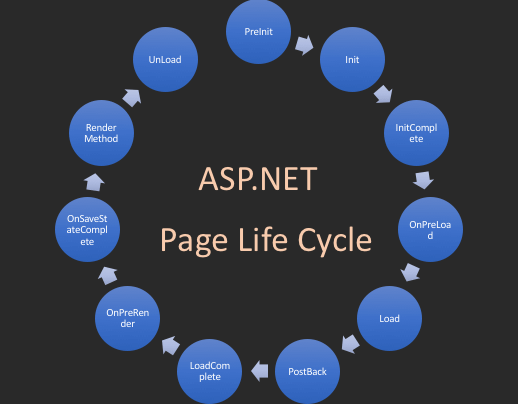
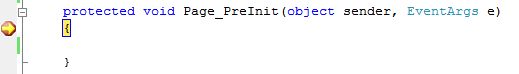
ASP.NET LIFECYCLE



| **Page Event** | **Typical Use** |
| --- | --- |
| [PreInit](https://msdn.microsoft.com/en-us/library/7x9azc75(v=vs.100)) | Raised after the start stage is complete and before the initialization stage begins.  Use this event for the following:   * Check the [IsPostBack](https://msdn.microsoft.com/en-us/library/5zabsw0t(v=vs.100)) property to determine whether this is the first time the page is being processed. The [IsCallback](https://msdn.microsoft.com/en-us/library/4kacsys6(v=vs.100)) and [IsCrossPagePostBack](https://msdn.microsoft.com/en-us/library/1c23y6wz(v=vs.100)) properties have also been set at this time. * Create or re-create dynamic controls. * Set a master page dynamically. * Set the [Theme](https://msdn.microsoft.com/en-us/library/kd4hkt0b(v=vs.100)) property dynamically. * Read or set profile property values.   **Note**  If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event. |
| [Init](https://msdn.microsoft.com/en-us/library/ddx12zse(v=vs.100)) | Raised after all controls have been initialized and any skin settings have been applied. The [Init](https://msdn.microsoft.com/en-us/library/ddx12zse(v=vs.100)) event of individual controls occurs before the [Init](https://msdn.microsoft.com/en-us/library/ddx12zse(v=vs.100)) event of the page.  Use this event to read or initialize control properties. |
| [InitComplete](https://msdn.microsoft.com/en-us/library/3wf2z4k4(v=vs.100)) | Raised at the end of the page's initialization stage. Only one operation takes place between the [Init](https://msdn.microsoft.com/en-us/library/ddx12zse(v=vs.100)) and [InitComplete](https://msdn.microsoft.com/en-us/library/3wf2z4k4(v=vs.100)) events: tracking of view state changes is turned on. View state tracking enables controls to persist any values that are programmatically added to the [ViewState](https://msdn.microsoft.com/en-us/library/hf83c91z(v=vs.100)) collection. Until view state tracking is turned on, any values added to view state are lost across postbacks. Controls typically turn on view state tracking immediately after they raise their [Init](https://msdn.microsoft.com/en-us/library/ddx12zse(v=vs.100)) event.  Use this event to make changes to view state that you want to make sure are persisted after the next postback. |
| [PreLoad](https://msdn.microsoft.com/en-us/library/3s4f1wwa(v=vs.100)) | Raised after the page loads **view state** for itself and all controls, and after it processes **postback data** that is included with the [Request](https://msdn.microsoft.com/en-us/library/xc67sd5e(v=vs.100)) instance. |
| [Load](https://msdn.microsoft.com/en-us/library/abk3yt37(v=vs.100)) | The [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object calls the [OnLoad](https://msdn.microsoft.com/en-us/library/ds9cf97s(v=vs.100)) method on the [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object, and then recursively does the same for each child control until the page and all controls are loaded. The [Load](https://msdn.microsoft.com/en-us/library/abk3yt37(v=vs.100)) event of individual controls occurs after the [Load](https://msdn.microsoft.com/en-us/library/abk3yt37(v=vs.100)) event of the page.  Use the [OnLoad](https://msdn.microsoft.com/en-us/library/ds9cf97s(v=vs.100)) event method to set properties in controls and to establish database connections. |
| Control events | Use these events to handle specific control events, such as a [Button](https://msdn.microsoft.com/en-us/library/3e83tsk6(v=vs.100)) control's [Click](https://msdn.microsoft.com/en-us/library/d02sce24(v=vs.100)) event or a [TextBox](https://msdn.microsoft.com/en-us/library/4b1xz97b(v=vs.100)) control's [TextChanged](https://msdn.microsoft.com/en-us/library/81a1axdx(v=vs.100)) event.  **Note**  In a postback request, if the page contains validator controls, check the **[IsValid](https://msdn.microsoft.com/en-us/library/9s0wde39(v=vs.100))** property of the [**Page**](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) and of individual validation controls before performing any processing. |
| [LoadComplete](https://msdn.microsoft.com/en-us/library/bs9d1zcy(v=vs.100)) | Raised at the end of the event-handling stage.  Use this event for tasks that require that all other controls on the page be loaded. |
| [PreRender](https://msdn.microsoft.com/en-us/library/768a7bzf(v=vs.100)) | Raised after the [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object has created all controls that are required in order to render the page, including child controls of composite controls. (To do this, the [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object calls [EnsureChildControls](https://msdn.microsoft.com/en-us/library/19ckw0e5(v=vs.100)) for each control and for the page.)  The [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object raises the [PreRender](https://msdn.microsoft.com/en-us/library/768a7bzf(v=vs.100)) event on the [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object, and then recursively does the same for each child control. The [PreRender](https://msdn.microsoft.com/en-us/library/768a7bzf(v=vs.100)) event of individual controls occurs after the [PreRender](https://msdn.microsoft.com/en-us/library/768a7bzf(v=vs.100)) event of the page.  Use the event to make final changes to the contents of the page or its controls before the rendering stage begins. |
| [PreRenderComplete](https://msdn.microsoft.com/en-us/library/d299wwfd(v=vs.100)) | Raised after each data bound control whose [DataSourceID](https://msdn.microsoft.com/en-us/library/hw694zdw(v=vs.100)) property is set calls its [DataBind](https://msdn.microsoft.com/en-us/library/hyyes28e(v=vs.100)) method. For more information, see Data Binding Events for Data-Bound Controls later in this topic. |
| [SaveStateComplete](https://msdn.microsoft.com/en-us/library/9tc8ys0a(v=vs.100)) | Raised after view state and control state have been saved for the page and for all controls. Any changes to the page or controls at this point affect rendering, but the changes will not be retrieved on the next postback. |
| [Render](https://msdn.microsoft.com/en-us/library/26zakx94(v=vs.100)) | This is not an event; instead, at this stage of processing, the [Page](https://msdn.microsoft.com/en-us/library/dfbt9et1(v=vs.100)) object calls this method on each control. All ASP.NET Web server controls have a [Render](https://msdn.microsoft.com/en-us/library/26zakx94(v=vs.100)) method that writes out the control's markup to send to the browser.  If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the [Render](https://msdn.microsoft.com/en-us/library/26zakx94(v=vs.100)) method. For more information, see [Developing Custom ASP.NET Server Controls](https://docs.microsoft.com/en-us/previous-versions/aspnet/zt27tfhy(v=vs.100)).  A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code. |
| [Unload](https://msdn.microsoft.com/en-us/library/fxf48ycz(v=vs.100)) | Raised for each control and then for the page.  In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.  For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks. |

PreInit

1. Check the IsPostBack property to determine whether this is the first time the page is being processed.
2. Create or re-create dynamic controls.
3. Set a master page dynamically.
4. Set the Theme property dynamically.

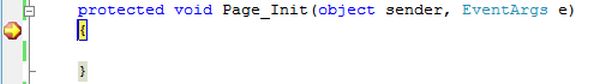


**Note**

If the request is a postback then the values of the controls have not yet been restored from the view state. If you set a control property at this stage, its value might be overwritten in the next event.

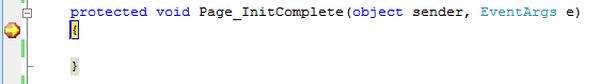
Init

1. This event fires after each control has been initialized.
2. Each control's UniqueID is set and any skin settings have been applied.
3. Use this event to read or initialize control properties.
4. The "Init" event is fired first for the bottom-most control in the hierarchy, and then fired up the hierarchy until it is fired for the page itself.



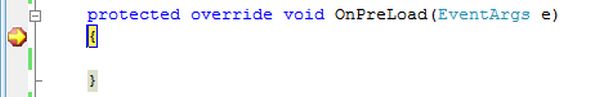
InitComplete

1. Until now the viewstate values are not yet loaded, hence you can use this event to make changes to the view state that you want to ensure are persisted after the next postback.
2. Raised by the Page object.
3. Use this event for processing tasks that require all initialization to be complete.



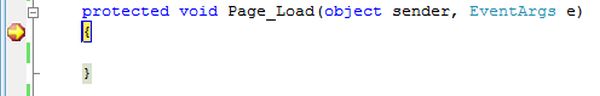
OnPreLoad

1. Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.
2. Before the Page instance raises this event, it loads view state for itself and all controls, and then processes any postback data included with the Request instance.
3. Loads ViewState: ViewState data are loaded to controls.
4. Loads Postback data: Postback data are now handed to the page controls.



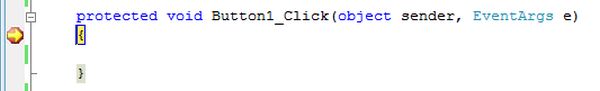
Load

1. The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded. The Load event of individual controls occurs after the Load event of the page.
2. This is the first place in the page lifecycle that all values are restored.
3. Most code checks the value of IsPostBack to avoid unnecessarily resetting state.
4. You may also call Validate and check the value of IsValid in this method.
5. You can also create dynamic controls in this method.
6. Use the OnLoad event method to set properties in controls and establish database connections.



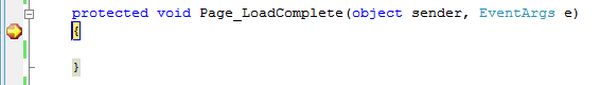
Control PostBack Event(s)

1. ASP.NET now calls any events on the page or its controls that caused the PostBack to occur.
2. Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.
3. In a postback request, if the page contains validator controls, check the IsValid property of the Page and of individual validation controls before performing any processing.
4. This is just an example of a control event. Here it is the button click event that caused the postback.



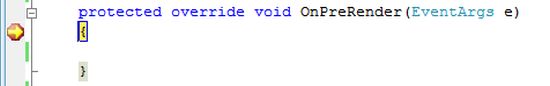
LoadComplete

1. Raised at the end of the event-handling stage.
2. Use this event for tasks that require that all other controls on the page be loaded.



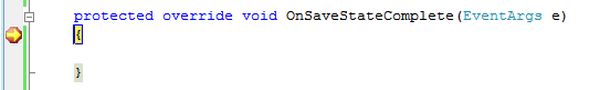
OnPreRender

1. Raised after the Page object has created all controls that are required in order to render the page, including child controls of composite controls.
2. The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control. The PreRender event of individual controls occurs after the PreRender event of the page.
3. The PreRender event of individual controls occurs after the PreRender event of the page.
4. Allows final changes to the page or its control.
5. This event takes place before saving ViewState, so any changes made here are saved.
6. For example: After this event, you cannot change any property of a button or change any viewstate value.
7. Each data bound control whose DataSourceID property is set calls its DataBind method.
8. Use the event to make final changes to the contents of the page or its controls.



OnSaveStateComplete

1. Raised after view state and control state have been saved for the page and for all controls.
2. Before this event occurs, ViewState has been saved for the page and for all controls.
3. Any changes to the page or controls at this point will be ignored.
4. Use this event perform tasks that require the view state to be saved, but that do not make any changes to controls.

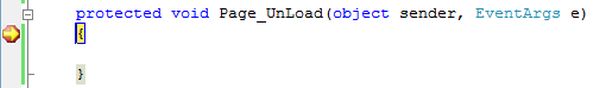


Render Method

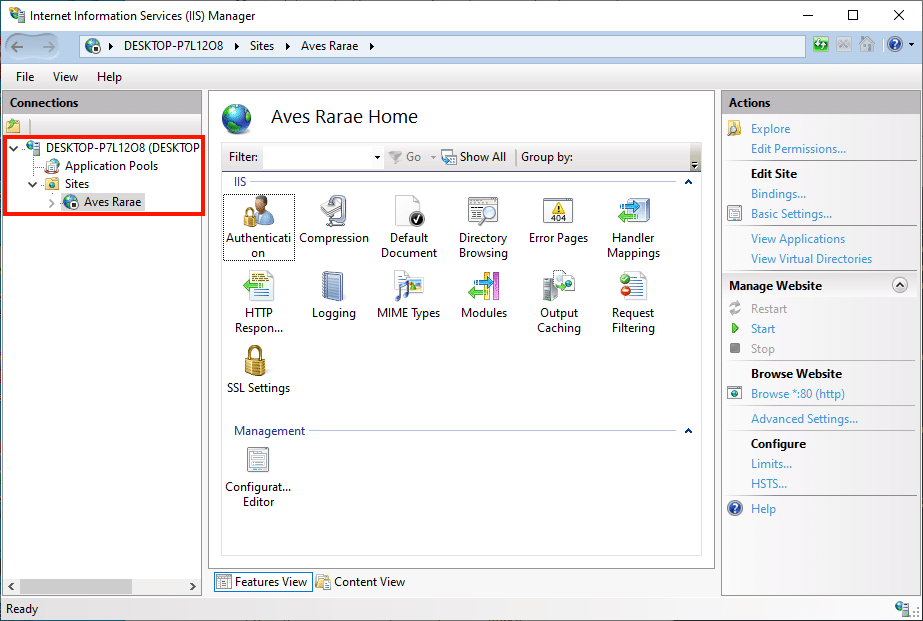
1. This is a method of the page object and its controls (and not an event).
2. The Render method generates the client-side HTML, Dynamic Hypertext Markup Language (DHTML), and script that are necessary to properly display a control at the browser.

UnLoad

1. This event is used for cleanup code.
2. At this point, all processing has occurred and it is safe to dispose of any remaining objects, including the Page object.
3. Cleanup can be performed on:  
   * Instances of classes, in other words objects
   * Closing opened files
   * Closing database connections.
4. This event occurs for each control and then for the page.
5. During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream.
6. If you attempt to call a method such as the Response.Write method then the page will throw an exception.



IIS



# Internet Information Services (IIS)

Internet Information Services (IIS) is a flexible, general-purpose web server from Microsoft that runs on Windows systems to serve requested HTML pages or files.

An IIS web server accepts requests from remote client computers and returns the appropriate response. This basic functionality allows web servers to share and deliver information across local area networks (LAN), such as corporate intranets, and wide area networks (WAN), such as the Internet.

A web server can deliver information to users in several forms, such as static webpages coded in HTML; through file exchanges as downloads and uploads; and text documents, image files and more.

### Web servers provide portals

Modern web servers can provide far more functionality for a business and its users. Web servers are often used as portals for sophisticated, highly interactive, web-based applications that tie enterprise middleware and back-end applications together to create enterprise-class systems. For example, Amazon Web Services allows users to administer public cloud resources through a web-based portal. Meanwhile, streaming media services, such as Spotify for music and Netflix for movies, deliver real-time streaming content through web servers.

### How IIS works

IIS works through a variety of standard languages and protocols. HTML is used to create elements such as text, buttons, image placements, direct interactions/behaviors and hyperlinks. The Hypertext Transfer Protocol (HTTP) is the basic communication protocol used to exchange information between web servers and users. HTTPS -- HTTP over Secure Sockets Layer (SSL) -- uses Transport Layer Security or SSL to encrypt the communication for added data security. The File Transfer Protocol (FTP), or its secure variant, FTPS, can transfer files.

Additional supported protocols include the Simple Mail Transfer Protocol (SMTP), to send and receive email, and the Network News Transfer Protocol, to deliver articles on USENET.

### IIS works with ASP.NET Core

The ASP.NET Core framework is the latest generation of Active Server Page (ASP), a server-side script engine that produces interactive webpages. A request comes in to the IIS server from the web, which sends the request to the ASP.NET Core application, which processes the request and sends its response back to the IIS server and the client who originated the request. Examples of applications written on ASP.NET Core include blog platforms and content management systems (CMS).

Developers can produce IIS websites with a number of tools, including WebDav, which can create and publish web content. Developers can also use integrated development tools, such as Microsoft Visual Studio.

### Versions of IIS

IIS has evolved along with Microsoft Windows. Early versions of IIS arrived with [Windows NT](https://searchwindowsserver.techtarget.com/definition/Windows-NT). IIS 1.0 appeared with Windows NT 3.51, and evolved through IIS 4.0 with Windows NT 4.0. IIS 5.0 shipped with [Windows 2000](https://searchenterprisedesktop.techtarget.com/definition/Windows-2000). Microsoft added IIS 6.0 to Windows Server 2003. IIS 7.0 offered a major redesign with [Windows Server 2008](https://searchwindowsserver.techtarget.com/definition/Microsoft-Windows-Server-2008) (IIS 7.5 is in Windows Server 2008 R2). IIS 8.0 came with [Windows Server 2012](https://searchwindowsserver.techtarget.com/definition/Windows-Server-2012-WS-2012) (Windows Server 2012 R2 uses IIS 8.5). And IIS 10 arrived with [Windows Server 2016](https://searchwindowsserver.techtarget.com/definition/Microsoft-Windows-Server-2016) and [Windows 10](https://searchenterprisedesktop.techtarget.com/definition/Windows-10).

With each iteration of IIS, Microsoft has added new features and updated existing functionality. For example, IIS 3.0 added ASP for dynamic scripting; IIS 6.0 added support for [IPv6](https://searchnetworking.techtarget.com/definition/IPv6-Internet-Protocol-Version-6) and improved security and reliability; and IIS 8.0 brought multicore scaling on [non-uniform memory access](https://whatis.techtarget.com/definition/NUMA-non-uniform-memory-access) hardware, centralized SSL certificate support and Server Name Indication.

### Features in IIS 10

IIS 10 also adds a number of new features and functionality.

IIS 10 adds support for the HTTP/2 protocol, to offer more efficient resource use and lower latency compared to HTTP 1.1. IIS 10 works on the minimal server deployment model [Nano Server](https://searchwindowsserver.techtarget.com/definition/Microsoft-Nano-Server) under Windows Server 2016, and can run ASP.NET Core, Apache [Tomcat](https://www.theserverside.com/definition/Tomcat) and PHP workloads on IIS on the Nano Server.

IIS 10 works in a container and [virtual machine](https://searchservervirtualization.techtarget.com/definition/virtual-machine), so developers and administrators have more flexibility in deployment choices, as well as the density to accommodate a broad range of web applications.

### IIS Express for testing

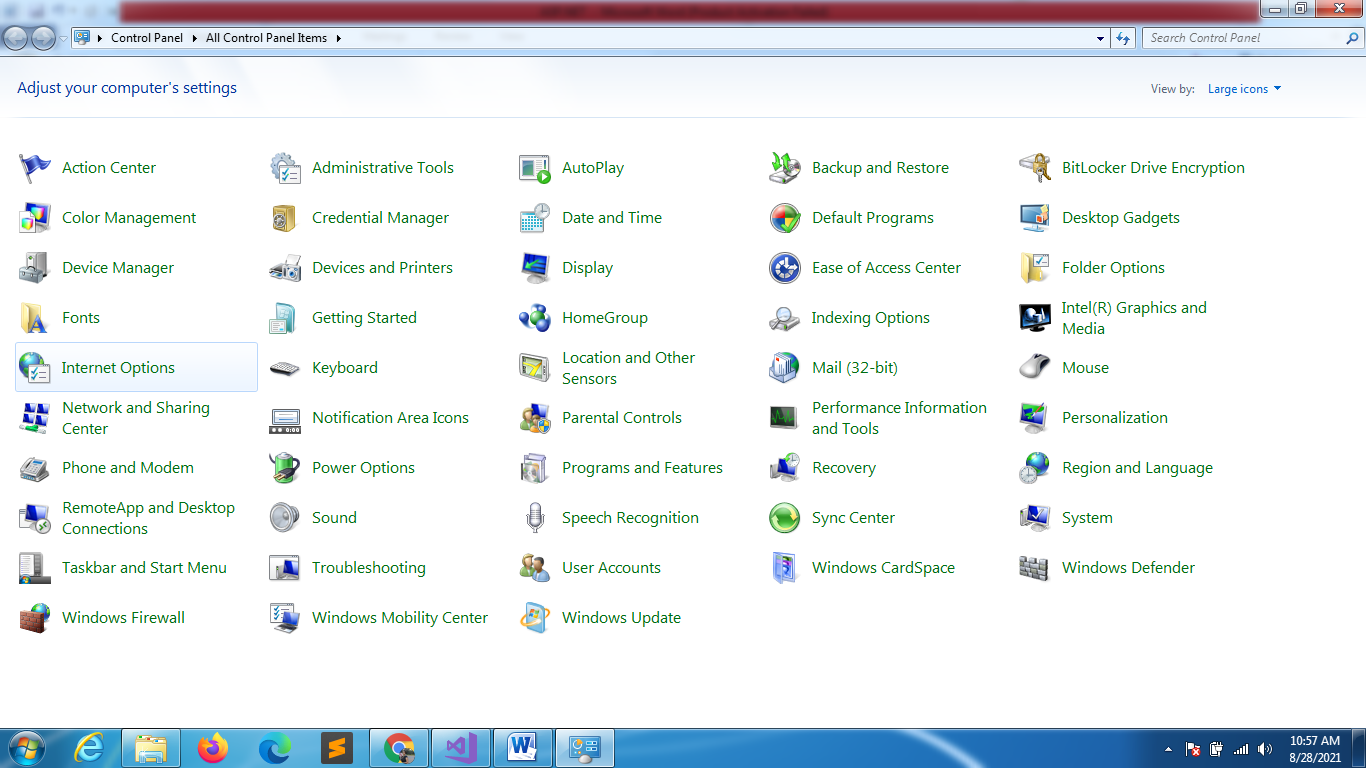
Microsoft provides a self-contained version of IIS, called IIS Express, for developers to test websites. IIS Express offers all the major capabilities of the full IIS web server, but allows many tasks to be performed without administrative privileges.

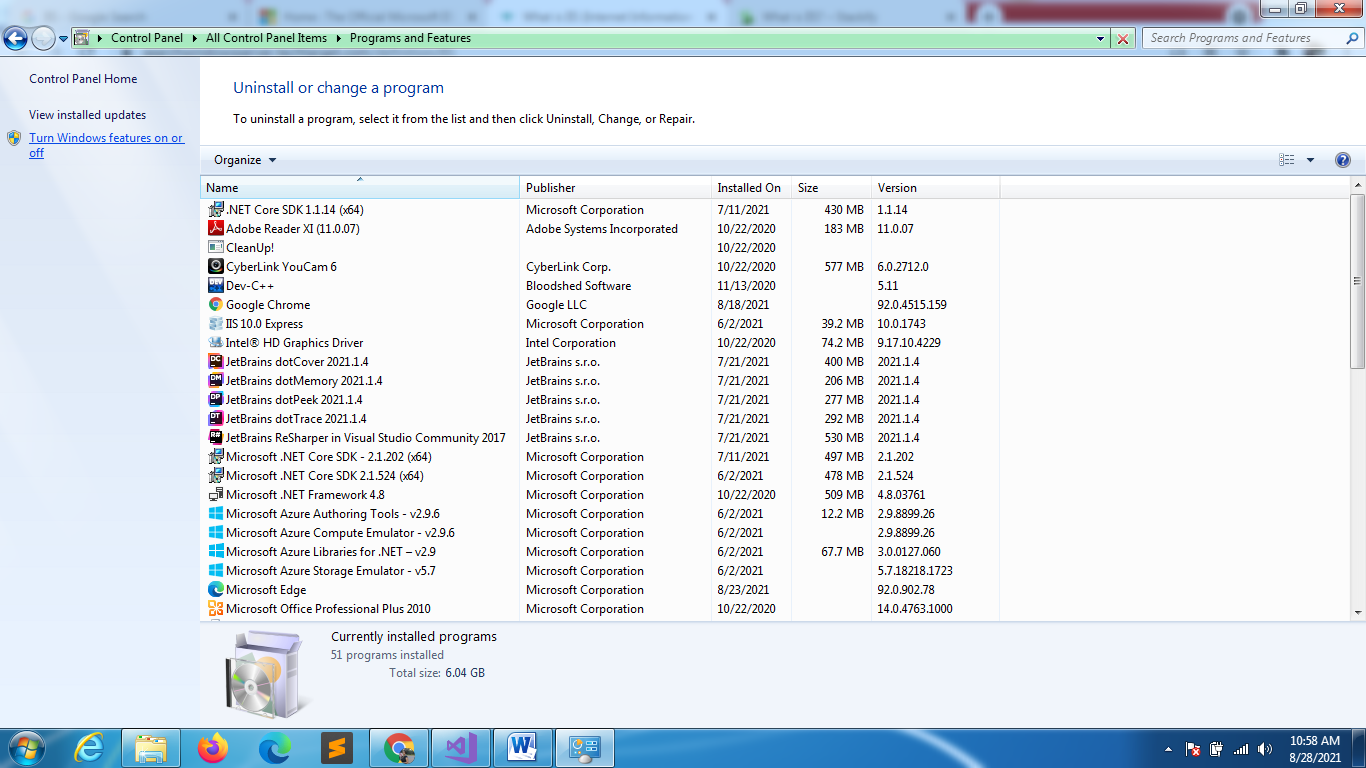
### Security

To ensure a website is secure, organizations need to take security measures to protect the web server from security breaches. Companies can use features built into IIS to harden the IIS.

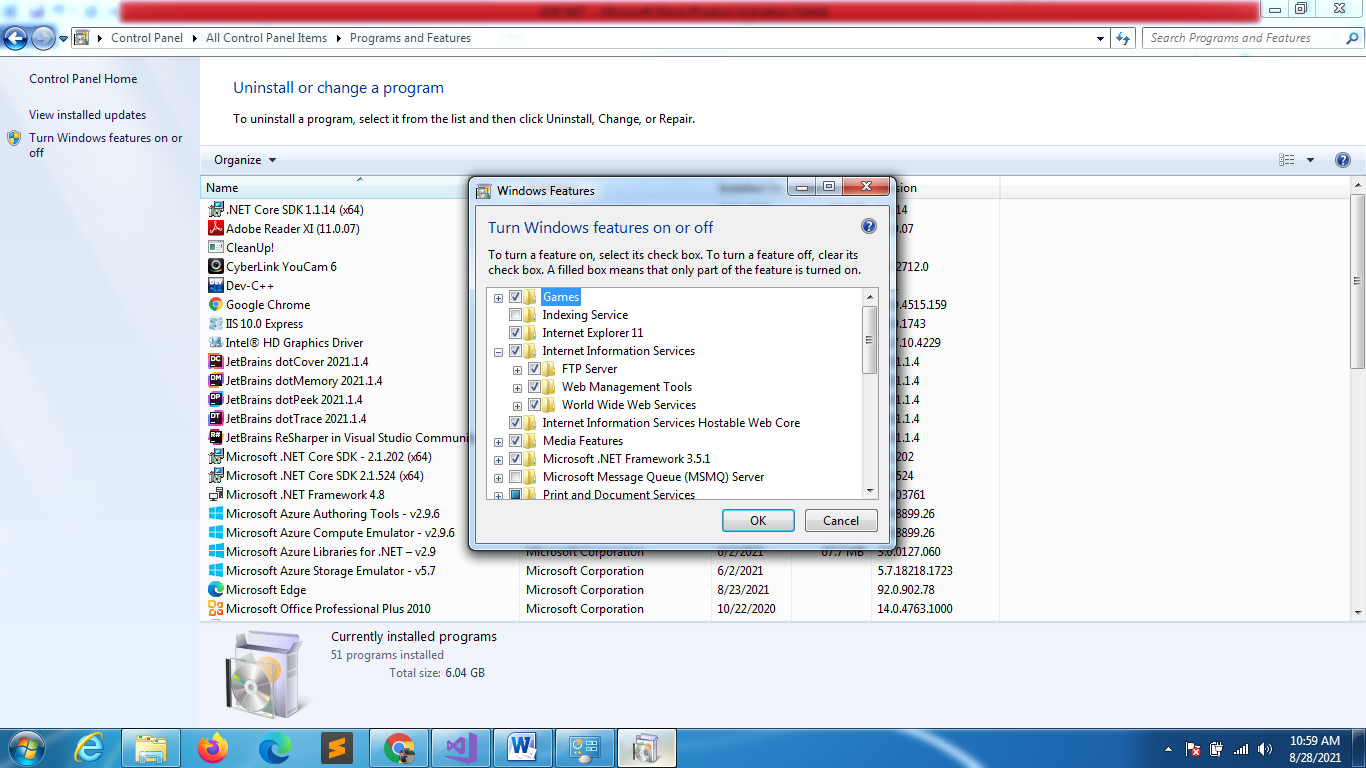
Some of the ways to harden Windows IIS include:

* Ensure the Windows operating system is updated with all security patches.
* Disable any features of IIS not in use to reduce potential attacks.
* Use [firewalls](https://searchsecurity.techtarget.com/definition/firewall) to ensure the server is only receiving valid packets.
* Control which IP addresses and domains can access the web server.
* Use [URL](https://searchnetworking.techtarget.com/definition/URL) authorization to apply rules for specific requests, such as dealing with particular URLs. A company can use URL authorization to only authorize certain users to view the requested pages.
* Use logging to view the visitors accessing the web server.
* Configure the error page to display only relevant information about an issue. Make certain that error pages don’t display too much information, such as usernames, passwords, the IP address of the server or any information that hackers could use to exploit the web server.





CHECK ALL THE CHECKBOXES INSIDE IIS



RUN THE COMMAND

