IIS Modules Overview

Article • 08/23/2022 • 8 contributors

Feedback

In this article

by IIS Team, Mike Volodarsky

Compatibility

Expand table

Version	Notes
IIS 7.0 and later	The features described in this article were introduced in IIS 7.0.
IIS 6.0 and earlier	The features described in this article were not supported prior to IIS 7.0.

Introduction

The IIS 7 and above Web server feature set is componentized into more than thirty independent modules.

A module is either a Win32 DLL (native module) or a .NET 2.0 type contained within an assembly (managed module). Similar to a set of building blocks, modules are added to the server in order to provide the desired functionality for your applications. Likewise, all IIS modules can be removed, or replaced with custom modules developed using the IIS C++ APIs, or the familiar ASP.NET 2.0 APIs.

This article describes common IIS module management tasks, and details each module including its configuration settings and the potential effect a module removal has on the Web server. The management examples are given using both the graphical IIS Manager and the **AppCmd** command line tool.

Prerequisites

To follow the steps in this document it is best to do a full install of IIS.

> Note
>
> Not all editions of Windows support all available IIS features. To see which features of IIS are supported on your operating system, you will need to consult the documentation which shipped with your version of Windows.

To Perform a Full Installation of IIS on Windows Vista, Windows 7, Windows 8, Windows 10 or Windows 11

1. Click **Start** and then click **Control Panel**.
2. In the **Control Panel**, click **Programs**.
3. Under **Programs and Features**, click **Turn Windows features on or off**.
4. In the **Windows Features** dialog, expand **Internet Information Services** and select all features. You may need to expand some categories to select all the features in that category.
5. Click **OK**.

To Perform a Full Install of IIS on Windows Server 2008/R2, Windows Server 2012/R2, or Windows Server 2016

1. Open **Server Manager** > **Roles** and select **Web Server (IIS)**.
2. Check all features under **Web Server**.

Getting Started with Modules

In order to add a module to the server, you must perform two steps:

1. Install a module on the server (native modules only).
2. Enable the module in an application.

The first step registers the module globally with the server, making it available within each server worker process. It is necessary only for native modules due to the trusted nature of native code, and is available only to administrators.

Note

A native module has unrestricted access to any resource available to the server worker process, just like an ISAPI filter or extension in previous versions. Because of this unrestricted access, you should install only native modules that come from a trusted source.

The second step enables the module to execute within a particular application and effectively allows the application administrator to control the server features enabled for the application. This step allows both installed native modules and managed modules to be enabled for each application.

To Install a Native Module

In order to install a native module, it must be registered with the server using one of the options below:

- Manually editing the IIS configuration store. In IIS 7.5 and later you can use the Configuration Editor in the IIS Manager.
- Using the IIS Manager
- Using the **AppCmd.exe** command line tool

All three of these options result in the module entry being added to the <globalModules> IIS configuration section, which can be set only at the server level. To examine the contents of this section, open the root configuration file located in `%windir%\system32\inetsrv\config\applicationhost.config`, and search for the string "<globalModules>".

After a full IIS installation, this section contains an entry for each of the native modules shipped with IIS, specifying a name and the path to the module DLL:

XML

```xml
<globalModules>
    <add name="DefaultDocumentModule" image="%windir%\system32\inetsrv\defdoc.dll" />
    <add name="DirectoryListingModule" image="%windir%\system32\inetsrv\dirlist.dll" />
    <add name="StaticFileModule" image="%windir%\system32\inetsrv\static.dll" />
    ...

</globalModules>
```

All of these modules are described in detail later in this document.

To Uninstall a Native Module

You can uninstall a native module if that module is no longer in use on the server, or if you would like to replace it with another module. Remove the corresponding module entry from the <globalModules> configuration list, and the associated entry in the `<modules>` configuration list using one of the following options:

- Manually editing the IIS configuration store. In IIS 7.5 and above you can use the Configuration Editor.
- Using the IIS Manager
- Using the **AppCmd.exe** command line tool

> Note
>
> Because the <globalModules> configuration section can be set only at the server level, you must be an administrator to uninstall a module.

Enabling a Module for an Application

A module must be enabled before it can provide service for a particular application. In order to enable a native module, it must first be installed on the server (see the previous section, To Install a Native Module).

A managed module does not require installation, and can be enabled directly for each application. This allows applications to include their managed modules directly within the application by registering them in the application's web.config file, and providing the implementation in /BIN or /App_Code directories.

To enable a module, do one of the following:

- Manually edit the IIS configuration store, either globally to enable the module for all applications on the server, or in a particular web.config file located within each application for which you would like to enable this module. In IIS 7.5 or above you can use the Configuration Editor.
- Use the IIS Manager
- Use the **AppCmd.exe** command line tool

All three of these options add the module entry to the `<modules>` IIS configuration section, which is can be set at both the server level and application level. Examine the contents of this section by opening the root configuration file located in `%windir%\system32\inetsrv\config\applicationhost.config`, and searching for the string "<modules>".

Unlike native modules, a managed module does not require adding an entry to the <globalModules> configuration section.

After a full IIS installation, the configuration section contains an entry for each of the modules (both managed and native) that shipped with IIS. The entry indicates that all these modules are enabled by default for all applications on the server. Each entry in this section specifies either the name of a native module that was previously installed on the server, or the name and .NET type of a managed module:

XML

```xml
<modules>
<add name="DefaultDocumentModule" />
<add name="DirectoryListingModule" />
<add name="StaticFileModule"/>
 ...

</modules>
```

Disabling a Module in an Application

Disable a module if that module is no longer in use in the application, or if you want to replace it with another module. To disable a module, remove the corresponding module entry from the `<modules>` configuration collection for a particular application where you do not want this module to execute. If the module is enabled at the server level, remove it there to disable it in all applications on the server by default. Use one of the following options to remove the module at the server level:

- Manually edit the <system.webServer>/<modules> configuration section in your application.
- Use the IIS Manager.
- Use the **AppCmd.exe** command line tool.

After the module is removed from the application, it will not be active in that application. However, if the module is native, it will still be loaded in the server worker process, and may be used by other applications that did not remove it.

ⓘ Note

The configuration section is unlocked by default if ASP.NET is installed. This default unlocking allows applications to disable both native and managed modules enabled globally, and to add new managed modules. This configuration section can be locked at the server level to prevent modification entirely, or prevent removal of specific module entries if desired by the Administrator.

Preconditions

There is another attribute on a module entry called precondition. The IIS core engine uses preconditions to determine when to enable a particular module. Performance reasons, for example, might determine that you only want to execute managed modules for requests that also go to a managed handler. The precondition in the following example (precondition="managedHandler") only enables the forms authentication module for requests that are also handled by a managed handler, such as requests to .aspx or .asmx files:

XML Copy

```xml
<add name="FormsAuthentication"type="System.Web.Security.FormsAuthenticationModule" preCon
```

If you remove the attribute precondition="managedHandler", Forms Authentication also applies to content that is not served by managed handlers, such as .html, .jpg, .doc, but also for classic ASP (.asp) or PHP (.php) extensions. See "How to Take Advantage of IIS Integrated Pipeline" for an example of enabling ASP.NET modules to run for all content.

You can also use a shortcut to enable all managed (ASP.NET) modules to run for all requests in your application, regardless of the "managedHandler" precondition. To enable all managed modules to run for all requests without configuring each module entry to remove the "managedHandler" precondition, use the **runAllManagedModulesForAllRequests** property in the `<modules>` section:

XML Copy

```xml
<modules runAllManagedModulesForAllRequests="true"/>
```

When you use this property, the "managedHandler" precondition has no effect and all managed modules run for all requests.

Querying, Adding and Removing Modules Using the IIS Manager

The IIS Manager provides an easy way to query, add or remove modules.

1. To open the IIS Manager, click **Start**, type **inetmgr** in the **Search** box, and then press ENTER.
2. Click the computer name of your IIS server.
3. Click the **Modules** icon in the **IIS** category.

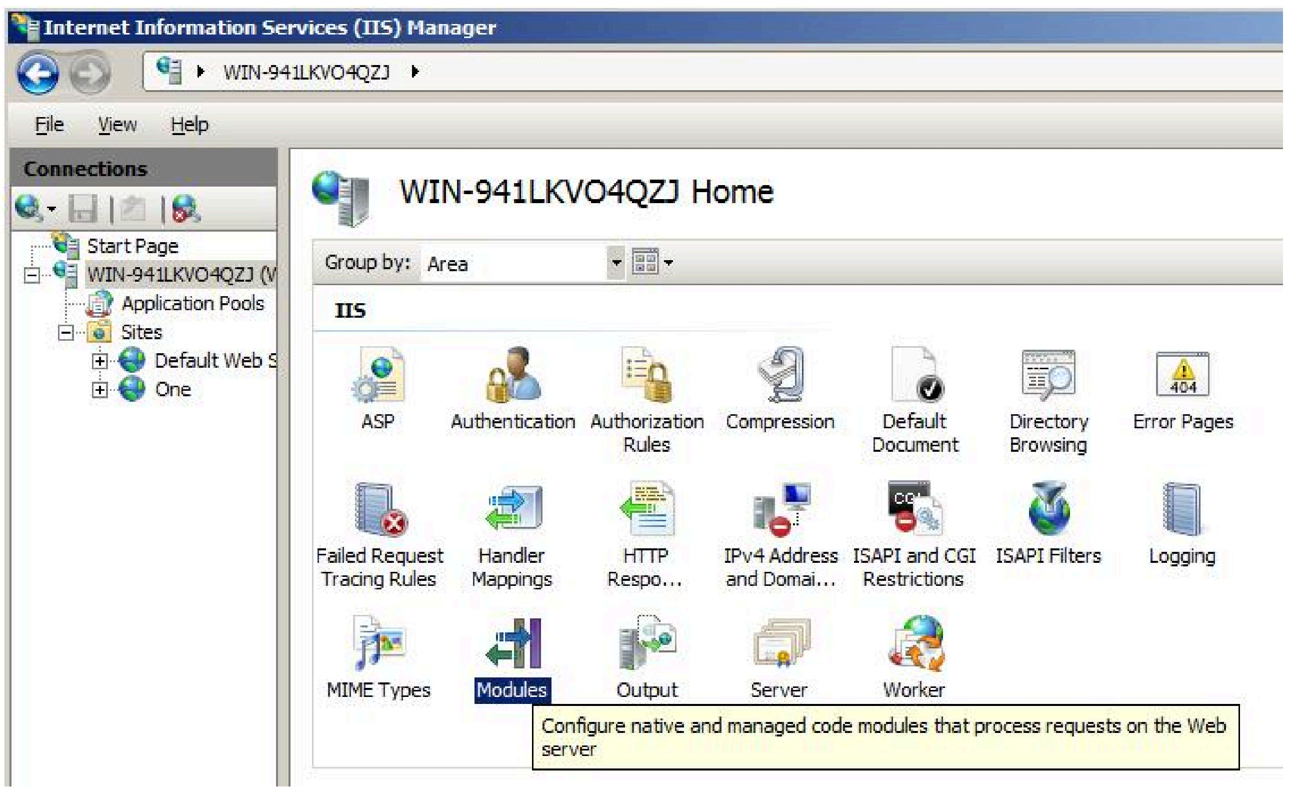


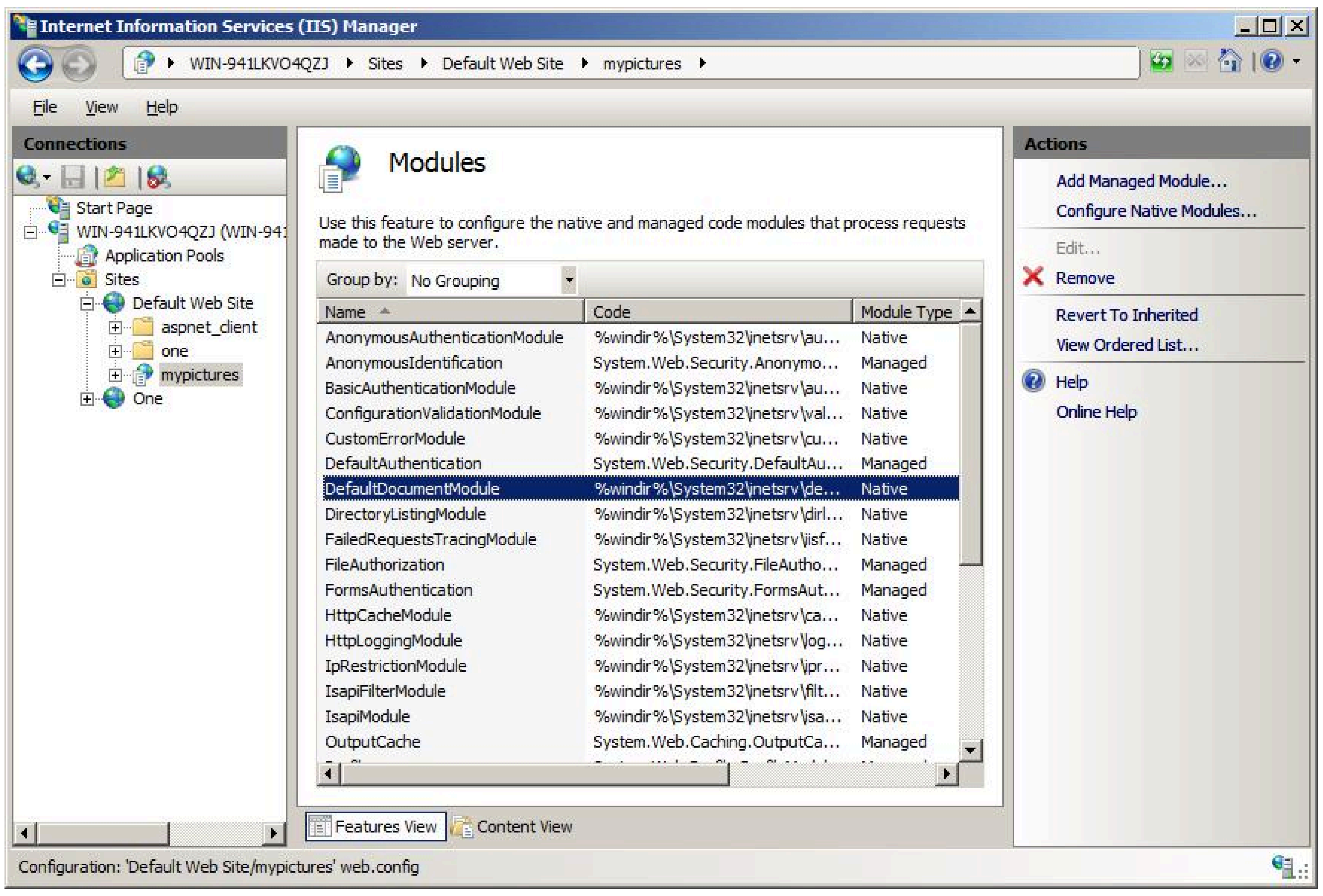
Figure 1: IIS Manager

To manage the enabled modules for a particular application

1. Connect to that application using the tree view on the left.
2. Navigate to the Modules feature pane.
3. Click **Add Managed Module** to add a Managed Module to your application. **Add Native Module** only allows you to add a module that is already registered on the Server level.

To remove a module from your application

1. Click the module in the list. The **Remove** task displays in the left pane.
2. Click **Remove**.



Modules

Use this feature to configure the native and managed code modules that process requests made to the Web server.

Group by: No Grouping

Name	Code	Module Type
AnonymousAuthenticationModule	%windir%\System32\inetsrv\au...	Native
AnonymousIdentification	System.Web.Security.Anonymo...	Managed
BasicAuthenticationModule	%windir%\System32\inetsrv\au...	Native
ConfigurationValidationModule	%windir%\System32\inetsrv\val...	Native
CustomErrorModule	%windir%\System32\inetsrv\cu...	Native
DefaultAuthentication	System.Web.Security.DefaultAu...	Managed
DefaultDocumentModule	%windir%\System32\inetsrv\de...	Native
DirectoryListingModule	%windir%\System32\inetsrv\dirl...	Native
FailedRequestsTracingModule	%windir%\System32\inetsrv\iisf...	Native
FileAuthorization	System.Web.Security.FileAutho...	Managed
FormsAuthentication	System.Web.Security.FormsAut...	Managed
HttpCacheModule	%windir%\System32\inetsrv\ca...	Native
HttpLoggingModule	%windir%\System32\inetsrv\log...	Native
IpRestrictionModule	%windir%\System32\inetsrv\ipr...	Native
IsapiFilterModule	%windir%\System32\inetsrv\filt...	Native
IsapiModule	%windir%\System32\inetsrv\isa...	Native
OutputCache	System.Web.Caching.OutputCa...	Managed

Configuration: 'Default Web Site/mypictures' web.config

Figure 2: Modules List in IIS Manager

Managing Modules from the Command Line

To quickly manage modules from the command line, or script module management tasks, use the **AppCmd.exe** command line tool.

To install a module using AppCmd.exe

At the Command Prompt, type the following command, replacing values in italics with the values for the module on your computer.

Console

```
Appcmd.exe install module /name:MODULE_NAME /image:PATH_TO_DLL
```

Note

Installing a native module automatically enables it at the server level.

For example, the following command installs the Default Document module.

Console

```
%windir%\system32\inetsrv\appcmd.exe install module /name:DefaultDocumentModule /image:%wi
```

If you run the command on a computer where the module is already loaded, you wil get an error message similar to the following:

ERROR (message:Failed to add duplicate collection element "DefaultDocumentModule".)

To uninstall a module using AppCmd.exe

At the Command Prompt, type the following command, replacing values in italics with the values for the module on your computer.

Console

```
Appcmd.exe uninstall module MODULE_NAME
```

Note

Uninstalling a module automatically disables it at the server level.

For example, the following command uninstalls the Default Document module. This command is provided only as example. You should only run it on a test server.

Console

```
%windir%\system32\inetsrv\appcmd.exe uninstall module DefaultDocumentModule
```

To list the modules enabled either for an application or globally

At the Command Prompt, type the following command, replacing values in italic text with the values for the module on your computer.

Console

```
Appcmd.exe list modules [/app.name:APPLICATION_NAME]
```

For example, the following command lists the modules enabled for the Default Web Site.

Console

```
%windir%\system32\inetsrv\appcmd.exe list modules /app.name:"Default Web Site"
```

And this command lists the modules installed globally.

Console

```
%windir%\system32\inetsrv\appcmd.exe list modules
```

To enable a module either for a particular application or globally

At the Command Prompt, type the following command, replacing values in italic text with the values for the module on your computer.

Console Copy

```console
Appcmd.exe add module /name:MODULE_NAME /type:MGD_TYPE
```

For example, the following command adds the Forms Authentication module to the Default Web Site.

Console Copy

```console
%windir%\system32\inetsrv\appcmd.exe add module /name:FormsAuthentication /type:System.Web
```

And this command adds the Forms Authentication Module globally.

Console Copy

```console
%windir%\system32\inetsrv\appcmd.exe add module /name:FormsAuthentication /type:System.Web
```

To disable a module either for a particular application or globally

At the Command Prompt, type the following command, replacing values in italic text with the values for the module on your computer.

Console Copy

```console
Appcmd.exe delete module MODULE_NAME [/app.name:APPLICATION_NAME]
```

For example, the following command removes the Forms Authentication module from the Default Web Site.

Console Copy

```console
%windir%\system32\inetsrv\appcmd.exe delete module FormsAuthentication /app.name:"Default
```

And this command removes the Forms Authentication Module globally.

Console Copy

```console
%windir%\system32\inetsrv\appcmd.exe delete module FormsAuthentication
```

To get more help on the syntax for each of the App Cmd.exe commands

To display the commands supported on the Module object type the following command:

Console Copy

```console
Appcmd.exe module /?
```

To display the usage of each command, type the following command:

Console Copy

```console
Appcmd.exe install module /?Appcmd add module /?
```

Module Reference

The IIS server is ready for customizing. Read the following list carefully to avoid side-effects, feature loss or removing security features.

Utility Modules

These modules do not provide request services, but instead assist the server engine with its internal operation.

Expand table

Module Name:	**UriCacheModule**
Description:	Implements a generic cache for URL-specific server state, such as configuration. With this module, the server only reads configuration for the first request for a particular URL, and reuse it on subsequent requests until it changes.
Configuration sections:	None.
Dependencies:	None.
Potential issues when removing this module	Performance loss due to state cached for each URL retrieval for every request.
Module Name:	**FileCacheModule**
Description:	Caches file handles for files opened by the server engine and modules.
Configuration sections:	None.
Dependencies:	None.
Potential issues when removing this module	Performance loss. If file handles are not cached, the files have to be opened for every request.
Module Name:	**TokenCacheModule**
Description:	Caches windows security tokens for password based authentication schemes (anonymous authentication, basic authentication, IIS client certificate authentication).
Configuration sections:	None.
Dependencies:	None.
Potential issues when removing this module	Performance loss. The users must be logged on for every request if the token is not cached. A major performance impact may result. For example, if a password protected html page references 50 images that are also protected, 51 logonUser calls to the local account database, or worse, to an off-box domain controller, result in a performance issue.
Module Name:	**ManagedEngine**
Description:	Managed Engine has a special place within all the other modules. It is responsible for providing the IIS integration to hook up with the ASP.NET runtime.
Configuration sections:	
Dependencies:	None.
Potential issues when removing this module	ASP.NET integration will be disabled. None of the managed modules declared in the `<modules>` or ASP.NET handlers declared in the `<handlers>` section are called when the Application Pool runs in Integrated Mode.
Module Name:	**HttpCacheModule**
Description:	The HttpCacheModule implements the IIS output cache and also the logic for caching items in the http.sys cache. Set the cache size, output cache profiles etc. via configuration.
Configuration sections:	System.webServer/caching
Dependencies:	None.
Potential issues when removing this module	Content will no longer be cached in kernel mode. Cache profiles are ignored. Removing the HttpCacheModule will probably have adverse effects on performance and resource usage.
Module Name:	**DynamicCompressionModule**
Description:	Implements in-memory compression of dynamic content.

Configuration sections:	system.webServer/httpCompression and system.webServer/urlCompression.
Dependencies:	There will not be any dependencies because Dynamic compression is turned off by default.
Module Name:	**StaticCompressionModule**
Description:	Implements compression (in memory as well as persistent in the file system) of static content.
Configuration sections:	system.webServer/httpCompression and system.webServer/urlCompression
Dependencies:	None.
Potential issues when removing this module	Potential Bandwidth saturation due to uncompressed content being sent back to the client.
Module Name:	**DefaultDocumentModule**
Description:	Implements default document functionality. Requests that come in with a trailing / will be rerouted to a document in the default document list.
Configuration sections:	system.webServer/defaultDocument
Dependencies:	None.
Potential issues when removing this module	Requests to /, for example `http://localhost/`, return a 404 error. If a directoryBrowsing is enabled, a directory listing is generated.
Module Name:	**DirectoryListingModule**
Description:	Implements directory browsing functionality.
Configuration sections:	system.webServer/directoryBrowse
Dependencies:	None.
Potential issues when removing this module	If the neither the default document module nor the directoryListing Module handle a request for a /, an empty response returns.
Module Name:	**ProtocolSupportModule**
Description:	Implements custom and redirect response headers. Implements the trace and Options HTTP verbs. Implements the supports which allow or turn off keep-alive support via configuration.
Configuration sections:	system.webServer/httpProtocol
Dependencies:	None.
Potential issues when removing this module	TRACE or OPTIONS requests return a "405 Method not allowed" error message.
Module Name:	**HttpRedirectionModule**
Description:	Implements redirect functionality.
Configuration sections:	system.webServer/httpRedirect
Dependencies:	None.
Potential issues when removing this module	Potential security issue if resources were protected by redirection. When the Redirection module is removed, the content becomes accessible again.
Module Name:	**ServerSideIncludeModule**
Description:	Implements server side includes. This module is mapped as a handler, only executing for requests ending in .stm, .shtm and .shtml.
Configuration sections:	system.webServer/serverSideInclude
Dependencies:	None.

Potential issues when removing this module	The static file module handles.stm, .shtm and .shtml files. If this module has a mimeMap for these extensions the files become served as text. Note, however, that this is not the default.
Module Name:	**StaticFileModule**
Description:	Sends out static files with the file extension .html, .jpg, as well as many others. The staticContent/mimeMap configuration collection determines the list of file extensions.
Configuration sections:	system.webServer/staticContent
Dependencies:	None.
Potential issues when removing this module	Static files no longer become served. Requests for files return a 404 Not Found error indicating that no handler was matched.
Module Name:	**AnonymousAuthenticationModule**
Description:	Implements anonymous authentication. This module generates the HttpUser object if a URL is configured to allow anonymous authentication.
Configuration sections:	system.webServer/security/authentication/anonymousAuthentication
Dependencies:	None.
Potential issues when removing this module	At least one authentication module must be configured. The IIS server core checks after the authentication phase if the HttpUser object is populated. The HttpUser object is an IIS data structure. A 401.2 error generates if there is no authentication populating the HttpUser object.
Module Name:	**CertificateMappingAuthenticationModule**
Description:	Maps SSL client certificates to an Active Directory account (Active Directory Certificate Mapping).
Configuration sections:	system.webServer/security/authentication/clientCertificateMappingAuthentication
Dependencies:	SSL must be configured for this module to work. The IIS machine must also be a member of an Active Directory domain.
Potential issues when removing this module	Requests are normally allowed if Active Directory Certificate Mapping is used to protect a directory; in this case, the module is removed.
Module Name:	**BasicAuthenticationModule**
Description:	implements HTTP Basic authentication described in RFC 2617.
Configuration sections:	system.webServer/security/authentication/basicAuthentication
Dependencies:	None.
Potential issues when removing this module	At least one authentication module has must be configured. The IIS server core checks after the authentication phase if the HttpUser object is populated. The HttpUser object is an IIS data structure. A 401.2 error generates if there is no authentication populating the HttpUser object.
Module Name:	**WindowsAuthenticationModule**
Description:	Implements Windows authentication (NTLM or Negotiate (Kerberos)).
Configuration sections:	system.webServer/security/authentication/windowsAuthentication
Dependencies:	None.
Potential issues when removing this module	At least one authentication module must be configured. The IIS server core checks after the authentication phase if the HttpUser object is populated. The HttpUser object is an IIS data structure. A 401.2 error is generates if there is no authentication populating the HttpUser object.
Module Name:	**DigestAuthenticationModule**
Description:	Implements digest authentication described in RFC 2617.
Configuration sections:	system.webServer/security/authentication/digestAuthentication
Dependencies:	IIS server must be part of an Active Directory domain.
Potential issues when removing this module	At least one authentication module must be configured. The IIS server core checks after the authentication phase if the HttpUser object is populated. The HttpUser object is an IIS data structure. A 401.2 error is generates if there is no authentication populating the HttpUser object.

Module Name:	**IISCertificateMappingAuthenticationModule**
Description:	Implements IIS certificate mapping. Maps SSL client certificates to a Windows account. Contrary to Active Directory Certificate mapping, user credentials and mapping rules are stored within the IIS configuration store
Configuration sections:	system.webServer/iisClientCertificateMappingAuthentication
Dependencies:	SSL with the requirement to receive client certificates has to be configured for this module to work.
Potential issues when removing this module	At least one authentication module must be configured. The IIS server core checks after the authentication phase if the HttpUser object is populated. The HttpUser object is an IIS data structure. A 401.2 error generates if there is no authentication populating the HttpUser object.
Module Name:	**UrlAuthorizationModule**
Description:	Implements authorization based on configuration rules.
Configuration sections:	system.webServer/security/authorization
Dependencies:	None.
Potential issues when removing this module	Authorization rules which protected content are no longer evaluated. Content that was supposed to be protected might be served.
Module Name:	**IsapiModule**
Description:	Implements ISAPI Extension functionality.
Configuration sections:	system.webServer/isapiCgiRestriction
Dependencies:	None.
Potential issues when removing this module	ISAPI Extensions mapped in the `<handlers>` section (modules="IsapiModule") or explicitly called ISAPI Extensions will no longer work.
Module Name:	**IsapiFilterModule**
Description:	Implements ISAPI filter functionality.
Configuration sections:	system.webServer/isapiFilters
Dependencies:	None.
Potential issues when removing this module	ISAPI filters often implement functionality applications rely on. Examples are ASP.NET or SharePoint. ASP.NET for example needs the aspnet_filter.dll to protect sensitive content and to rewrite URLs. Removing this module prevents IIS from loading ISAPI filters. Applications might stop working or sensitive content may be exposed.
Module Name:	**IpRestrictionModule**
Description:	Implements an authorization scheme based on the IPv4 address of the client request.
Configuration sections:	system.webServer/security/ipSecurity
Dependencies:	IPv4 stack must be installed.
Potential issues when removing this module	Clients with IP addresses on the ipSecurity list will be allowed.
Module Name:	**RequestFilteringModule**
Description:	Implements a powerful set of security rules which reject suspicious request at a very early stage. This module is the successor to the ISAPI filter UrlScan.DLL that was shipped for IIS 5.0 and 6.0.
Configuration sections:	system.webServer/security/requestFiltering
Dependencies:	None.
Potential issues when removing this module	If this module is removed, the rules specified in the requestFiltering section no longer apply. Potential security issues may result.
Module Name:	**CustomLoggingModule**

Description:	Implements the ILogPlugin interface on top of IIS. ILogPlugin is a previous COM implementation that allowed customers to extend IIS logging. We do not recommend extending IIS using this interface. Instead, customers should write a module and subscribe to the RQ_LOG_REQUEST notification.
Configuration sections:	system.webServer/httpLogging and system.applicationhost/sites/site/logFile/customLogPluginClsid
Dependencies:	None.
Potential issues when removing this module	A custom log plug-in will no longer be called. For example, ODBC Logging is implemented as ILogPlugin.
Module Name:	**CustomErrorModule**
Description:	Implements custom errors and the IIS detailed error feature.
Configuration sections:	system.webServer/httpErrors
Dependencies:	None.
Potential issues when removing this module	IIS returns blank pages with minimal information when errors within the core server occur. Remote users may see detailed error information coming from server components which can result in disclosing information.
Module Name:	**HttpLoggingModule**
Description:	Implements standard IIS logging by telling HTTP.SYS what to log.
Configuration sections:	system.applicationHost/log and system.webServer/httpLogging
Dependencies:	None.
Potential issues when removing this module	Standard IIS logging will no longer work.
Module Name:	**FailedRequestsTracingModule**
Description:	Implements tracing of failed requests. Define and set rules for failed requests via configuration.
Configuration sections:	system.webServer/tracing and system.webServer/httpTracing
Dependencies:	None.
Potential issues when removing this module	Tracing http requests will no longer work.
Module Name:	**RequestMonitorModule**
Description:	Implements the IIS Run-time State and Control Interface (RSCA). RSCA allows users to query for run-time information such as currently executing request, start/stop state of a web-site, or currently executing application domains.
Configuration sections:	None.
Dependencies:	None.
Potential issues when removing this module	Tools will be unable to enumerate currently executing requests.
Module Name:	**CgiModule**
Description:	Implements CGI on top of IIS.
Configuration sections:	system.webServer/cgi and system.webServer/isapiCgiRestriction
Dependencies:	None.
Potential issues when removing this module	CGI programs will stop working.
Module Name:	**TracingModule**
Description:	Implements ETW tracing.

Configuration sections:	system.webServer/httpTracing
Dependencies:	None.
Potential issues when removing this module	ETW tracing will not work if this module is removed.
Module Name:	**ConfigurationValidationModule**
Description:	Validates that ASP.NET application configuration has been migrated to work in Integrated mode.
Configuration sections:	system.webServer/Validation
Dependencies:	None.
Potential issues when removing this module	Applications that specify legacy ASP.NET configuration for modules and handlers will not generate migration errors; a new application that has not been migrated will function incorrectly.

Managed Modules:

Expand table

Module Name:	**OutputCache**
Description:	Implements the ASP.NET Output Caching feature.
Configuration sections:	system.web/caching/outputCache
Dependencies:	ManagedEngine module has to be installed.
Potential issues when removing this module	ASP.NET will be unable to output cache responses to pages configured to be output cached.
Module Name:	**Session**
Description:	See ASP.NET 2.0 documentation for details
Configuration sections:	system.web/sessionState
Dependencies:	ManagedEngine module must be installed
Potential issues when removing this module	Managed session state is not available.
Module Name:	**WindowsAuthentication**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	system.web/authentication
Dependencies:	ManagedEngine module has to be installed
Potential issues when removing this module	WindowsAuthentication.OnAuthenticate event will not be raised, which may prevent some custom ASP.NET authentication code from running. Also, the authenticated user will not be replaced with the UNC user when on UNC share (legacy ASP.NET behavior). This module does not affect NTLM/Kerberos authentication for ASP.NET applications in Integrated mode, and is not required outside of the WindowsAuthentication.OnAuthenticate event and the legacy UNC behavior.
Module Name:	**FormsAuthentication**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	system.web/authentication
Dependencies:	ManagedEngine module has to be installed
Potential issues when removing this module	ASP.NET forms-based authentication feature will not be available, resulting in clients with forms authentication tickets not being able to access protected resources.

Module Name:	**DefaultAuthentication**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	system.web/authentication
Dependencies:	ManagedEngine module has to be installed
Potential issues when removing this module	Some ASP.NET features may not work for anonymous requests if ASP.NET authentication mode is Forms. Also, DefaultAuthentication.OnAuthenticate event will not be raised.
Module Name:	**RoleManager**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	None.
Dependencies:	ManagedEngine module must be installed.
Potential issues when removing this module	Role Manager functionality not available.
Module Name:	**UrlAuthorization**
Description:	See ASP.NET 2.0 documentation for details. The native UrlAuthorization module implements URL authorization functionality in native code. This provides a scalable and fast native alternative for the managed URL authorization module.
Configuration sections:	system.web/authorization.
Dependencies:	ManagedEngine module must be installed.
Potential issues when removing this module	ASP.NET authorization rules will be ignored, possibly resulting in information disclosure and other security compromises.
Module Name:	**AnonymousIdentification**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	
Dependencies:	ManagedEngine module has to be installed.
Potential issues when removing this module	The anonymous identification feature used by the ASP.NET Profile will not work.
Module Name:	**Profile**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	
Dependencies:	ManagedEngine module must be installed.
Potential issues when removing this module	The ASP.NET Profile feature will not work.
Module Name:	**UrlMappingsModule**
Description:	See ASP.NET 2.0 documentation for details.
Configuration sections:	
Dependencies:	ManagedEngine module must be installed.
Potential issues when removing this module	The ASP.NET URL mappings will not work.

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

Additional resources

Training

Learning path

Implement finance and operations apps - Training

Plan and design your project methodology to successfully implement finance and operations apps with FastTrack services, data management and more.