[Introduction to IIS Architectures | Microsoft Learn](https://learn.microsoft.com/en-us/iis/get-started/introduction-to-iis/introduction-to-iis-architecture" \l "HTTP)

IIS Component

1:N

N:1

App Domain [Website]

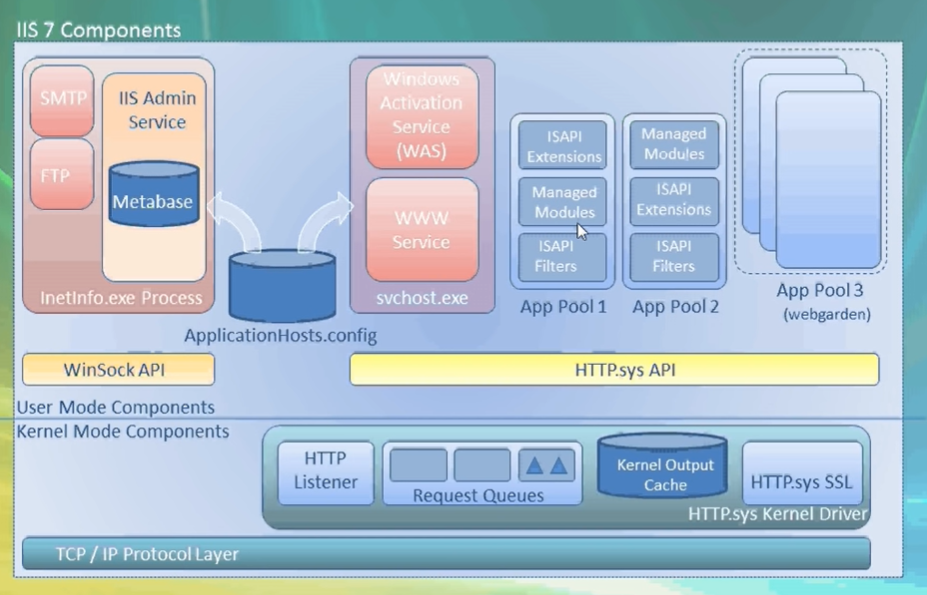
Worker Process

Application Pool

Diagram

Description automatically generated

1. An HTTP request first goes to HTTP.sys and now, HTTP.SYS is responsible for passing the request to a particular application pool.
2. HTTP.sys contacts WAS and WAS requests configuration information from the xml file.
3. The configuration information is sent to WWW service receives.
4. The WWW service uses the configuration information to configure HTTP.sys.
5. Configured HTTP.sys contacts WAS and now, WAS starts a worker process for the application pool to which the request was made.
6. The worker process processes the request and returns a response to HTTP.sys. The request is passed through an ordered series of module in the processing pipeline.



1. Kernel Mode
   1. Http.Sys
2. User Mode
   1. WWW Service
   2. WAS (Windows Activation Service)
   3. ApplicationHost.Config [C:\Windows\System32\inetsrv\config]
   4. App pool
   5. Worker process
   6. ISAPI\_Extensions i.e., Aspnet\_ISAPI.dll
   7. Asp.Net Runtime

Managed Pipeline : Integrated versus Classic

Accounts

1. Local Service
2. Local System
3. Network Service
4. Application Pool Identity
5. IUser

<https://medium.com/@itIsMadhavan/how-do-web-servers-like-iis-works-on-its-background-11c9c0aec701>

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

HTTP Module, HTTP Handler, HTTP Extension, and HTTP Filter are all components in the ASP.NET and IIS request processing pipeline, each serving a specific purpose. Here's a breakdown of their differences:

1. HTTP Module:
   * An HTTP Module is a reusable component that plugs into the ASP.NET request pipeline and participates in processing incoming requests and outgoing responses.
   * Modules are event-driven and can intercept and modify both the request and response at different stages of the request processing pipeline.
   * Modules are registered in the web.config file or at the machine level, and their order of execution can be controlled.
   * Typical uses of HTTP Modules include authentication, logging, URL rewriting, and custom header manipulation.
2. HTTP Handler:
   * An HTTP Handler is responsible for processing incoming requests for a specific resource type (e.g., .aspx, .ashx, .svc) and generating the response.
   * When a request matches the handler's resource type, the handler is invoked to process the request and generate the appropriate response.
   * Handlers are associated with specific file extensions or URL patterns in the web.config file or at the machine level.
   * Common examples of HTTP Handlers include ASP.NET Web Forms (.aspx), ASP.NET AJAX requests (.ashx), and WCF services (.svc).
3. HTTP Extension (IIS Extension):
   * An HTTP Extension is similar to an HTTP Handler in that it processes requests for specific resources based on file extensions.
   * However, HTTP Extensions are implemented outside the ASP.NET runtime and are registered at the IIS level rather than being managed by ASP.NET.
   * They are typically used for non-ASP.NET resources, such as CGI scripts, PHP scripts, or static files that don't require ASP.NET processing.
   * HTTP Extensions can be registered and configured in IIS Manager.
4. HTTP Filter (IIS Filter):
   * An HTTP Filter is a component that intercepts and modifies incoming requests or outgoing responses at the IIS level.
   * Filters operate at a lower level in the request processing pipeline compared to HTTP Modules and Handlers.
   * HTTP Filters can inspect and modify HTTP headers, manipulate the request or response body, or perform other low-level operations.
   * Filters can be useful for scenarios requiring advanced manipulation of incoming or outgoing data at a low level.

In summary, HTTP Modules and HTTP Handlers are specific to ASP.NET and participate in the ASP.NET request pipeline. HTTP Extensions and HTTP Filters operate at the IIS level and can handle non-ASP.NET resources and perform low-level processing. Understanding the differences between these components helps developers and administrators choose the right tool for different stages of request processing in their web applications.

Top of Form