ISAPI Extension Overview

By Archiveddocs

Archived: 2026-04-05 18:48:36 UTC

ISAPI extensions are true applications that run on IIS and have access to all of the functionality provided by IIS. As an example of how powerful ISAPI extensions can be, ASP pages are processed through an ISAPI extension called ASP.dll. In general, clients can access ISAPI extensions the same way they access a static HTML file or dynamic ASP file.

ISAPI extensions are implemented as DLLs that are loaded into a process that is controlled by IIS. Like ASP and HTML pages, IIS uses the virtual location of the DLL file in the file system to map the ISAPI extension into the URL namespace that is served by IIS.

Extensions and filters are the two types of applications that can be developed using ISAPI. An ISAPI extension runs when requested just like any other static HTML file or dynamic ASP file. Since ISAPI applications are compiled code, they are processed much faster than ASP files or files that call COM+ components.

Both ISAPI filters and ISAPI extensions can only be developed using C or C++. Visual Studio includes wizards that speed ISAPI development.

Application Mappings

Application mappings (or script mappings) are the Web server equivalent of file associations in Windows. For example, in Windows, when you open a file that ends in ".txt", the file usually opens in Notepad, because TXT files are mapped to Windows Notepad.exe.

In IIS, ASP functionality is contained in an ISAPI extension called ASP.dll. Any file that is requested from the IIS server that ends in ".asp" is mapped to ASP.dll which is assigned to process the file before displaying its output in the client's window.

A client requests an ISAPI extension in the following way:

```
http://Server_name/ISAPI_name.dll/Parameter
```

To request an ASP file, a client can request a URL like https://Server_name/ASP.dll/File_name.asp because ASP files are processed by the ISAPI extension named %windir%\system32\inetsrv\ASP.dll. However, to simplify ASP requests, IIS uses a script mapping that associates ".asp" file name extensions with ASP.dll. When a request such as https://Server_name/File_name.asp is received, IIS runs the ASP.dll ISAPI extension to service the request and

load that file for processing. Many applications that run on IIS are actually ISAPI extensions that are script-mapped to process files with specific file name extensions.

ISAPI Extension Processing Sequence

The following events occur when IIS receives a request that maps to an ISAPI extension:

1. IIS loads the DLL, if it is not already in memory. When the DLL is loaded, Windows automatically calls the optional DLL entry/exit function (usually **DllMain**). IIS then calls the extension's GetExtensionVersion entry-point function.

2. IIS performs minor preprocessing on the incoming request.

3. IIS creates and populates an EXTENSION_CONTROL_BLOCK structure to pass request data and callback function pointers to the extension.

4. IIS calls the ISAPI extension's HttpExtensionProc function, passing a pointer to the EXTENSION_CONTROL_BLOCK structure created for this request.

5. The ISAPI extension carries out the actions it was designed to perform: for example, reading more data from the client (as in a POST operation), or writing headers and data back to the client.

6. The extension informs IIS that it is finished processing the request by exiting the HttpExtensionProc function. For synchronous operations, the function returns the HSE_STATUS_SUCCESS return code; for asynchronous operations, the return code is HSE_STATUS_PENDING. For more information about asynchronous operations, see Asynchronous I/O Processing.

7. IIS performs cleanup on the connection used for the request, after which it closes the connection if Keep-Alive functionality is not enabled.

8. Once the ISAPI extension is no longer needed, IIS calls the TerminateExtension function, if the extension provides one. If IIS is configured to cache ISAPI extensions, TerminateExtension is not called until the IIS Web server is shut down or restarted.

Note

GetExtensionVersion is not called for every request. In contrast, HttpExtensionProc is called exactly once for every request for the ISAPI extension. Additionally, one EXTENSION_CONTROL_BLOCK structure is used for each incoming request.

ISAPI Compared to CGI

The Internet Server Application Programming Interface (ISAPI) model was developed as a faster alternative to the Common Gateway Interface (CGI). ISAPI provides a number of advantages over CGI, including lower overhead, faster loading, and better scalability. The chief difference between the CGI and ISAPI programming models is how processing is handled.

With CGI, the system creates a unique process for every request. Each time an HTTP server receives a request, it initiates a new process. Because the operating system must maintain all these processes, CGI requires many of resources. This inherent limitation makes it difficult to develop responsive Internet applications with CGI.

With ISAPI, requests do not require a separate process. Threads are used to isolate and synchronize work items, resulting in a more efficient use of system resources. For more information, see ISAPI and Web Application Architecture.

An ISAPI extension differs from a CGI executable file in several other ways. An ISAPI extension does the following:

- Receives most of its data through the **lpbData** member of the EXTENSION_CONTROL_BLOCK structure, instead of reading the data from the standard input (STDIN) handle for the process. For any additional data, the extension uses the ReadClient callback function.
- Sends data back to the client using the WriteClient callback function, instead of writing to the standard output (STDOUT) handle for the process.
- Accesses common CGI variables through the **EXTENSION_CONTROL_BLOCK** structure. For other variables, the extension calls the GetServerVariable function. In a CGI executable file, these variables are retrieved from the environment table by using **getenv**.
- Specifies completion status by either sending the header directly using the WriteClient callback function, or by calling the HSE_REQ_SEND_RESPONSE_HEADER_EXServerSupportFunction, instead of writing the header to STDOUT.
- Redirects requests with a Location: or URL: header. If the URL is local, the extension uses the HSE_REQ_SEND_URL structure instead of writing the header to STDOUT. If the URL is remote or unknown, the extension uses HSE_REQ_SEND_URL_REDIRECT_RESP in the ServerSupportFunction callback function. When IIS receives a request for a particular extension, it loads the DLL into memory, where it services other requests. When IIS unloads the extension, it calls the extension's TerminateExtension function if it is present. Use of **TerminateExtension** is recommended to free any resources that the extension may have locked or allocated during its initial Load.Keep-Alive connections.
