
App Agent For .NET
The agent loads in all three AppDomains and creates a node for each.

Configure Monitoring for Multiple Application Domains

If the application you monitor contains multiple AppDomains, the App Agent for .NET automatically instruments each AppDomain and creates a node. *New in 3.7.12*, you can configure the App Agent for .NET to instrument only the AppDomains you specify. This is useful to exclude AppDomains you don't want to monitor and to limit the number of nodes in a tier.

You can configure application domain monitoring for:

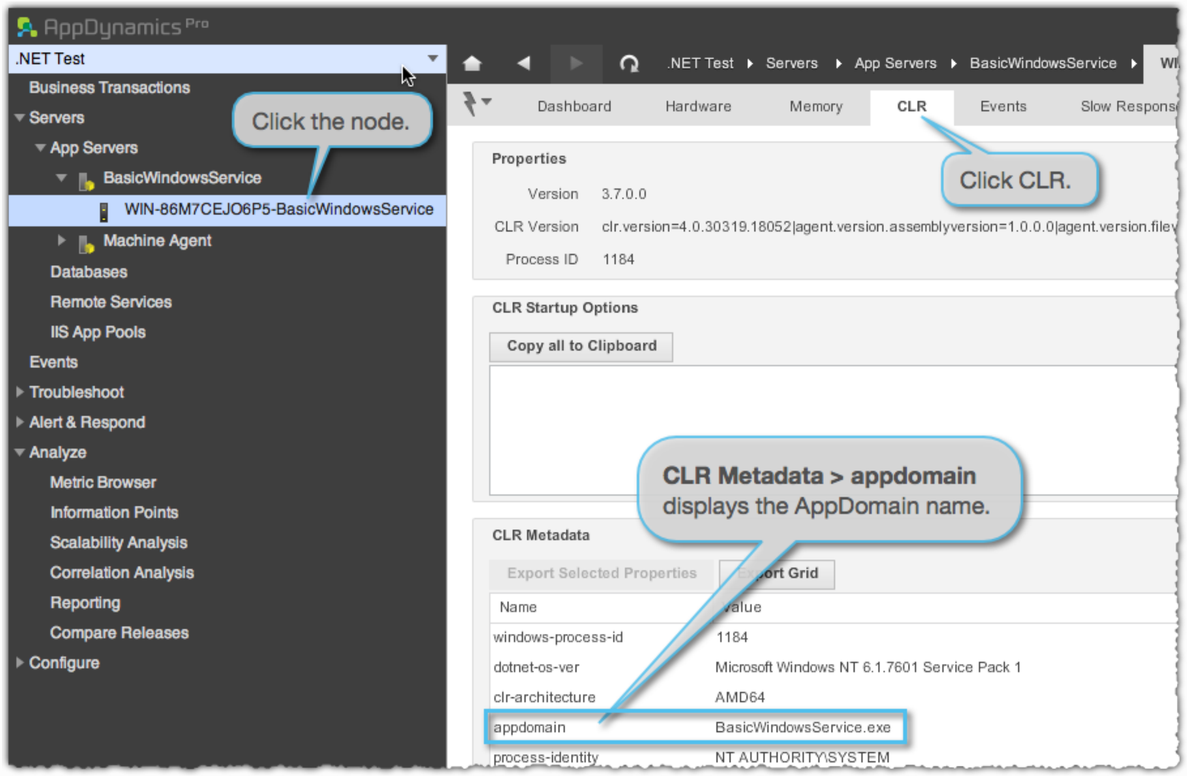
- Windows Services
- Standalone Applications

To configure Application Domain instrumentation.

Configure all instrumentation settings for the App Agent for .NET in the config.xml file. See Where to Configure App Agent Properties.

1. Identify the name of the AppDomains you want to instrument.

> If you have already instrumented your application, you can see the AppDomain names in the Node Dashboard.
>
> Click the node in the left navigation pane, then click **CLR**. See Node Dashboard.



Properties

Version	3.7.0.0
CLR Version	clr.version=4.0.30319.18052|agent.version.assemblyversion=1.0.0.0|agent.version.filev
Process ID	1184

CLR Metadata

Name	Value
windows-process-id	1184
dotnet-os-ver	Microsoft Windows NT 6.1.7601 Service Pack 1
clr-architecture	AMD64
appdomain	BasicWindowsService.exe
process-identity	NT AUTHORITY\SYSTEM

2. Launch a text editor as administrator.

3. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.

4. Find the element that corresponds to your application with multiple AppDomains:

   Windows Service element: `<windows-service name="MyWindowsService">`

Standalone Application element: `<standalone-application executable="MyWindowsApplication.exe">`

5. Add the **app-domain-name** attribute to the element.

For example, to instrument the MyApp.exe AppDomain for the MyApp.exe standalone application:

```
<standalone-application executable="MyApp.exe" app-domain-name="MyApp.exe">
    <tier name="StandaloneApplication Tier"/>
</standalone-application>
```

ⓘ As soon as you instrument one AppDomain in the config.xml, the agent instruments only the AppDomains you specify. Other AppDomains are not instrumented.

6. To instrument additional AppDomains, add an element for each AppDomain as if they were separate applications.

For example, to instrument MyAppDomain1 in MyApp.exe:

```
<standalone-application executable="MyApp.exe" app-domain-name="MyAppDomain1">
    <tier name="StandaloneApplication Tier"/>
  </standalone-application>
```

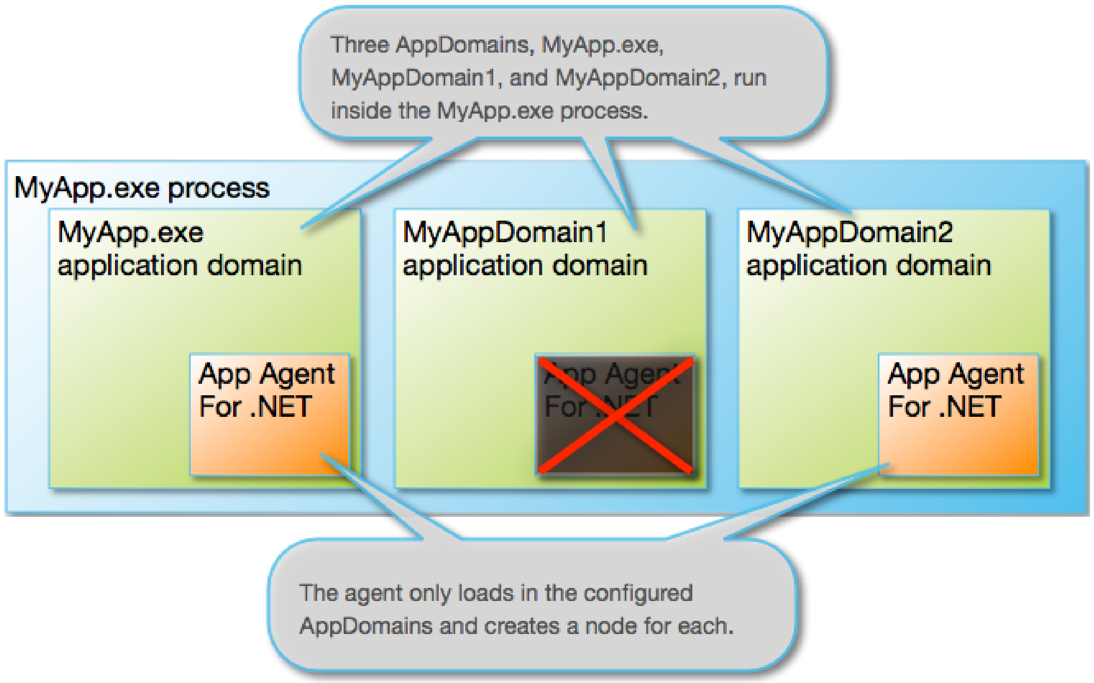
7. Save the config.xml file.

8. Restart the AppDynamics.Agent.Coordinator service.

9. Restart instrumented applications: Windows services or standalone applications.

Sample Standalone Application configuration with multiple AppDomains

This sample config.xml shows the configuration for the application MyApp.exe. Instrumentation only applies to the AppDomains specified in the config.xml: MyApp.exe and MyAppDomain2.


Three AppDomains, MyApp.exe, MyAppDomain1, and MyAppDomain2, run inside the MyApp.exe process.
MyApp.exe process
MyApp.exe application domain
App Agent For .NET
MyAppDomain1 application domain
App Agent For .NET
MyAppDomain2 application domain
App Agent For .NET
The agent only loads in the configured AppDomains and creates a node for each.

```
<?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="MyDotNetApplication" />
  </controller>
  <machine-agent />
  <app-agents>
    <standalone-applications>
      <standalone-application executable="MyApp.exe"
app-domain-name=""MyApp.exe">
        <tier name="StandaloneApplication Tier"/>
      </standalone-application>
      <standalone-application executable="MyApp.exe"
app-domain-name=""MyAppDomain2">
        <tier name="StandaloneApplication Tier"/>
      </standalone-application>
    </standalone-applications>
  </app-agents>
</appdynamics-agent>
```

Learn More

- Instrument DefaultDomain for Standalone Applications
- Application Domains
- App Agent for .NET Configuration Properties
- Enable the App Agent for .NET for Windows Services

- Enable the App Agent for .NET for Standalone Applications

Instrument the DefaultDomain for Standalone Applications

This topic provides instructions to identify if your application runs in the .NET DefaultDomain, and, if so, how to instrument it.

Instrument the DefaultDomain for Standalone Applications

By default, the App Agent for .NET does not instrument the **DefaultDomain** AppDomain. Before you instrument the DefaultDomain:

- Follow the instructions to instrument a standalone application.
- Create a POCO entry point for a class/method in the application.

If you complete those steps and still don't see business transactions in the Controller, check if your managed code runs in the DefaultDomain. If so, you must configure the agent to instrument the DefaultDomain.

To check if your application runs in the DefaultDomain

If you are unfamiliar with your application's managed code, you can use the agent logs to identify the AppDomain.

1. Open the agent log:
   Windows Server 2008 and later: `%ProgramData%\AppDynamics\DotNetAgent\Logs\AgentLog.txt`
   Windows Server 2003: `%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Logs\AgentLog.txt`

2. Search the agent log for "AppDomain".
Few log entries contain "AppDomain" when the agent starts up. Look for an entry by "dllhost" or your instrumented application similar to the following:

```
2013-12-16 08:23:02.3120 3068 MYPROGRAM 1 1 Info Configuration
appDomainName=DefaultDomain appDomainId=1 iis-app=null site=null port=null
appPoolId=
2013-12-16 08:23:02.6240 3192 dllhost 1 17 Info ConfigurationManager Not
instrumenting DefaultDomain for pid 3068
```

In this example MYPROGRAM is the name of the instrumented standalone application. You can see the name of the AppDomain in the log entry: `appDomainName=DefaultDomain`.

To instrument the DefaultDomain

Configure all instrumentation settings for the App Agent for .NET in the config.xml file. See Where to Configure App Agent Properties.

1. Launch a text editor as administrator.

2. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.

3. Copy the code block below to a child element of the **Standalone Application** element. See Standalone Application Element:

```
<profiler>
        <instrument-defaultdomain enabled="true"/>
    </profiler>
```

The **Profiler** element must follow the **Standalone Application Tier** element.

For 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.mycompany.com" port="8090" ssl="false">
    <application name="My Business Application" />
  </controller>
  <machine-agent />
  <app-agents>
    <IIS>
        <applications />
    </IIS>
    <standalone-applications>
      <standalone-application executable="MyStandaloneApp.exe">
        <tier name="Standalone Tier" />
        <profiler>
            <instrument-defaultdomain enabled="true"/>
        </profiler>
      </standalone-application>
    </standalone-applications>
  </app-agents>
</appdynamics-agent>
```

4. Save the config.xml file.

5. Restart the AppDynamics.Agent.Coordinator service.

6. Restart the standalone application for your changes to take effect.

Learn More

Enable the App Agent for .NET for Standalone Applications
POCO Entry Points
App Agent for .NET Configuration Properties

Getter Chains in .NET Configurations

This topic provides some guidance and examples of the correct syntax for using getter chains in AppDynamics configurations.

Using Getter Chains

You can use getter chains to:

- Configure method invocation data collectors. See Configure Data Collectors.
- Define a new business transaction custom match rule that uses a POCO object instance as the mechanism to name the transaction. See POCO Entry Points.

Note: If a getter chain calls on a method that does a lot of processing, such as making numerous SQL calls, it can degrade the performance of the application and the App Agent for .NET. Ideally, use getter chains only with simple gets.

An example of a simple getter would be a method to return a property from a class, such as GetName().

```
public class MyUser
{
private String Name {get; set;}
private String Url;
public GetUrl(){
 return this.Url;
 }
}
```

Separators in Getter Chains

The following special characters are used as separators:

- comma (,) for separating parameters
- forward slash (/) for separating a type declaration from a value in a parameter
- dot (.) for separating the methods and properties in the getter chain

If a slash or a comma character is used in a string parameter, use the backslash (\) escape character. Except in the case of type / value.

If a literal dot is used in a string parameter, use the backslash escape character before the dot.

Getter Chain Examples

> These examples are intended to get you started with the getter chain syntax for AppDynamics.

- Getter chain with integer parameters in the substring method using the forward slash as the type separator:

  ```
  GetAddress(appdynamics, sf).Substring(int/0, int/10)
  ```

  This example returns the first 10 characters of a string. The int parameter types resolve which overloaded method to call.

- Getter chain with various non-string parameter types:

  ```
  GetAddress(appdynamics, sf).MyMethod(float/0.2, boolean/true,
  boolean/false, int/5)
  ```

  For cases when you specify `type / char`, you don't need to escape anything after the type.

- Getter chain with forward slash escaped; escape character needed here for the string parameter:

  ```
  GetUrl().Split(char[]//)
  ```

  This example splits a URL on the forward slash character and returns a string array. The following examples are more likely.

- Getter chain with an array element:

  ```
  GetUrl().Split(char[]//).[4]
  ```

- Getter chain with multiple array elements separated by commas:

  ```
  GetUrl().Split(char[]//).[1,3]
  ```

- Getter chain retrieves property values, such as the length of an array:

  ```
  GetUrl().Split(char[]//).Length
  ```

- Getter chain using backslash to escape the dot in the string parameter;
  the call is getParam (a.b.c).

```
GetAddress.GetParam(a\.b\.c\.)
```

- In the following getter chain, the first dot requires an escape character because it is in a string method parameter (inside the parentheses). The second dot does not require an escape character because it is not in a method parameter (it is outside the parentheses).

```
GetUser(suze\.smith).GetGroup().GetId()
```

- The following getter chain is from a transaction splitting rule on URIs that use a semicolon as a delimiter; for example:

  /my-webapp/xyz;sessionid=BE7F31CC0235C796BF8C6DF3766A1D00?act=Add&uid=c42ab7ad-48a7-4353-bb11-0dfeabb798b5

  The getter chain splits on the API name, so the resulting split transactions are "API.abc", API."xyz" and so on.

  The call gets the URI using GetUri() and then splits it using the escaped forward slash. From the resulting array it takes the third entry (as the split treats the first slash as a separator) and inserts what before the slash (in this case, nothing) into the first entry. Then it splits this result using the semicolon, getting the first entry of the resulting array, which in this case contains the API name.

```
GetUri().Split(char[]//).[2].Split(char[];).[0]
```

Learn More

- Configure Business Transaction Detection
- Configure Data Collectors

Enable Monitoring for Windows Performance Counters

- Performance Counters and the .NET Machine Agent
  - To configure additional performance counters for .Net
  - Sample .NET Machine Agent configuration with additional performance counters

Performance Counters and the .NET Machine Agent

By default, the .NET Machine Agent uses Microsoft Performance Counters to gather and report .NET metrics. For details on the preconfigured .NET metrics see Monitor CLRs and Monitor IIS.

You can specify additional performance counters to be reported by the .NET Machine Agent.

To configure additional performance counters for .Net

1. Shut down the AppDynamics.Agent.Coordinator service.
2. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.
3. Add the Performance Counters block as a child of the Machine Agent element.

```
<perf-counters>
    <perf-counter cat="" name="" instance=""/>
  </perf-counters>
```

4. Create a Performance Counter element for each performance counter you want to add. Use any of the performance counters as specified in Performance Counters in .NET Framework.
    - Set the cat attribute to the category of the performance counter.
    - Set the name attribute to the performance counter name.
    - Set the instance attribute to the of the performance counter.

    If a particular performance counter has many instances you can specify the following options:
    - instance ="*" OR
    - instance ="all" (This will report the sum of all instances)

    For example, to add the performance counter for measuring CPU Idle time(%), add the following element in the <perf-counters> block:

    ```
    <perf-counter cat="Processor" name="% Idle Time"
    instance="_Total"/>
    ```

5. Save the config.xml.
6. Start the AppDynamics.Agent.Coordinator service.

Sample .NET Machine Agent configuration with additional performance counters

```
<machine-agent>
  <!-- Additional machine level Performance Counters -->
  <perf-counters>
    <perf-counter cat="Processor" name="% Idle Time" instance="_Total"/>
  </perf-counters>
</machine-agent>
```

Configure the .NET Machine Agent

    - Machine Agent Tier
- Learn More

This topic provides instructions to configure the .NET Machine Agent that installs automatically with the App Agent for .NET.

The .NET Machine Agent

The App Agent for .NET includes an embedded .NET Machine Agent that runs as part of the AppDynamics.Agent.Coordinator service. Among other things, the Machine Agent regularly gathers system performance data and reports it back to the Controller as metrics.

Do not confuse the .NET Machine Agent with the Standalone Machine Agent, a Java application. The Standalone Machine Agent provides the capability to use extensions (plugins, metric listener, orchestration). See Standalone Machine Agent.

Machine Agent Configuration

The app agent MSI Installer package and App Agent for .NET Configuration Utility automatically install and configure the .NET Machine Agent to connect to the Controller. The connection information is the same for the app agent and the .NET Machine agent. See Configure the App Agent for .NET.

You can customize the .NET Machine Agent to enable additional functionality.

To configure the .NET Machine Agent

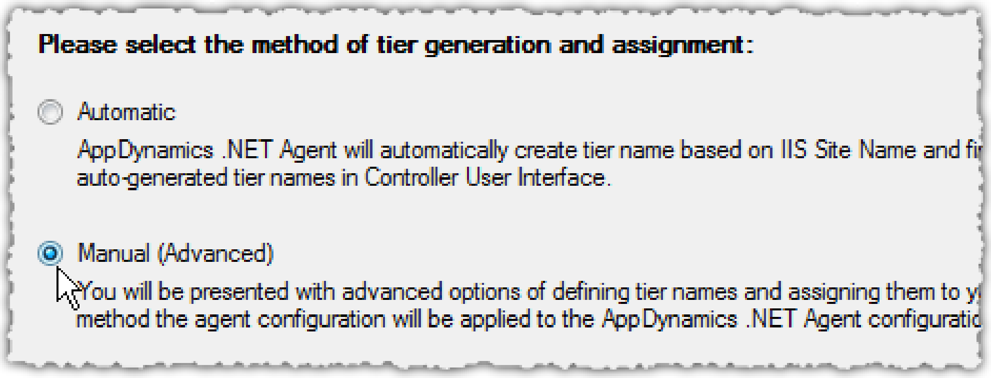
1. If you haven't already, Install the App Agent for .NET.
2. Customize instrumentation settings for the Machine Agent element in the config.xml file. See Where to Configure App Agent Properties.

   See the following topics for .NET Machine Agent configuration options:

   - Enable Monitoring for Windows Performance Counters
   - Enable Thread Correlation for .NET
   - Enable Correlation for .NET Remoting
   - Enable Instrumentation for WCF Data Services

To Configure the .NET Machine Agent without app agents

1. If you haven't already, Install the App Agent for .NET.
2. Launch the AppDynamics Agent Configuration utility and follow the steps until you reach the Assign IIS applications to tiers window.
3. Click **Manual** for the method of tier generation and assignment and click **Next**.

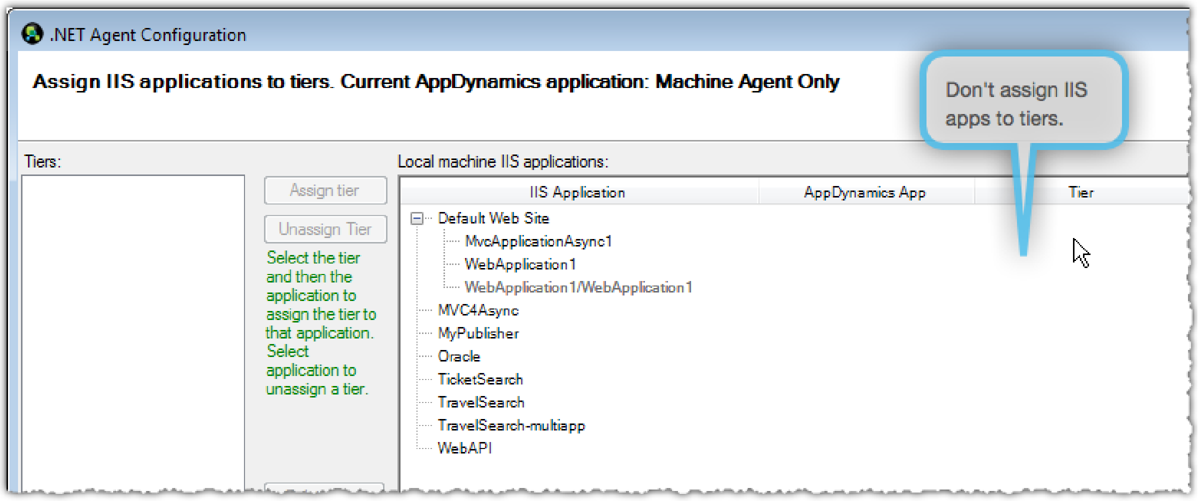


Please select the method of tier generation and assignment:

Automatic
AppDynamics .NET Agent will automatically create tier name based on IIS Site Name and fi
auto-generated tier names in Controller User Interface.

[✓ Manual (Advanced)]
You will be presented with advanced options of defining tier names and assigning them to y
method the agent configuration will be applied to the AppDynamics .NET Agent configuratic

4. Don't assign any IIS applications to tiers, click **Next**.



.NET Agent Configuration

Assign IIS applications to tiers. Current AppDynamics application: Machine Agent Only

Don't assign IIS apps to tiers.

Tiers:

Local machine IIS applications:

Select the tier and then the application to assign the tier to that application. Select application to unassign a tier.

IIS Application	AppDynamics App	Tier
Default Web Site		
MvcApplicationAsync1		
WebApplication1		
WebApplication1/WebApplication1		
MVC4Async		
MyPublisher		
Oracle		
TicketSearch		
TravelSearch		
TravelSearch-multiapp		
WebAPI		

If you configured any Windows services or standalone applications, manually disable those agents in the config.xml.

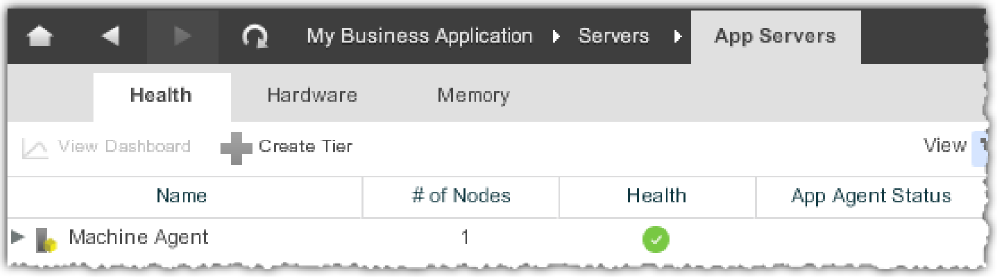
5. Continue with the remaining steps and click **Done**.

   Monitoring resumes for the .NET Machine Agent only.

Machine Agent Tier

Immediately after you install and configure the App Agent for .NET, the .NET Machine Agent registers with the Controller and starts reporting performance data.

Frequently the machine agent metrics reach the controller before the app agent has had time to instrument and register IIS applications, Windows services, or standalone applications. If there are no application tiers, the machine agent registers as the Machine Agent tier.



Name	# of Nodes	Health	App Agent Status
Machine Agent	1	✓	

Once the app agent begins reporting metrics for configured application tiers, the .NET Machine Agent reports to the application tiers and stops sending data to the Machine Agent tier.

Learn More

- Install the App Agent for .NET
- Monitor CLRs

Enable Instrumentation for WCF Data Services

- To enable instrumentation for WCF Data Services

Learn More

AppDynamics supports instrumentation for WCF Data Services including the WCF RIA Services for Microsoft LightSwitch.

To enable instrumentation for WCF Data Services

Configure all instrumentation settings for the App Agent for .NET in the config.xml file. See Where to Configure App Agent Properties.

1. Launch a text editor as administrator.

2. Edit the config.xml file as an administrator. See Where to Configure App Agent Properties.

3. Copy the code block below to a child element of the Machine Agent element. (See Machine Agent Element):

```
<instrumentation>
        <instrumentor name="WCFDSEntryInstrumentor" enabled="true" />
    </instrumentation>
```

For 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">
...
 <machine-agent>
    <!--Enable thread correlation-->
    <instrumentation>
        <instrumentor name="WCFDSEntryInstrumentor" enabled="true" />
    </instrumentation>
  </machine-agent>
...
</appdynamics-agent>
```

The configuration syntax is **enabled="true"**.

4. Save the config.xml file.

5. Restart the AppDyanmics.Agent.Coordinator Service.

6. Restart instrumented applications for your changes to take effect.

Learn More

Configure Business Transaction Detection for .NET

App Agent for .NET Configuration Properties

Monitor .NET Applications

- Monitor CLRs
- Monitor IIS
- Monitor Async Transactions for .NET
- Monitor Oracle Backends for .NET with AppDynamics for Databases

The .NET embedded machine agent reports hardware metrics such as:

- CPU activity
- Memory usage
- Disk reads and writes
- Network traffic

When IIS is installed on the machine, the .NET machine agent also reports IIS, ASP.NET and ASP.NET Application metrics. See Monitor IIS.

The App Agent for .NET reports CLR metrics, which are also based on MS Performance Counters. See Monitor CLRs.

If you install the Standalone Machine Agent, which is a Java application, in a .NET environment, then you can add custom monitors and extensions such as the HTTP listener. See Standalone Machine Agent for details.

Learn More

- Monitor Databases
- Monitor Remote Services

Monitor CLRs

AppDynamics uses Microsoft Windows Performance Counters to gather .NET metrics. These preconfigured CLR metrics can be viewed from the Node Dashboard and in the Metric Browser.

Refer to the Microsoft documentation for more information:

- General overview: Performance Counters
- ASP.NET Counters: Performance Counters for ASP.NET

CLR Metrics

CLR metrics give insight into how the .NET runtime is performing. The AppDynamics preconfigured CLR metrics include:

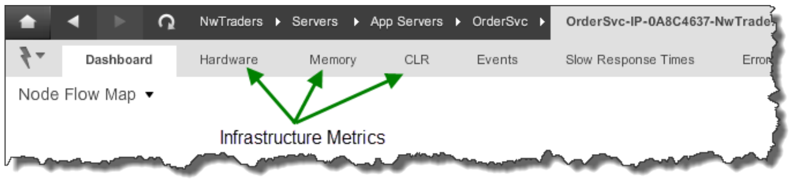
- .NET CLR memory usage
- Total classes loaded and how many are currently loaded
- Garbage collection time spent, and detailed metrics about GC memory pools and caching
- Locks and thread usage
- Memory heap and non-heap usage, including the large object heap
- Percent CPU process usage

To access the Node Dashboard

1. Select the business application.

2. In the left navigation pane, click **Servers -> App Servers -> <Tier> -> <Node>**. AppDynamics displays the Node Dashboard for the selected node.

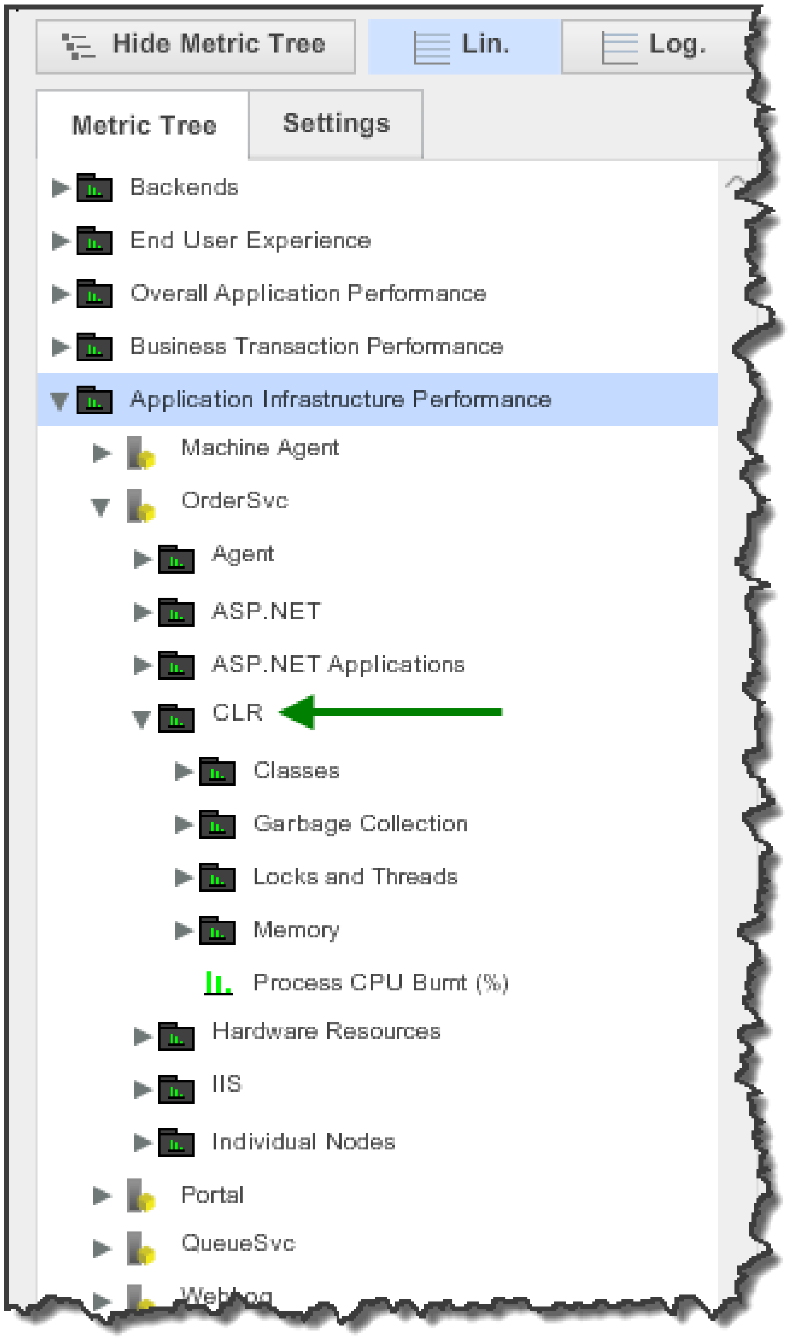
3. Click the tab for the metrics you want to view.



See Node Dashboard for more details.

To access the Metric Browser

1. In the left navigation pane of the Controller, click **Analyze -> Metric Browser**.
2. Expand the **Application Infrastructure Performance** branch of the **Metric Tree**.
3. Expand the tier that you want to view.
4. Expand the tree for each category of metrics.
5. To view the CLR metrics in the Metric Browser, expand **Application Infrastructure Performance -> <Node> -> CLR**.



See Metric Browser for more details.

Alerting for CLR Health

You can set up health rules based on the infrastructure metrics. Once you have a health rule, you can create specific policies based on health rule violations. One type of response to a health rule violation is an alert.

In addition to the default metrics, you may be interested in additional metrics. You can specify additional performance counters to be reported by the .NET Machine Agent. See Enable Monitoring for Windows Performance Counters for details.

Once you add a custom metric you can create a health rule for it and receive alerts when conditions indicate problems.

See Alert and Respond for details on health rules and policies.

Learn More

- Infrastructure Monitoring
- Performance Counters for ASP.NET (external website)
- Install the App Agent for .NET
- Enable Monitoring for Windows Performance Counters

Monitor IIS

AppDynamics uses Microsoft Windows Performance Counters to gather .NET metrics. These preconfigured IIS related metrics can be viewed in the Metric Browser.

For more information on Windows Performance Counters, refer to the Microsoft documentation Pe rformance Counters for ASP.NET.

Default Metrics Available in the .NET Environment

Internet Information Services (IIS) must be installed on the machine for you to view the metrics for the following:

- IIS
- ASP.NET
- ASP.NET Application

ASP.NET Metrics

To view the ASP.NET metrics in the Metric Browser, expand **Application Infrastructure Performance -> <Node> -> ASP.NET**.

AppDynamics reports the following ASP.NET metrics:

- Application Restarts
- Applications Running
- Requests Disconnected
- Requests Queued
- Requests Rejected
- Request Wait Time
- Worker Process Restarts

ASP.NET Application Metrics

To view the ASP.NET Application metrics in the Metric Browser, expand **Application Infrastructure Performance -> <Node> -> ASP.NET Applications**.

AppDynamics reports the following ASP.NET Application metrics:

- Anonymous Requests
- Anonymous Requests/sec
- Cache Total Entries
- Cache Total Hit Ratio
- Cache Total Turnover Rate
- Cache API Entries
- Cache API Hit Ratio
- Cache API Turnover Rate
- Errors Unhandled During Execution/sec
- Errors Total/sec
- Errors During Preprocessing
- Errors During Compilation
- Errors During Execution
- Errors Unhandled During Execution
- Errors Unhandled During Execution/sec
- Errors Total
- Errors Total/sec
- Output Cache Entries
- Output Cache Hit Ratio
- Output Cache Turnover Rate
- Pipeline Instance Count
- Requests Executing
- Requests Failed
- Requests In Application Queue
- Requests Not Found
- Requests Not Authorized
- Requests Succeeded
- Requests Timed Out
- Requests Total
- Requests/sec
- Session State Server Connections Total
- Session SQL Server Connections Total
- Sessions Active

- Sessions Abandoned
- Sessions Timed Out
- Sessions Total
- Transactions Aborted
- Transactions Committed
- Transactions Pending
- Transactions Total
- Transactions/sec

IIS Metrics

From the Metric Browser:

- To view the IIS metrics for a tier, expand **Application Infrastructure Performance -> <Tier> -> IIS**.
- To View the IIS metrics for a node, expand **Application Infrastructure Performance -> <Tier> -> Individual Nodes -> <Node> -> IIS**.

Each metric is reported for the entire tier, each individual application pool, and each individual node as follows:

- **Application Infrastructure Performance -> <Tier> -> IIS** = combined for all IIS processes in all Application Pools for this tier
- **Application Infrastructure Performance -> <Tier> -> Application Pools -> <application pool name>** = combined for all processes in this specific Application Pool
- **Application Infrastructure Performance -> <Tier> -> Individual Nodes -> <Node>** = metrics for the specific node.

Monitoring IIS Application Pools

You can monitor the health of IIS application pools for the instrumented .NET nodes in a tier. You can view the information by application pool, machine, and process IDs in varying hierarchies.

These groupings enable you to visualize key performance indicators for your infrastructure:

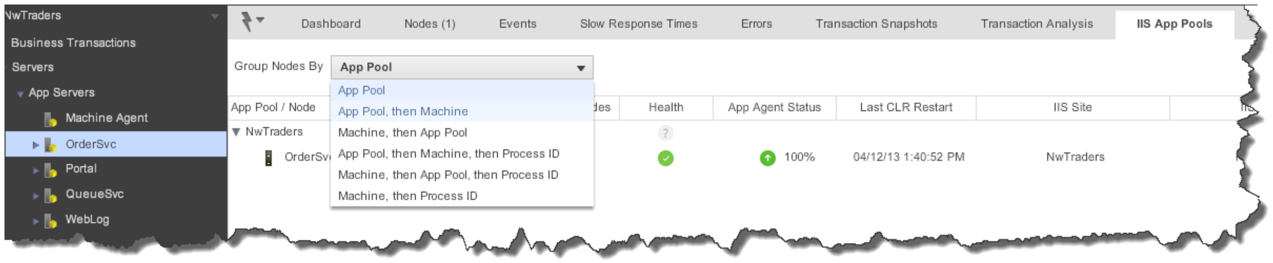
- Health of the node in the specified time-range for a particular group.
- App Server Agent's status in the specified time-range.
- Last CLR restart.
- A link to the parent tier.

To view the IIS application pools:

1. In the left navigation pane, select the tier that you want to monitor: **Servers -> AppServers -> <Tier>**.

2. Click the **IIS App Pools** tab.
The IIS App Pools list displays.

3. (Optional) From the **Group Nodes By** dropdown menu, select how you want to view the application pools and machines for this tier.



Group Nodes By: App Pool

App Pool / Node	Health	App Agent Status	Last CLR Restart	IIS Site
NwTraders	?			
OrderSv	✓	100%	04/12/13 1:40:52 PM	NwTraders

4. To view a node's dashboard, double-click the node in the list. From there you can select the various tabs for details about performance of the node. See Node Dashboard.

If a machine or application pool name is not available for a .NET node, an "**Unknown App Pool**" / "**Unknown Machine**" grouping is created.

Monitor Async Transactions for .NET

This topic describes how to enable asynchronous exit point detection in the App Agent for .NET. It covers the .NET asynchronous programming patterns the agent detects and provides instruction on how to use AppDynamics Controller features that represent asynchronous transactions.

For more information about remote services and exit points, see Monitor Remote Services and Co nfigure Backend Detection for .NET.

Asynchronous Transactions in .NET

Developers use asynchronous programming patterns to create scalable, more performant applications. Microsoft .NET lets you designate methods as asynchronous tasks. The .NET runtime releases resources for asynchronous methods while tasks complete. When task processing finishes, the runtime calls back to the originating asynchronous method so the method may continue processing.

Because tasks may execute in parallel, AppDynamics sometimes represents asynchronous activity differently from synchronous activity in the Controller.

Supported asynchronous programming patterns

The agent discovers the following asynchronous programming patterns for HTTP, Web Service, and WCF exit points:

Microsoft .NET 4.5 **async** and **await** keywords. See Asynchronous Programming with Async and Await.

Not limited to exit point discovery:

Microsoft .NET 4.5 **TaskFactory.FromAsync** wrapper. See TPL and Traditional .NET Framework Asynchronous Programming.

If the entry point on a downstream WCF tier is asynchronous, the agent doesn't detect the transaction on the downstream tier.

To enable asynchronous exit point detection

To enable asynchronous exit point detection, register the **async-tracking** node property.

For instructions on how to set a node property, see App Agent Node Properties.

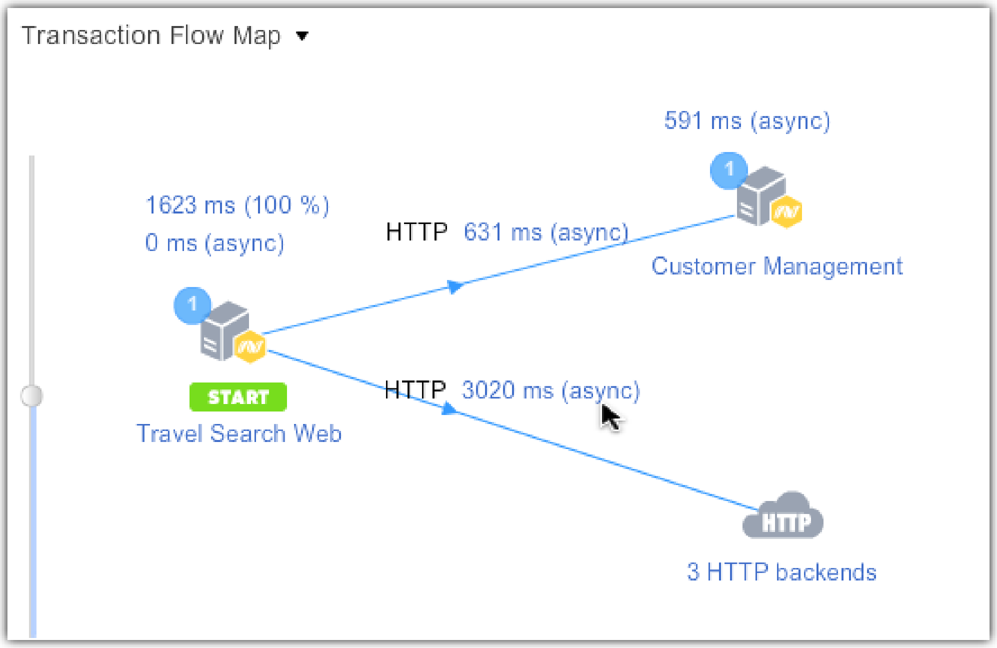
Name: async-tracking
Description: Enable detection of asynchronous exit points. *New in 3.8.5*, enable thread correlation for ThreadPoolQueueUserWorkItem.
Type: Boolean
Value: True

The default value is **False**.

Identify Asynchronous Transactions in Dashboards

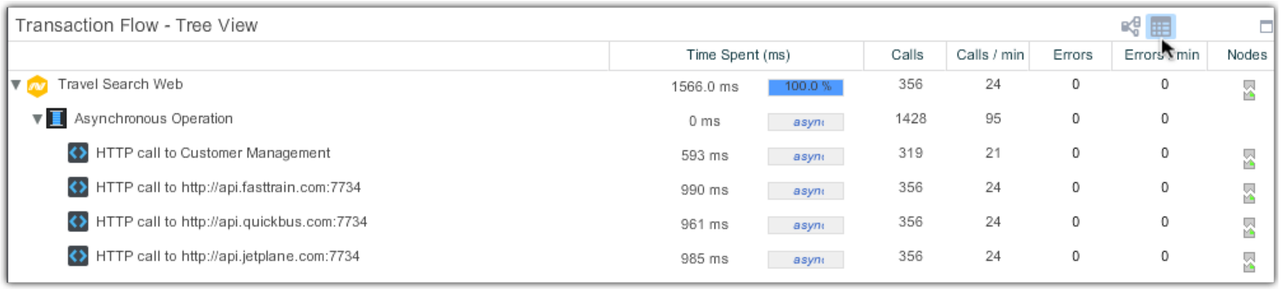
When AppDynamics detects asynchronous exit points in an application it labels the calls as **async** in the dashboards. Because they may execute simultaneously the Controller doesn't display percentage value of the end-to-end transaction time for asynchronous calls.

For example, consider a travel site application that asynchronously searches fares on multiple provider sites and displays them back to the customer. In this case, the agent discovers the asynchronous HTTP calls and displays them on the transaction flow map.



For further visibility of asynchronous calls, edit the current flow map and click **Use dotted line** under **Asynchronous Activity**.

Use the transaction flow tree view of the asynchronous transaction to display errors and time spent in asynchronous tasks.


Transaction Flow - Tree View

	Time Spent (ms)		Calls	Calls / min	Errors	Errors / min	Nodes
Travel Search Web	1566.0 ms	100.0 %	356	24	0	0	
Asynchronous Operation	0 ms	async	1428	95	0	0	
HTTP call to Customer Management	593 ms	async	319	21	0	0	
HTTP call to http://api.fasttrain.com:7734	990 ms	async	356	24	0	0	
HTTP call to http://api.quickbus.com:7734	961 ms	async	356	24	0	0	
HTTP call to http://api.jetplane.com:7734	985 ms	async	356	24	0	0	

Troubleshoot Asynchronous Calls in Transaction Snapshots

Transaction snapshots include several features to help you discover problem areas in business transactions that use asynchronous methods. For an overview of transaction snapshots, see Transaction Snapshots.

Transaction Snapshot Flow Map

The Transaction Snapshot Flow Map graphically represents the business transaction. It displays the user experience, execution time, and timestamp of the transaction. The flow map also provides details of the overall time that is spent in a particular tier and in database and remote service calls. The **async** label indicates asynchronous calls.

Snapshot Execution Waterfall View

The transaction Snapshot Execution Waterfall View shows a timeline representation of the end-to-end execution of the business transaction. Synchronous and asynchronous processes appear on a bar diagram illustrating their relative execution time arranged in chronological order.

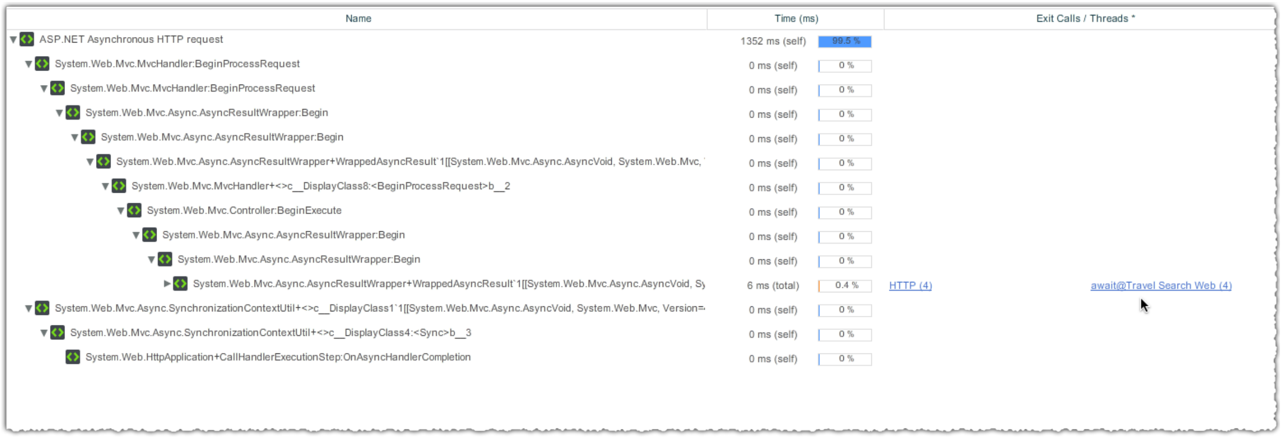
The waterfall view enables you to visually identify which processes are running the longest. Double-click a bar to drill down to a call graph and investigate problematic code.

Call Graph

When the agent detects asynchronous exit points, it displays an **await** link in the **Exit Calls/Threads** column of the call graph in the Call Drill Down. Click the link to display a list of the asynchronous calls. The format of the await link is as follows:

```
await@<tier name>
```

For example, the Travel Search Web tier makes asynchronous calls to the internal customer management system and to provider backends. The link shows as "await@Travel Search Web".

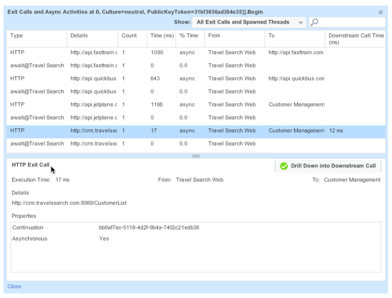


Name	Time (ms)		Exit Calls / Threads *	
ASP.NET Asynchronous HTTP request	1352 ms (self)	99.5 %		
System.Web.Mvc.MvcHandler:BeginProcessRequest	0 ms (self)	0 %		
System.Web.Mvc.MvcHandler:BeginProcessRequest	0 ms (self)	0 %		
System.Web.Mvc.Async.AsyncResultWrapper:Begin	0 ms (self)	0 %		
System.Web.Mvc.Async.AsyncResultWrapper:Begin	0 ms (self)	0 %		
System.Web.Mvc.Async.AsyncResultWrapper+WrappedAsyncResult`1[[System.Web.Mvc.Async.AsyncVoid, System.Web.Mvc,	0 ms (self)	0 %		
System.Web.Mvc.MvcHandler+<>c__DisplayClass8:<BeginProcessRequest>b__2	0 ms (self)	0 %		
System.Web.Mvc.Controller:BeginExecute	0 ms (self)	0 %		
System.Web.Mvc.Async.AsyncResultWrapper:Begin	0 ms (self)	0 %		
System.Web.Mvc.Async.AsyncResultWrapper:Begin	0 ms (self)	0 %		
System.Web.Mvc.Async.AsyncResultWrapper+WrappedAsyncResult`1[[System.Web.Mvc.Async.AsyncVoid, Sy	6 ms (total)	0.4 %	HTTP (4)	await@Travel Search Web (4)
System.Web.Mvc.Async.SynchronizationContextUtil+<>c__DisplayClass1`1[[System.Web.Mvc.Async.AsyncVoid, System.Web.Mvc, Version=	0 ms (self)	0 %		
System.Web.Mvc.Async.SynchronizationContextUtil+<>c__DisplayClass4:<Sync>b__3	0 ms (self)	0 %		
System.Web.HttpApplication+CallHandlerExecutionStep:OnAsyncHandlerCompletion	0 ms (self)	0 %		

The Exit Calls and Async Activities window shows a list of the exit calls and the corresponding **aw ait** continuation calls.

- Exit calls display by type: HTTP, web service, or WCF.
- Continuations/call backs display as `await@tier` name.

For example:
HTTP call to the Customer Management tier



Exit Calls and Async Activities at 0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]].Begin

Show: All Exit Calls and Spawned Threads

Type	Details	Count	Time (ms)	% Time	From	To	Downstream Call Time (ms)
HTTP	http://api.fasttrain.c	1	1090	async	Travel Search Web	http://api.fasttrain.com:	
await@Travel Search	http://api.fasttrain.c	1	0	0.0	Travel Search Web		
HTTP	http://api.quickbus.	1	643	async	Travel Search Web	http://api.quickbus.com	
await@Travel Search	http://api.quickbus.	1	0	0.0	Travel Search Web		
HTTP	http://api.jetplane.c	1	1198	async	Travel Search Web	Customer Management	
await@Travel Search	http://api.jetplane.c	1	0	0.0	Travel Search Web		
HTTP	http://crm.travelsea	1	17	async	Travel Search Web	Customer Management	12 ms
await@Travel Search	http://crm.travelsea	1	0	0.0	Travel Search Web		

HTTP Exit Call

Drill Down into Downstream Call

Execution Time: 17 ms From: Travel Search Web To: Customer Management

Details

http://crm.travelsearch.com:8989/CustomerList

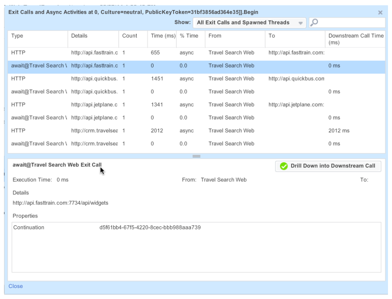
Properties

Continuation	bb9af7ec-5118-4d2f-9b4a-7402c21edb36
Asynchronous	Yes

Close

Continuation call back to the

Travel Search tier



Exit Calls and Async Activities at 0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]].Begin

Show: All Exit Calls and Spawned Threads

Type	Details	Count	Time (ms)	% Time	From	To	Downstream Call Time (ms)
HTTP	http://api.fasttrain.c	1	655	async	Travel Search Web	http://api.fasttrain.com:	
await@Travel Search \	http://api.fasttrain.c	1	0	0.0	Travel Search Web		0 ms
HTTP	http://api.quickbus.	1	1451	async	Travel Search Web	http://api.quickbus.con	
await@Travel Search \	http://api.quickbus.	1	0	0.0	Travel Search Web		0 ms
HTTP	http://api.jetplane.c	1	1341	async	Travel Search Web	http://api.jetplane.com:	
await@Travel Search \	http://api.jetplane.c	1	0	0.0	Travel Search Web		0 ms
HTTP	http://crm.travelsea	1	2012	async	Travel Search Web		2012 ms
await@Travel Search \	http://crm.travelsea	1	0	0.0	Travel Search Web		0 ms

await@Travel Search Web Exit Call

Execution Time: 0 ms From: Travel Search Web To:

Details

http://api.fasttrain.com:7734/api/widgets

Properties

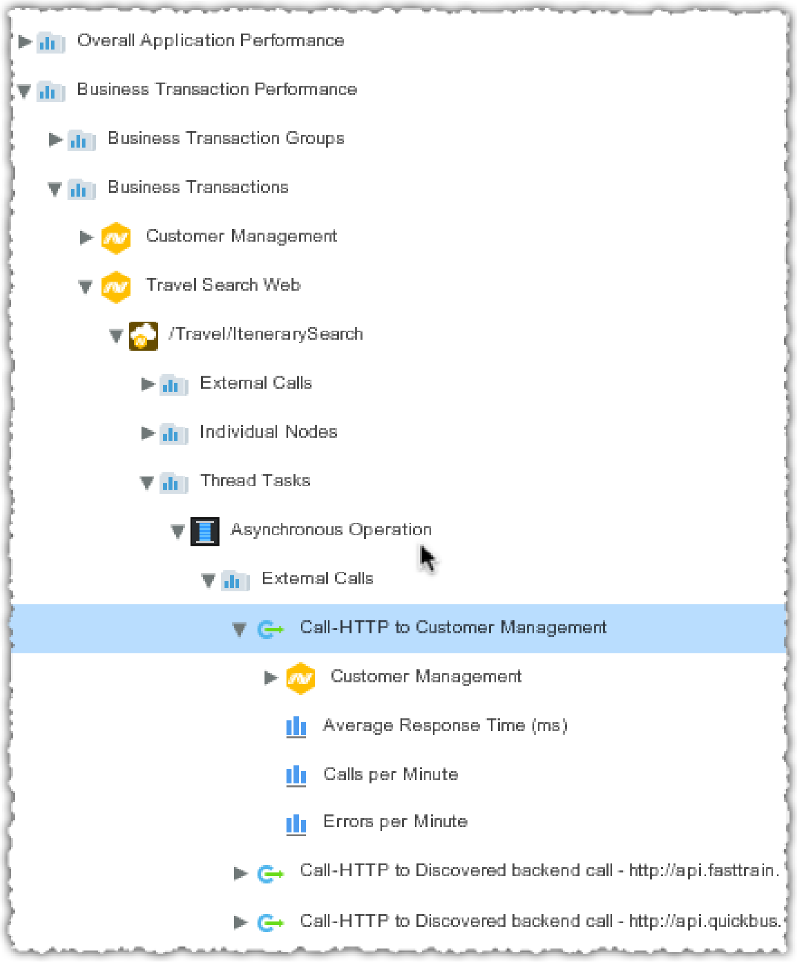
Continuation	d5f61bb4-67f5-4220-8cec-bbb988aaa739

Analyze Asynchronous Activity in the Metric Browser

The Metric Browser displays asynchronous activity in the following places:

- **Business Transaction Performance -> Business Transactions -> tier > business transaction -> Thread Tasks -> Asynchronous Operation -> External Calls**
- **Overall Application Performance -> tier -> Thread Tasks -> Asynchronous Operation -> External Calls**
- **Overall Application Performance -> tier -> Individual Nodes -> node name -> Thread Tasks -> Asynchronous Operation -> External Calls**

For example:



- ▶ Overall Application Performance
- ▼ Business Transaction Performance
  - ▶ Business Transaction Groups
  - ▼ Business Transactions
    - ▶ Customer Management
    - ▼ Travel Search Web
      - ▼ /Travel/IteneratySearch
        - ▶ External Calls
        - ▶ Individual Nodes
        - ▼ Thread Tasks
          - ▼ Asynchronous Operation
            - ▼ External Calls
              - ▼ Call-HTTP to Customer Management
                - ▶ Customer Management
                - Average Response Time (ms)
                - Calls per Minute
                - Errors per Minute
              - ▶ Call-HTTP to Discovered backend call - http://api.fasttrain.
              - ▶ Call-HTTP to Discovered backend call - http://api.quickbus.

For more information on how to use the Metric Browser, see Metric Browser.

Learn More

Using Asynchronous Methods in ASP.NET 4.5

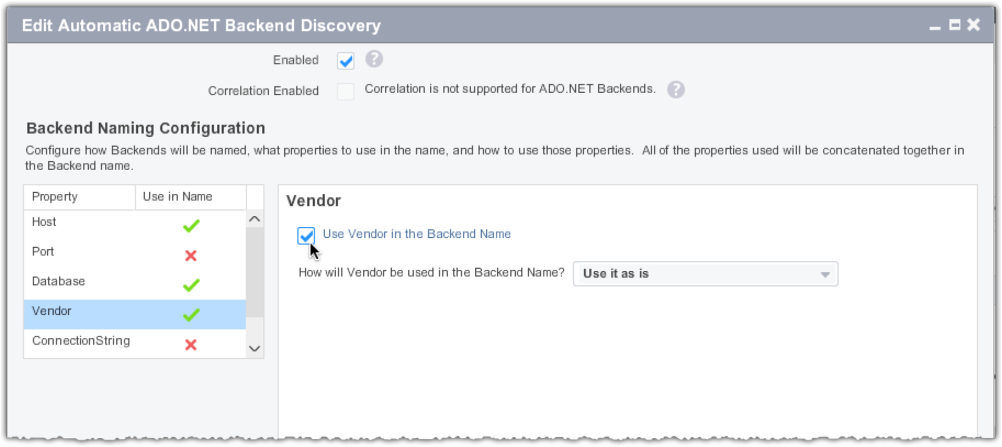
Monitor Oracle Backends for .NET with AppDynamics for Databases

Oracle ODP.NET database backends integrate with AppDynamics for Databases. This topic covers how to configure integration and an introduction to the enhanced features.

Configure Prerequisites

Before you can take advantage of the integration for Oracle backends with AppDynamics for Databases, you must perform the following setup:

1. Install AppDynamics for Databases and add a collector for your Oracle database. See Install AppDynamics for Databases and Add an AppDynamics for Databases Collector.
2. Enable integration with AppDynamics for Databases in the Controller. See Integrate with AppDynamics for Databases.
3. Log off from the Controller, then log on again.
4. Enable the Vendor property for ADO.NET backend naming. For more information see Changing the Default ADO.NET Automatic Discovery and Naming.


Edit Automatic ADO.NET Backend Discovery

Enabled ✓

Correlation Enabled — Correlation is not supported for ADO.NET Backends.

Backend Naming Configuration

Configure how Backends will be named, what properties to use in the name, and how to use those properties. All of the properties used will be concatenated together in the Backend name.

Property	Use in Name
Host	✓
Port	✗
Database	✓
Vendor	✓
ConnectionString	✗

Vendor

✓ Use Vendor in the Backend Name

How will Vendor be used in the Backend Name? Use it as is

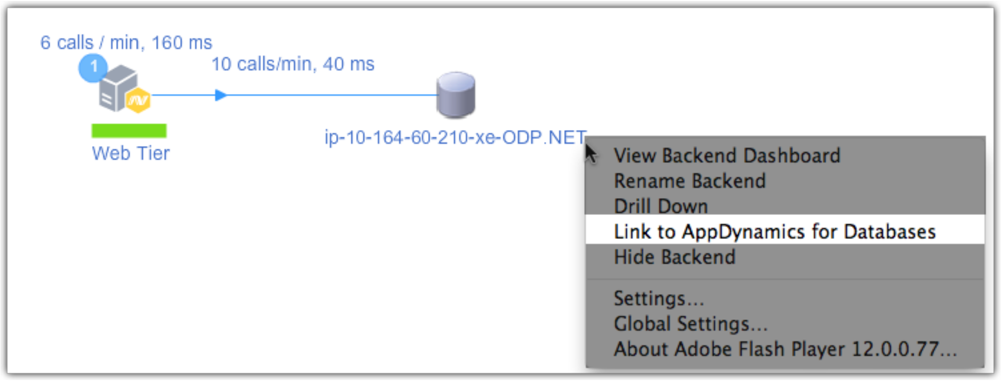
AppDynamics for Databases requires the Vendor property to identify the Oracle database.

Monitor Oracle Database Backends

After you configure integration, AppDynamics enables links from the Controller to AppDynamics for Databases.

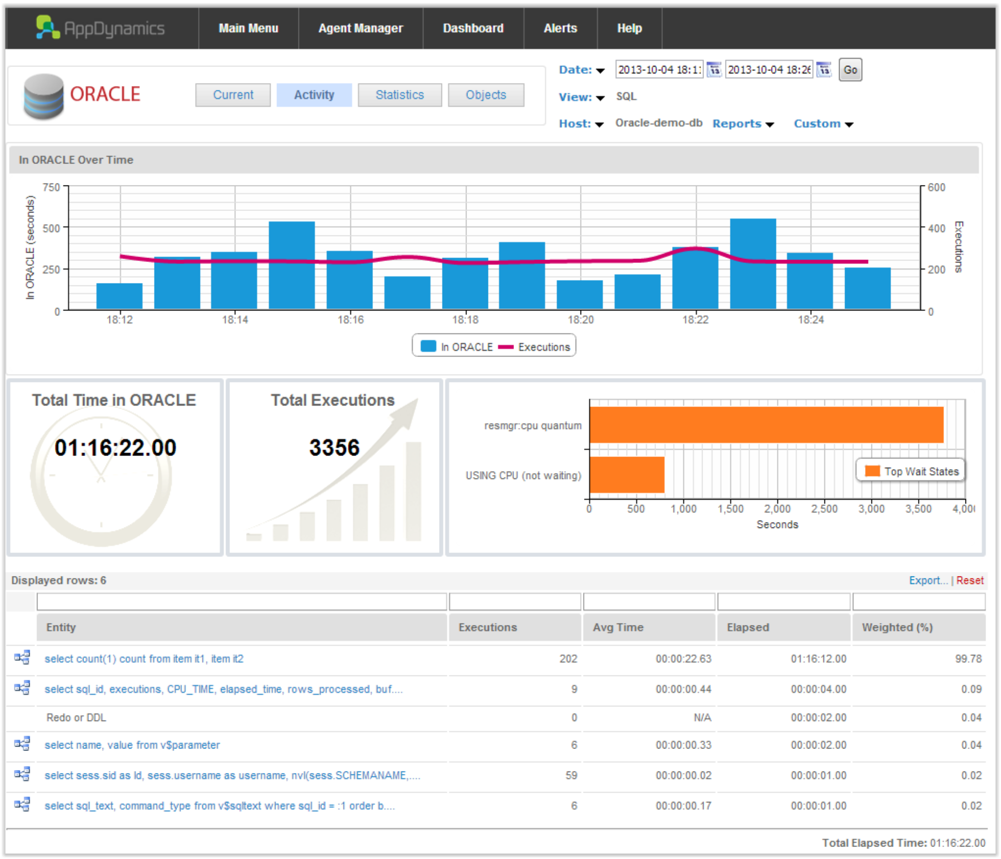
To monitor Oracle databases from the Controller

1. Right-click the Oracle database and click **Link to AppDynamics for Databases**.



In addition to the flow map, you can right-click the Oracle database in **Servers > Databases** to display the **Link to AppDynamics for Databases**.

AppDynamics for Databases looks for a database match in its repository and displays the database platform window.

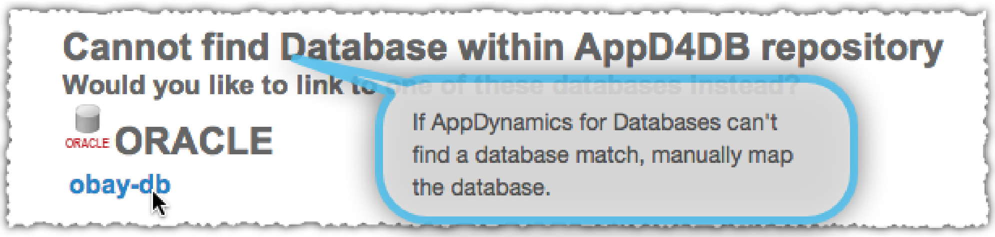


Displayed rows: 6 Export... | Reset

Entity	Executions	Avg Time	Elapsed	Weighted (%)
select count(1) count from item it1, item it2	202	00:00:22.63	01:16:12.00	99.78
select sql_id, executions, CPU_TIME, elapsed_time, rows_processed, buf....	9	00:00:00.44	00:00:04.00	0.09
Redo or DDL	0	N/A	00:00:02.00	0.04
select name, value from v$parameter	6	00:00:00.33	00:00:02.00	0.04
select sess.sid as Id, sess.username as username, nvl(sess.SCHEMANAME,....	59	00:00:00.02	00:00:01.00	0.02
select sql_text, command_type from v$sqltext where sql_id = :1 order b....	6	00:00:00.17	00:00:01.00	0.02

Total Elapsed Time: 01:16:22.00

2. If it doesn't find a match, you can select your database from a list and manually map it. You only need to map the database once. The next time you link, the database platform window displays automatically.



Cannot find Database within AppD4DB repository

Would you like to link to one of these databases instead?

ORACLE

obay-db

If AppDynamics for Databases can't find a database match, manually map the database.

For information on manually mapping your database, see Manually Map a Database.

3. See Monitor Databases for more information on monitoring with AppDynamics for Databases.

Learn More

Introduction to AppDynamics for Databases

Monitor Databases

Tutorials for .NET

Quick Tour of the User Interface

Overview Tutorials for .NET

Quick Tour of the User Interface

Manual Installation and Configuration

Administer App Agents for .NET

- AppDynamics Applications and IIS Applications
- Log Files
- Additional App Agent for .NET Administration Topics

AppDynamics Applications and IIS Applications

Usually there is not a 1-to-1 correspondence between an AppDynamics business application and an IIS application. A typical AppDynamics configuration assigns one IIS application per tier, and multiple tiers to a single AppDynamics business application. For example, a front end is a single tier, various services are represented as another tier or tiers, and they all belong to a single business application. See Name Business Applications, Tiers, and Nodes.

Log Files

The configuration file that controls log files for the App Agent for .NET is located at:

```
C:\Program Files\AppDynamics\AppDynamics .NET Agent\AppDynamicsAgentLog.config
```

The configuration file uses NLog rules. See http://nlog-project.org/.

Additional App Agent for .NET Administration Topics

Disable Instrumentation for an IIS Application Pool

- When to Consider Disabling Instrumentation
    - To disable an IIS application pool or pools

When to Consider Disabling Instrumentation

By default when you install the App Agent for .NET on a machine and use automatic tier naming, the agent instruments every IIS application. You may not need to monitor all application pools.

To disable an IIS application pool or pools

1. Open the config.xml file as administrator and edit the file as follows:

Windows Server 2008 and later

```
%ProgramData%\AppDynamics\DotNetAgent\Config\config.xml
```

Windows Server 2003

```
%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Config\config.
```

2. Add the Application Pools block as a child of the IIS element. See App Agent for .NET Configuration Properties.

```
<!-- Disable instrumentation for an application pool -->
  <application-pools>
    <!-- Do not instrument applications in DefaultAppPool when "enabled"
attribute is set to false -->
    <application-pool name="DefaultAppPool" enabled="false" />
  </application-pools>
```

Set the Application Pool element name attribute to the application pool name. This example disables instrumentation for the DefaultAppPool. You may add multiple Application Pool elements.

3. Restart the AppDynamics.Agent.Coordinator.

4. Restart IIS.

Naming Conventions for .NET Nodes

This topic describes the conventions that the App Agent for .NET uses to automatically name nodes.

Node Naming Syntax

Node naming relies on the application name to directly link nodes to an application. This naming convention clarifies node names for applications in an application pool hosting multiple applications and node names for applications in a recycle process.

If you are upgrading to 3.8 from the App Agent for .NET version 3.7.7 or earlier, the new node naming convention takes effect upon restart. Nodes named under the old scheme become historical nodes with no correlation to the new nodes.

Syntax for IIS Nodes

The naming pattern for IIS nodes is as follows:

```
<machine-name>-<tier>-<site>/<app>
```

- We omit <app> when the application is the default site root application.
- We omit <tier> when the tier name is the same as the site name.

For example:

```
WIN-86M7CEJO6P5-Order Server-OrderSvc
```

- WIN-86M7CEJO6P5 is the machine name.
- Order Server is the tier name.

- OrderSvc is the site name. The application is the default app in the site root, so we omit it.

  Different .NET versions of the same application have their own versions of the CLR and run on independent processes. Therefore they will show up as different nodes.

Machine Name

Machine name is the NetBIOS name of the local computer where the application runs.

Tier

Tier is the name of the logical tier for the application. See Logical Model.

Site/App

Site is the IIS Site name.
App is the virtual path within the site.

For example:

```
WIN-86M7CEJO6P5-Order Server-Store/ProcessOrder
```

- Store is the IIS Site name.
- ProcessOrder is the application name within the site.

Syntax for IIS Web Garden Nodes

The syntax for web gardens is the same as IIS Nodes, except that we append a process index to differentiate between the worker processes in the web garden.

```
<machine-name>-<tier>-<site>/<app>-<process-index>
```

Process Index

The process index represents the zero-based index of the process. For a web garden-IIS-hosted application with five worker processes, the index could be 0, 1, 2, 3 or 4.

Don't confuse the process index with the Windows process id.

When IIS first launches web garden processes, the agent assigns a sequential index to each process. However, as IIS recycles processes, the agent reuses the available indexes freed by terminated processes. This means there will likely be no correlation between the index sequence and the chronological start of the process.

There are edge cases where you may see more nodes than the maximum number of worker processes. This can happen when a long-running request prevents a process from shutting down before its replacement launches.

Syntax for Windows Service Nodes

The naming pattern for Windows service nodes is as follows:

```
<machine-name>-<tier>-<service-name>
```

- We omit <tier> when the tier name is the same as the service name.

Machine Name

Machine name is the NetBIOS name of the local computer where the Windows service runs.

Service Name

Service name is the same as **Service name** in the service properties window.

For Example:

```
WIN-86M7CEJO6P5-MyWindowsService
```

- WIN-86M7CEJO6P5 is the machine name.
- MyWindowsService is the Service name.

Syntax for Standalone Application Nodes

The naming pattern for standalone applications is as follows:

```
<machine-name>-<tier>-<executable-name>
```

- We omit <tier> when the tier name is the same as the executable name.

Machine Name

Machine name is the NetBIOS name of the local computer where the standalone application runs.

Executable Name

Executable name is the standalone application executable file name.

For Example:

```
WIN-86M7CEJO6P5-MyStandaloneApp.exe
```

- WIN-86M7CEJO6P5 is the machine name.
- MyStandaloneApp.exe is the executable file name.

Maximum Number of Nodes Generated

Below is the algorithm for calculating how many nodes will be generated. It refers to all nodes that are alive, as historical node counts can change via retention and deletion time frames as well as manual deletion, etc.

IISApp1_AppPool_MaxWorkerProcesses
+
IISApp2_AppPool_MaxWorkerProcesses
+...
IISAppN_AppPool_MaxWorkerProcesses
+...
Self-Hosted Process (Windows Service, Console Application, etc)
=
Number of Nodes

In the case of a webgarden, by default AppDynamics flags a node as historical within five minutes after it recycles. Wait 5 minutes for a recycled node to disappear from the application flow map.

App Agent for .NET Configuration Properties

Configure the App Agent for .NET properties using a single configuration file. This topic describes the structure and content of the config.xml file to help you customize agent behavior. To use the .NET Agent Configuration Utility, see Configure the App Agent for .NET.

Overview

The App Agent for .NET uses a single configuration file to control agent behavior: Controller connectivity, machine agent operations, and app agent functionality for IIS applications, Windows services, and standalone applications such as console applications, WinForms, or WPF. Benefits of the unified configuration file include:

- Maintain agent configurations separately from web.config files.
- Enable instrumentation of Windows services and standalone applications without

environment variables.
- Control agent behavior for specific applications with hierarchical configuration.

To configure connection to the Controller, See Controller Element.

To configure the .NET machine agent, See Machine Agent Element.

To configure app agents for IIS applications, see App Agents - IIS Element.

To configure an app agent for a Windows service, see App Agents - Windows Services Element.

To configure an app agent for a standalone application, see App Agents - Standalone Applications Element.

Where to Configure App Agent Properties

Configure the agent properties in the config.xml file in the agent **Config** directory. If you run the .NET Agent Configuration Utility, it writes the config.xml to the following locations:

Windows Server 2008 and later

```
%ProgramData%\AppDynamics\DotNetAgent\Config\config.xml
```

Windows Server 2003

```
%AllUsersProfile%\Application Data\AppDynamics\DotNetAgent\Config\config.xml
```

Sample config.xml files install to the following location:

```
%ProgramFiles%\AppDynamics\AppDynamics .NET Agent\SampleConfigurations
```

⚠ After you edit the config.xml, you must restart the AppDynamics.Agent.Coordinator service. Then restart IIS, your Windows service, or standalone application for your instrumentation changes to take effect.

Minimal .NET App Agent controller-info.xml File

The most basic configuration demonstrates the required sections for agent configuration. This sample instruments all IIS applications using the automatic element (`<automatic />`). No Windows services or standalone applications are instrumented.

```
<?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="MyDotNetApplication" />
  </controller>
  <machine-agent />
  <app-agents>
    <IIS>
      <automatic />
    </IIS>
  </app-agents>
</appdynamics-agent>
```

AppDynamics Agent Element

The Appdyanmics Agent element is the root container element for configurations in the config.xml.

Required Element: `<appdynamics-agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">`

Controller Element

The Controller element is a child of the AppDynamics Agent element. It specifies the connection information for the AppDyanmics Pro Controller.

The App Agent for .NET only supports configuration of one Controller and business application per server. Use tiers to organize different applications you instrument on a single server. See Logical Model.

Required Element: `<controller host="mycontroller.mycompany.com" port="8090" ssl="false">`

Controller host attribute

The Controller host attribute indicates the host name or the IP address of the AppDynamics Controller. For an on-premise Controller, use the value for Application Server Host Name you provided when you installed the Controller. If you use the AppDynamics SaaS Controller, see the Welcome email from AppDynamics.

Type: String
Default: None
Required: Yes

Controller port attribute

The Controller port attribute specifies the HTTP(S) port of the AppDynamics Controller. If the Controller ssl attribute is set to true, specify the HTTPS port of the Controller; otherwise specify the HTTP port.

Type: Positive Integer
Default: 8090

For On-premise installations, the defaults are port 8090 for HTTP and port 8181 for HTTPS.

For the SaaS Controller, use port 80 for HTTP or port 443 for HTTPS.

Required: Yes

Controller ssl attribute

To enable encryption over SSL between the agent and the Controller, set the Controller ssl attribute to "true".

Type: Boolean
Default: false
Required: No

Controller Application Element

The Controller Application element is a child of the Controller element. It indicates the name of the logical business application you see in the Controller interface.

Required Element: `<application name="MyDotNetApplication"/>`

Application name attribute

Set the application name attribute to the business application you use in the controller. If the application name does not exist, the Controller will create it when the agent registers. All instrumented applications in the config.xml register with the same business application in the Controller. See Logical Model.

Type: String
Default: None
Required: Yes

You specify a Tier for individual applications in the App Agents Element.

Account Element

The Account element is a child of the Controller element. If the AppDynamics Controller runs in multi-tenant mode or if you use the AppDynamics SaaS Controller, specify the account name and account access key for the agent to authenticate with the Controller. If you are using the AppDynamics SaaS Controller, the account name is provided in the Welcome email from AppDynamics.

Optional Element: `<account name="mycontroller.saas.appdynamics.com" password="myaccesskey"/>`

Account name attribute

The account name attribute indicates to the account name for the SaaS or multi-tenant Controller.

Type: String
Default: None

Required: Only for SaaS or multi-tenant Controllers

Account password attribute

The account password attribute indicates the access key for the SaaS or multi-tenant Controller.

Type: String
Default: None
Required: Only for SaaS or multi-tenant Controllers

Proxy Element

The Proxy element is a child of the Controller element. Use it to configure connection to the Controller through a proxy server with no authentication.

Optional Element: `<proxy host="proxy-name" port="3128" enabled="true"/>`

Proxy host attribute

The proxy host attribute indicates the proxy server host name or IP address.

Type: String
Default: None
Required: host is required for the proxy element

Proxy port attribute

The proxy port attribute indicates the proxy server port.

Type: Positive Integer
Default: None
Required: port is required for the proxy element

Proxy enabled attribute

To enable Controller access through a proxy server, set the proxy enabled attribute to "true".

Type: Boolean
Default: false
Required: No

Machine Agent Element

The Machine Agent element is a child of the AppDynamics Agent element. An empty Machine Agent element enables the default instrumentation for the .NET machine agent (See Monitor CLRs and Monitor IIS). Enable optional additional Microsoft Performance Counters or App Agent for .NET instrumentors as children of the Machine Agent element.

Required Element: `<machine-agent/>`

Performance Counters Element

The Performance Counters element is a child of the Machine Agent element. It is a container for all performance counters.

Optional Element: `<perf-counters>`

Performance Counter element

The Performance Counter element is a child of the Performance Counters element. For a list of performance counters to enable, see Performance Counters in the .NET Framework.

Optional Element: `<perf-counter cat="category" name="name" instance="instance"/>`

Performance Counter cat attribute

The performance counter cat attribute indicates the performance counter category.

Type: String
Default: None
Required: Category is required for the Performance Counter element.

Performance Counter name attribute

The performance counter name attribute indicates the performance counter name.

Type: String
Default: None
Required: Name is required for the Performance Counter element.

Performance Counter instance attribute

The performance counter instance attribute is the performance counter instance value.

Type: String
Default: None
Required: Instance is required for the Performance Counter element.

Sample Machine Agent Configuration with Performance Counters

```
<machine-agent>
    <!-- Additional machine level Performance Counters -->
    <perf-counters>
      <perf-counter cat="Network Interface" name="Bytes Sent" instance="Local
Area Connection"/>
    </perf-counters>
  </machine-agent>
```

Instrumentation Element

The Instrumentation element is a child of the Machine Agent element. It allows you to enable additional App Agent for .NET instrumetors such as thread correlation or correlation for .NET remoting.

Optional Element: `<instrumentation>`

Instrumentor element

The **Instrumentor** element is a child of the Instrumentation element. The instrumentor element specifies the App Agent for .NET instrumentor to implement.

Optional Element: `<instrumentor name="instrumentor name" enabled="true"/>/>`

Instrumentor name attribute

The instrumentor name attribute indicates the instrumentor name.

Type: String
Default: None
Required: Name is required for the Instrumentor element.

Instrumentor enabled attribute

Set the instrumentor enabled attribute to "true" to enable instrumentation.

Type: Boolean
Default: false
Required: No.

The current configuration syntax is **enabled="true"**. Versions prior to 3.7.8 used disabled="false".

Sample Machine Agent Configuration with Thread Correlation Instrumentors

```
<machine-agent>
    <!--Enable thread correlation-->
    <instrumentation>
        <instrumentor name="ThreadCorrelationThreadPoolCLR2Instrumentor"
enabled="true"/>
        <instrumentor name="ThreadCorrelationThreadPoolCLR4Instrumentor"
enabled="true"/>
        <instrumentor name="ThreadStartCLR2Instrumentor" enabled="true"/>
        <instrumentor name="ThreadStartCLR4Instrumentor" enabled="true"/>
    </instrumentation>
  </machine-agent>
```

Additional App Agent for .NET Instrumetors

- See Enable Thread Correlation for .NET.
- See Enable Correlation for .NET Remoting.
- See Enable Instrumentation for WCF Data Services.

App Agents Element

The App Agents element is a child of the AppDynamics Agent element. It is a container for app agent configurations for IIS applications, Windows services, and standalone applications.

Required Element: `<app-agents enabled="true">`

App agents enabled attribute

To disable application monitoring on the server, set the app agents enabled attribute to "false".

Type: Boolean
Default: true
Required: No

App Agents - IIS Element

The **IIS** element is a child of the App Agents element. There are three options to configure IIS applications:

- Automatic configuration
- Application pool configuration
- Application configuration

The settings for any application pool apply to all applications within the app pool unless the individual application has a specific configuration.

Explicit child-level configurations override parent-level configurations. Otherwise, children inherit parent configurations.

Optional Element: `<IIS>`

IIS Automatic Instrumentation Element

The Automatic element is a child of the IIS element. Use the Automatic element to enable or disable automatic instrumentation for all IIS apps. You can configure automatic instrumentation and manual instrumentation both. Manual configurations override automatic ones.

Optional Element: `<automatic enabled="false" />`

Automatic enabled attribute

Set the automatic enabled attribute to "true" to enable instrumentation for all IIS applications. This is the default setting if you use the .NET Agent Configuration Utility automatic configuration option. To disable automatic instrumentation for all IIS applications, set the value to "false".

Type: Boolean
Default: true
Required: No

IIS Application Pools Element

The IIS Application Pools element is a child of the IIS element. It is a container element for all the IIS application pools you configure for instrumentation.
Optional Element: `<application-pools>`

IIS Application Pool Element

The Application Pool element is a child of the Application Pools element.. You may have multiple application pool elements distinguished by the name attribute. Use the application pool element to configure the app agent for all applications within an application pool. For more information on IIS application pools, see Managing Application Pools in IIS.

⚠ Application-specific configurations in the IIS Applications element override application pool configurations.

Optional Element: `<application-pool name="DefaultAppPool" enabled="false">`

Application pool name attribute

The application pool name attribute indicates the name of the IIS Application Pool.

Type: String
Default: None
Required: Name is required for the Application Pool element.

Application pool enabled attribute

Set the application pool enabled attribute to "false" to disable instrumentation for all applications in the application pool. Set the value to "true" to instrument all applications in the application pool.

Type: Boolean
Default: None. Defaults to true if not specified.
Required: No

Application Pool Tier Element

The Tier element is a child of the Application Pool element. If you enable instrumentation for an Application pool, you must use a Tier element to assign the pool's applications to a tier in the Controller. See Logical Model.

Required Element: `<tier name="Inventory" />`

Tier name attribute

Use the tier name attribute to specify the tier.

Type: String
Default: None
Required: Yes

IIS Applications Element

The **IIS Applications** element is a child of the IIS element. It is a container element for all the IIS applications you configure for instrumentation.
Optional Element: `<applications>`

Application Element

The Application element is a child of the Applications element. Use multiple application elements to instrument different sites and applications. To learn more about IIS sites and applications, see Understanding Sites, Applications, and Virtual Directories on IIS 7 and Above.

Optional Element: `<application path="/" site="FirstSite" port="8008">`

Application site attribute

The application site attribute indicates the root site in IIS for the application.

Type: String
Default: None
Required: Site is required for the Application element.

Application path attribute

The application path attribute indicates the application's path relative to the root site. Use the forward slash to indicate the root site and instrument all children applications. Use the path to an application to instrument the specific application and any children.

For example: Site1 hosts two applications AppX and AppY. To instrument Site 1, AppY and AppZ, set the path to "/". To instrument AppY, but not AppZ, set the path to "/AppY".

Type: String
Default: /
Required: Path is required for the Application element.

Application port attribute

For cases where two or more sites in IIS 6 have the same site name, set the site port attribute to differentiate between the sites.

Type: Positive Integer
Default: None
Required: No

Application enabled attribute

In certain cases you may want to enable instrumentation for a parent application, but disable it for a child application. In this case create an Application element for the child application to disable and set the application enabled attribute to to "false".

Type: Boolean
Default: true
Required: No

Application Tier Element

The Tier element is a child of the Application element. If you enable instrumentation for an application, your must use a Tier element to assign the application to a tier in the Controller. See Logical Model.

Required Element: `<tier name="Consumer" />`

Tier name attribute

The tier name attribute indicates the business application tier.

Type: String
Default: None
Required: Yes

Sample IIS Application Configuration

```
<IIS>
      <!-- Automatic instruments all IIS applications when enabled -->
      <automatic enabled="false" />

      <!-- Application Pool agent configurations -->
      <application-pools>
        <!-- Do not instrument applications in DefaultAppPool when "enabled"
attribute is set to false -->
        <application-pool name="DefaultAppPool" enabled="false">
          <tier name="Tier Name"/>
        </application-pool>

        <!-- Instrument applications in the OtherAppPpool and assign them to the
Inventory tier -->
        <application-pool name="OtherAppPool">
          <tier name="Inventory"/>
        </application-pool>
      </application-pools>

      <applications>
        <!-- Instrument all applications in the First Site -->
        <application path="/" site="FirstSite">
          <tier name="Order"/>
        </application>
        <!-- Instrument the /app application and child apps in the Second Site
-->
        <!-- but not the root Second Site application -->
        <application path="/app" site="SecondSite">
          <tier name="Consumer"/>
        </application>
      </applications>
    </IIS>
```

App Agents - Windows Services Element

The Windows Services element is a child of the App Agents element. It is a container element for all the Windows Services you configure for instrumentation.

For instructions to instrument Windows services, see Enable the App Agent for .NET for Windows Services.

Optional Element: `<windows-services>`

Windows Service Element

The Windows Service element is a child of the Windows Services element. It specifies a Windows service to instrument.

Optional Element: `<windows-service name="MyWindowsService">`

Service name attribute

The service name attribute indicates the service name for the Windows service to instrument.
⚠ Use the value of the **Service name** from the service properties window. Do not use the **Display name**.

Type: String
Default: None
Required: Yes

Service app-domain-name attribute

For applications with multiple application domains, the app-domain-name attribute enables you to limit instrumentation to specific application domains. See Configure Application Domain Monitoring
.

Type: String
Default: None
Required: No

Windows Service Tier Element

The **Tier** element is a child of the Windows Service element. If you enable instrumentation for an application, your must use a Tier element to assign the application to a tier in the Controller. See L
ogical Model.

Required Element: `<tier name="Consumer" />`

Tier name attribute

The tier name attribute indicates the business application tier.

Type: String
Default: None
Required: Yes

Sample Windows Service Configuration

```
<windows-services>
      <windows-service name="MyWindowsService">
        <tier name="Service Tier"/>
      </windows-service>
    </windows-services>
```

App Agents - Standalone Applications Element

The Standalone Applications element is a child of the App Agents element. It is a container element for all the standalone applications you configure for instrumentation.

ⓘ For instructions to instrument standalone applications, see Enable the App Agent for .NET for Standalone Applications.

Optional Element: `<standalone-applications>`

Standalone Application Element

The Standalone Application element is a child of the Standalone Applications element. It specifies a standalone application to instrument.

Optional Element: `<standalone-application executable="MyWindowsApplication.exe">`

Standalone Application executable attribute

The standalone application executable attribute specifies the file name of the Windows application to instrument.

⚠ Only use the application file name. Do not include the full path to the file. For example, if you want to instrument, C:\Program Files\My Application\MyApp.exe, the executable value is "MyApp.exe". The file extension is optional, so "MyApp" also works.

Type: String
Default: None
Required: Yes

Standalone application app-domain-name attribute

For applications with multiple application domains, the app-domain-name attribute enables you to limit instrumentation to specific application domains. See Configure Application Domain Monitoring
.

Type: String
Default: None
Required: No

Standalone Application Tier Element

The Tier element is a child of the Standalone Application element. If you enable instrumentation for an application, your must use a Tier element to assign the application to a tier in the Controller. See Logical Model.

Required Element: `<tier name="Consumer" />`

Tier name attribute

The tier name attribute indicates the business application tier.

Type: String
Default: None
Required: Yes

Sample Standalone Application Configuration

```
<standalone-applications>
      <standalone-application executable="MyWindowsApplication.exe">
        <tier name="StandaloneApplication Tier"/>
      </standalone-application>
    </standalone-applications>
```

Learn More

- Logical Model
- Name Business Applications, Tiers, and Nodes
- Configure the App Agent for .NET
- Enable the App Agent for .NET for Windows Services
- Enable the App Agent for .NET for Standalone Applications

Troubleshoot .NET Application Problems

Troubleshoot Slow Response Times for .NET

How Do You Know Response Time is Slow?

There are two primary ways you can learn that your application's response time is slow: receiving an alert and looking at an Application Dashboard.

You Received an Alert

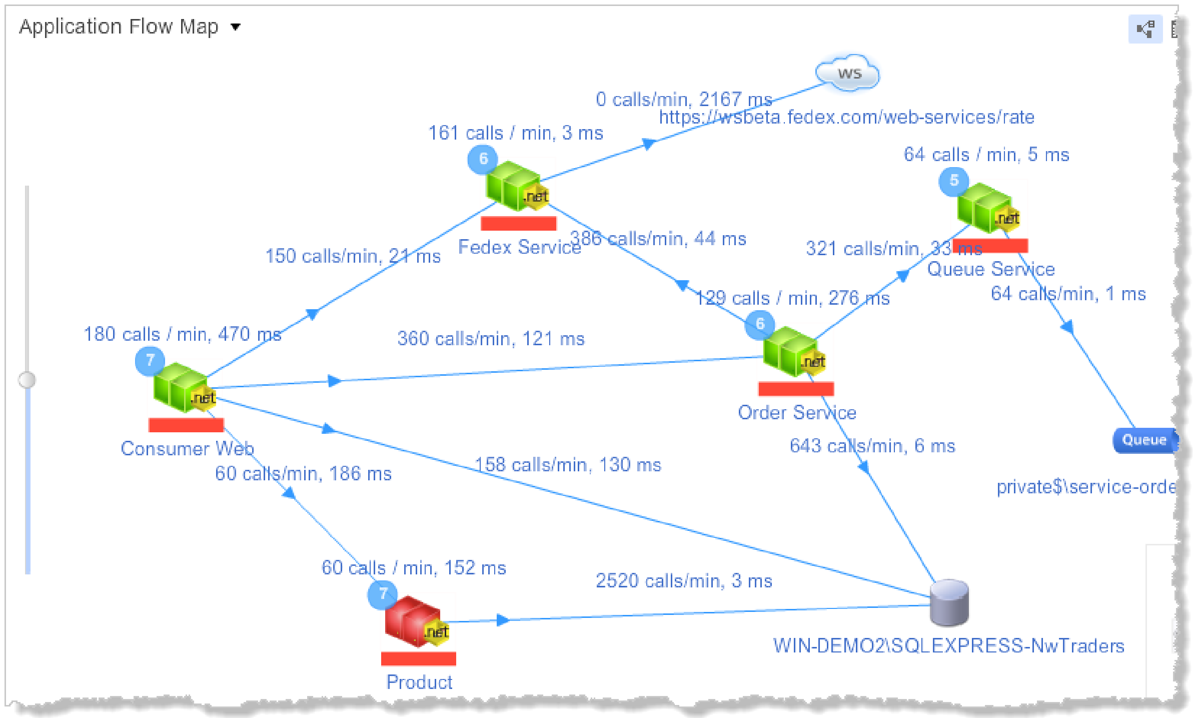
If you have received an email alert from AppDynamics that was configured through the use of Heal th Rules, Policies, Actions or Workflow Overview, the email message provides a number of details about the problem that triggered the alert. See Email Notifications.

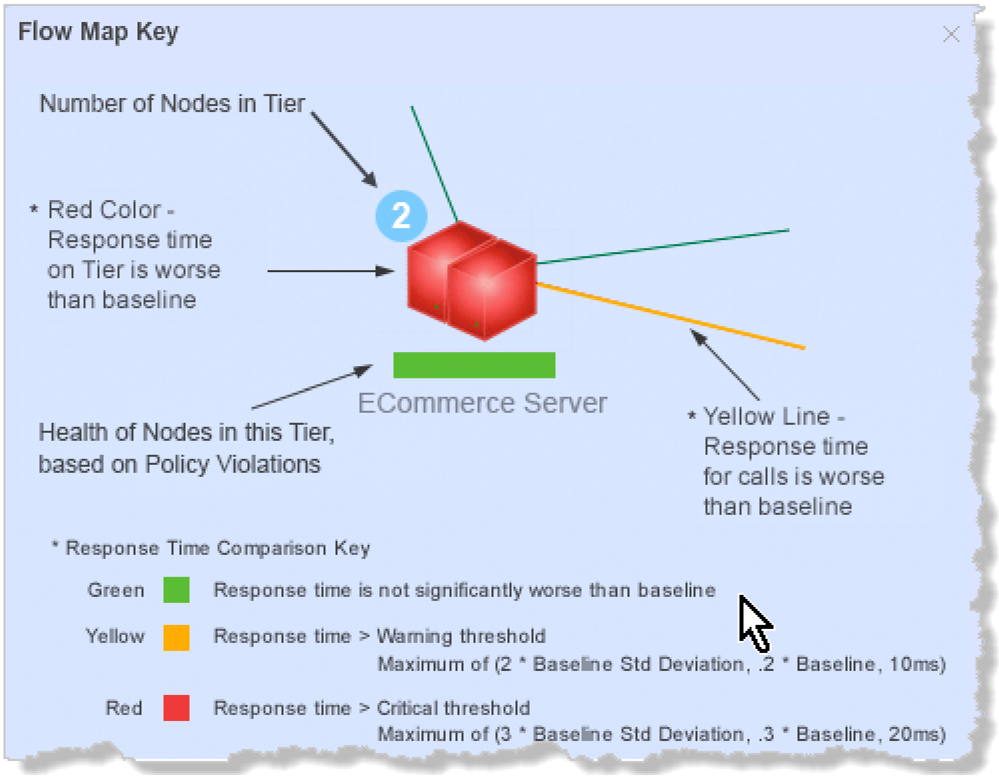
If the problem is related to slow response time, see Initial Troubleshooting Steps.

You are Viewing the Application Dashboard for a Business Application

NOTE: If you know the slow response time relates to a particular business transaction, e.g. an internal tester reported "Searching for a hotel is slow," skip to the answer No in Step 2.

1. Display the Application Dashboard (flow map). Look for lines that are yellow or red or tiers that are not entirely green.




Flow Map Key
Number of Nodes in Tier
2
* Red Color - Response time on Tier is worse than baseline
ECommerce Server
Health of Nodes in this Tier, based on Policy Violations
* Yellow Line - Response time for calls is worse than baseline

* Response Time Comparison Key

Green	🟩	Response time is not significantly worse than baseline
Yellow	🟨	Response time > Warning threshold Maximum of (2 * Baseline Std Deviation, .2 * Baseline, 10ms)
Red	🟥	Response time > Critical threshold Maximum of (3 * Baseline Std Deviation, .3 * Baseline, 20ms)

- If multiple tiers are red or yellow, the problem might be related to a backend (database or other remote service). Skip to Step 3.
- Otherwise, begin troubleshooting with Initial Troubleshooting Steps.

Need more help?

- Application Dashboard
- Flow Maps

Initial Troubleshooting Steps

In some cases, the source of your problem might be easily diagnosed by choosing **Troubleshoot -> Slow Response Times** in the Navigation Pane. See Troubleshoot Slow Response Times.

If you've tried to diagnose the problem using those techniques and haven't found the problem, use the following troubleshooting methodology to find other ways to determine the root cause of your issue.

Troubleshooting Methodology

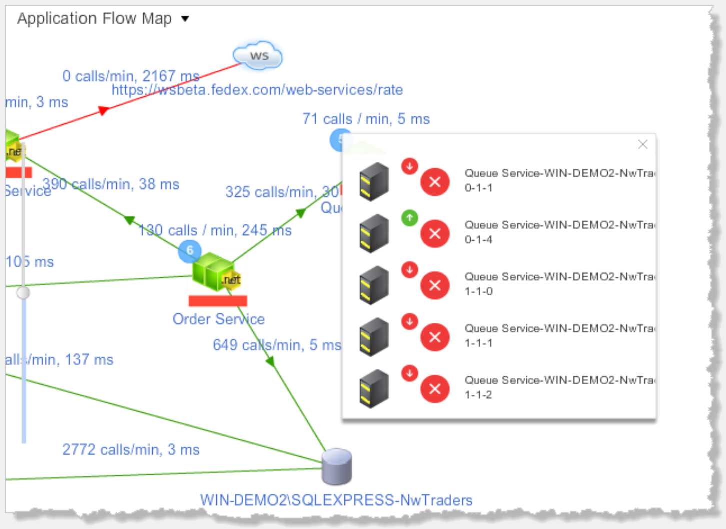
Step 1 - All nodes?

Is the problem affecting all nodes in a slow tier?

How do I know?

How do I know if the problem is affecting all nodes?

- In the Application or Tier Flow Map, click the number that represents how many nodes are in the tier. This provides a quick overview of the health of each node in the tier. The small circle icon indicates whether the server is up with the agent reporting, and the larger circle icon indicates Health Rule violation status.

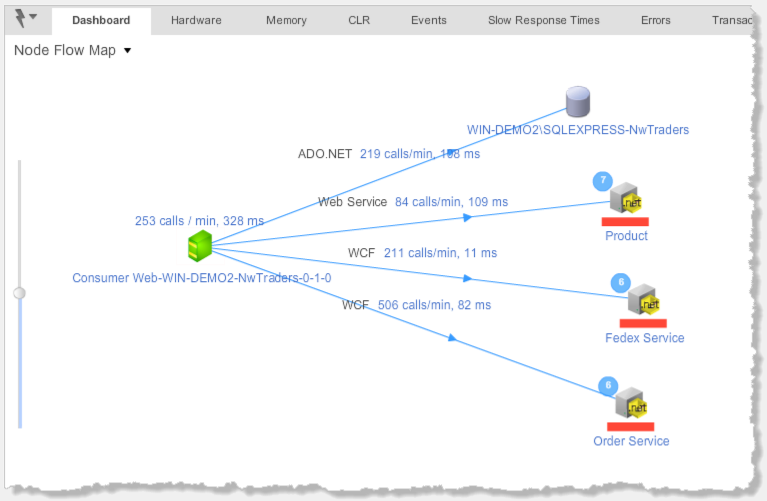

Application Flow Map
WS
0 calls/min, 2167 ms
https://wsbeta.fedex.com/web-services/rate
min, 3 ms
71 calls / min, 5 ms
Service
390 calls/min, 38 ms
325 calls/min, 30
130 calls / min, 245 ms
105 ms
Order Service
649 calls/min, 5 ms
alls/min, 137 ms
2772 calls/min, 3 ms
WIN-DEMO2\SQLEXPRESS-NwTraders
Queue Service-WIN-DEMO2-NwTra 0-1-1
Queue Service-WIN-DEMO2-NwTra 0-1-4
Queue Service-WIN-DEMO2-NwTra 1-1-0
Queue Service-WIN-DEMO2-NwTra 1-1-1
Queue Service-WIN-DEMO2-NwTra 1-1-2

If all the nodes are yellow or red, the answer to the question in Step 1 is Yes. Otherwise, the answer is No.

Yes – Go to Step 2.

No – The problem is either in the node's hardware or in the way the software is configured on the node. If only one node in a tier is affected, the problem is probably not related to the application code.

- In the left navigation pane, click **Servers -> App Servers -> <slow tier> -> <problematic node>** to display the Node Dashboard flow map.



- Click the Dashboard tab to get a view of the overall health of the node.
- Click the Hardware tab to see if there is a hardware resource issue.
- Click the Memory tab and sort on various column headings to determine if there is a shortage of memory or other memory issue.

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- Node Dashboard

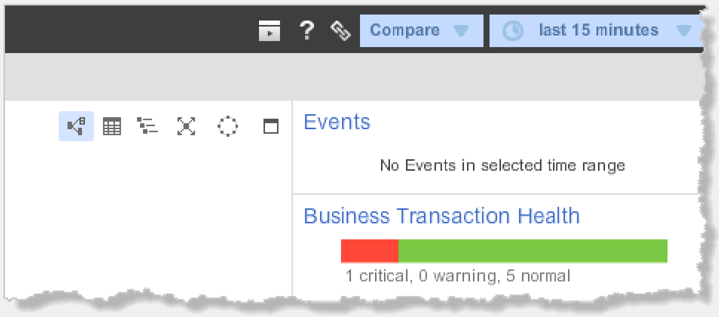
Step 2 - Most business transactions ?

Is the problem affecting most of the business transactions?

How do I know?

How do I know if the problem is affecting most of the business transactions?

1. On the Application Dashboard, look at the Business Transaction Health pane on the right side of the screen.



Events

No Events in selected time range

Business Transaction Health

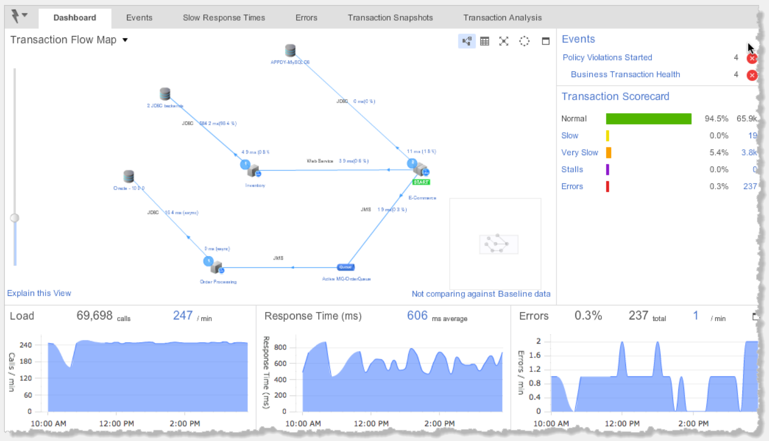
1 critical, 0 warning, 5 normal

If the bar representing business transaction health is primarily yellow or red, the answer to the question in Step 2 is Yes. Otherwise, the answer is No.

Yes – Go to Step 3.

No –

1. In the left navigation pane, click **Business Transactions**.
2. Sort by Health, Server Time, or other column headings to find the business transaction that is experiencing issues.
3. Double-click the problematic business transaction to see its Dashboard, then use the tabs to diagnose the problem.


Dashboard
Events
Slow Response Times
Errors
Transaction Snapshots
Transaction Analysis
Transaction Flow Map
Explain this View
Not comparing against Baseline data
Events
Policy Violations Started 4
Business Transaction Health 4
Transaction Scorecard
Load 69,698 calls 247 /min
Response Time (ms) 606 ms average
Errors 0.3% 237 total 1 /min
10:00 AM
12:00 PM
2:00 PM

Normal	94.5%	65.9k
Slow	0.0%	19
Very Slow	5.4%	3.8k
Stalls	0.0%	0
Errors	0.3%	237

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- Business Transaction Dashboard
- Business Transaction Monitoring
- Business Transactions List
- Transaction Snapshots

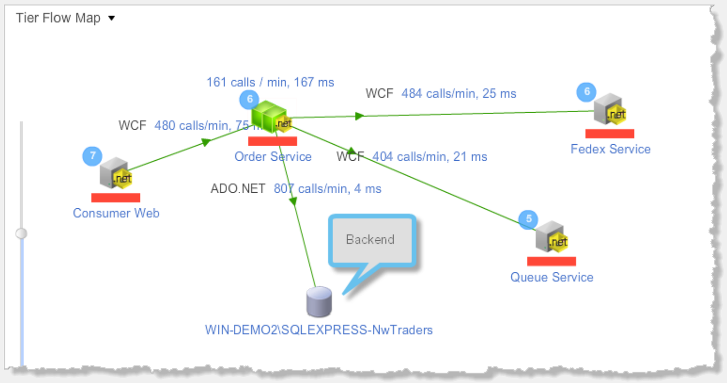
Step 3 - Backend problem?

Are the nodes linked to a backend (database or other remote service) that might be causing your problem?

How do I know?

How do I know if the nodes are linked to a backend (database or other remote service) that might be causing my problem?

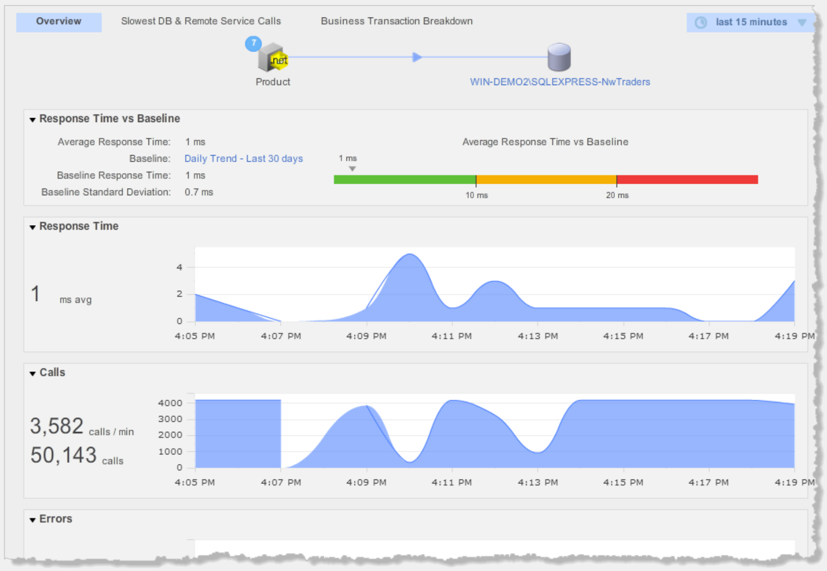
- Display the Tier Flow Map. If any nodes are linked to a backend, links to those backends are displayed in the flow map.



If a backend or the line connecting to a backend is red, the answer to the question in Step 3 is Yes. Otherwise, the answer is No.

Yes –

- Click the line connecting to the backend to see an information window about the backend. The contents of the information window vary depending on the type of backend. Use the various tabs to find the source of the issue.

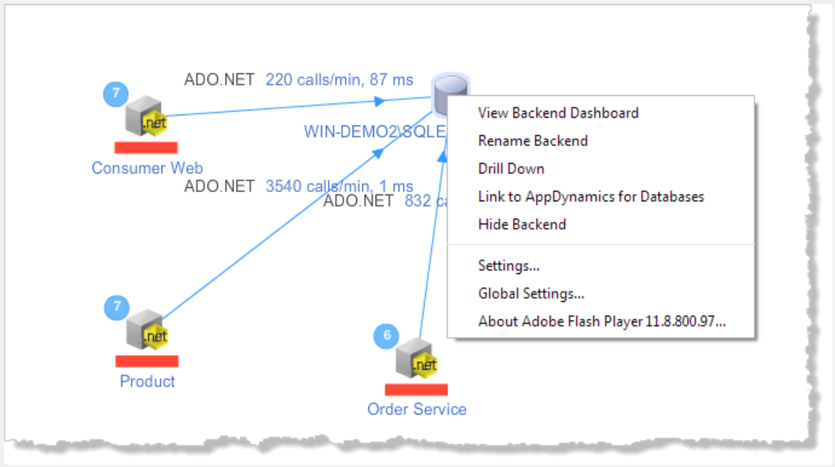

Overview
Slowest DB & Remote Service Calls
Business Transaction Breakdown
last 15 minutes
Product
WIN-DEMO2\SQLEXPRESS-NwTraders
Response Time vs Baseline
Average Response Time vs Baseline
Response Time
Calls
Errors

Average Response Time: 1 ms
Baseline: Daily Trend - Last 30 days
Baseline Response Time: 1 ms
Baseline Standard Deviation: 0.7 ms

1 ms avg

3,582 calls / min
50,143 calls

- If the backend is a database, right-click the database icon. You have a number of options that let you see the dashboard, drill down, etc. If you have AppDynamics for Databases, choose Link to AppDynamics for Databases. You can use AppDynamics for Databases to diagnose database issues.



You have isolated the problem and don't need to continue with the rest of the steps below.

No – Go to Step 4.

Need more help?

- Backend Monitoring
- Configure Backend Detection for .NET
- AppDynamics for Databases

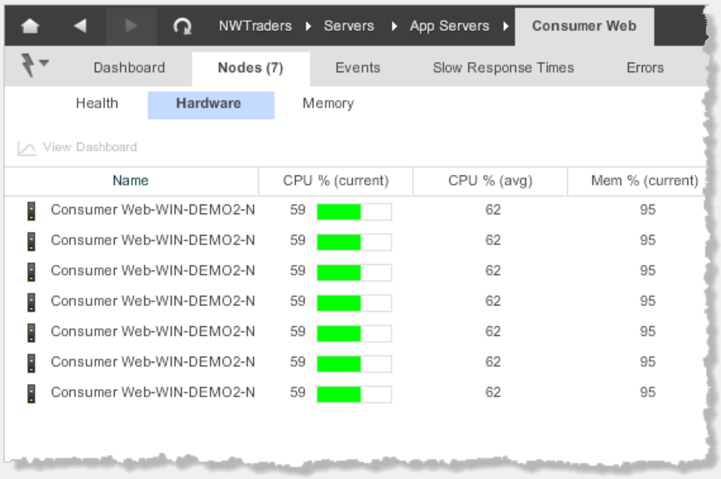
Step 4 - CPU saturated?

Is the CPU of the CLR saturated?

How do I know?

How do I know if the CPU of the CLR is saturated?

1. Display the Tier Flow Map.
2. Click the Nodes tab, and then click the Hardware tab.
3. Sort by CPU % (current).



Name	CPU % (current)	CPU % (avg)	Mem % (current)
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95
Consumer Web-WIN-DEMO2-N	59	62	95

If the CPU % is 90 or higher, the answer to the question in Step 4 is Yes. Otherwise, the answer is No.

Yes – Go to Step 5.

No – Review various metrics in the Metric Browser to pinpoint the problem.

In the left navigation pane, click **Servers -> App Servers -> <slow tier>**. Review these metrics in particular:

- ASP.NET -> Application Restarts
- ASP.NET -> Request Wait Time
- ASP.NET -> Requests Queued

- CLR -> Locks and Threads -> Current Logical Threads
- CLR -> Locks and Threads -> Current Physical Threads

- IIS -> Number of working processes
- IIS -> Application pools -> <Business application name> -> CPU%
- IIS -> Application pools -> <Business application name> -> Number of working processes
- IIS -> Application pools -> <Business application name> -> Working Set

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- Monitor .NET Applications

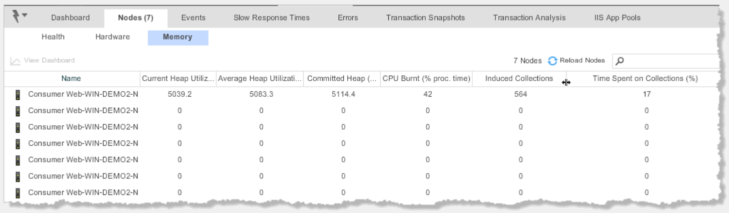
Step 5 - Significant garbage collection activity?

Is there significant garbage collection activity?

How do I know?

How do I know if there is significant garbage collection activity?

1. Display the Tier Flow Map.
2. Click the Nodes tab, and then click the Memory tab.
3. Sort by Time Spent on Collections (%) to see what percentage of processing time is being taken up with garbage collection activity.



Name	Current Heap Utiliz...	Average Heap Utilizati...	Committed Heap (...	CPU Burnt (% proc. time)	Induced Collections	Time Spent on Collections (%)
Consumer Web-WIN-DEMO2-N	5039.2	5083.3	5114.4	42	564	17
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0
Consumer Web-WIN-DEMO2-N	0	0	0	0	0	0

If Time Spent on Collections (%) is higher than acceptable (say, over 40%), the answer to the question in Step 5 is Yes. Otherwise, the answer is No.

Yes – Go to Step 6.

No – Use your standard tools to produce memory dumps; review these to locate the source of the problem.

You have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- Monitor .NET Applications

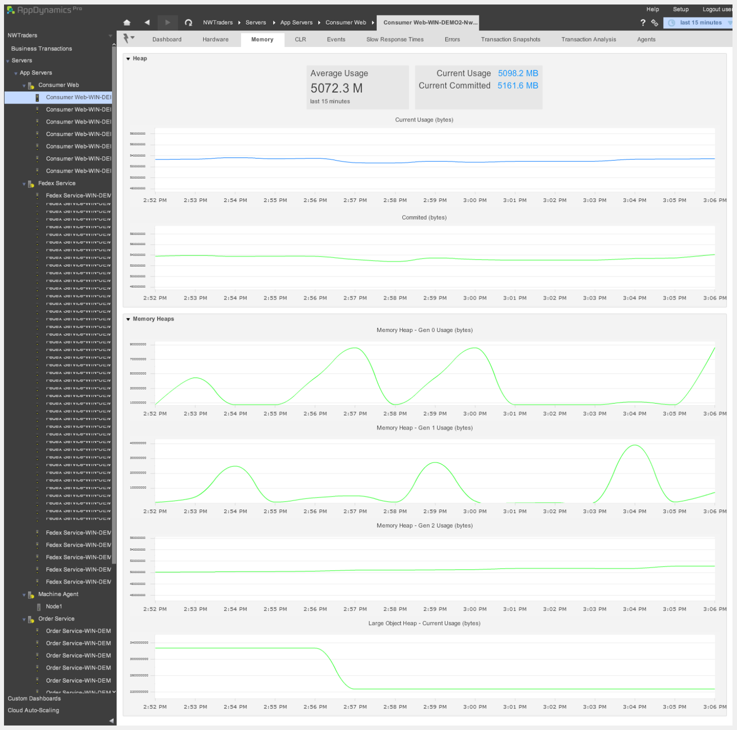
Step 6 - Memory leak?

Is there a memory leak?

How do I know?

How do I know if there is a memory leak?

1. From the list of nodes displayed in the previous step (when you were checking for garbage collecting activity), double-click a node that is experiencing significant GC activity.
2. Click the Memory tab, then review the committed bytes counter and the size of the Gen0, Gen1, Gen2 and large heaps.


Average Usage 5072.3 M
last 15 minutes

Current Usage 5098.2 MB
Current Committed 5161.6 MB

If memory is not being released (one or more of the above indicators is trending upward), the answer to the question in Step 6 is Yes. Otherwise, the answer is No.

Yes – Use your standard tools for troubleshooting memory problems. You can also review ASP.NET metrics; click **Servers -> App Servers -> <slow tier> -> ASP.NET**.

No – Use your standard tools to produce memory dumps; review these to locate the source of the problem.

Whether you answered Yes or No, you have isolated the problem and don't need to continue with the rest of the steps below.

Need more help?

- Monitor .NET Applications

None of the above?

If slow response time persists even after you've completed the steps outlined above, you may need to perform deeper diagnostics.

If you can't find the information you need on how to do so in the AppDynamics documentation, consider posting a note about your problem in a community discussion topic. These discussions are monitored by customers, partners, and AppDynamics staff. Of course, you can also contact AppDynamics support.

Need more help?

- AppDynamics Pro Documentation
- Community Discussion Boards (If you don't see AppDynamics Pro as a topic, click Sign In at the upper right corner of the screen.)