ISAPI Filter Overview

By Archiveddocs

Archived: 2026-04-05 23:21:33 UTC

ISAPI filters are DLL files that can be used to modify and enhance the functionality provided by IIS. ISAPI filters always run on an IIS server, filtering every request until they find one they need to process. The ability to examine and modify both incoming and outgoing streams of data makes ISAPI filters powerful and flexible.

Filters are registered at either the site level or the global level (that is, global filters apply to all sites on the IIS server), and are initialized when the worker process is started. A filter listens to all requests to the site on which it is installed.

Both ISAPI filters and ISAPI extensions can only be developed using C/C++. Visual Studio comes with wizards that make ISAPI development fast and easy.

ISAPI filters can be registered with IIS to modify the behavior of a server. For example, filters can perform the following tasks:

- Change request data (URLs or headers) sent by the client
- Control which physical file gets mapped to the URL
- Control the user name and password used with anonymous or basic authentication
- Modify or analyze a request after authentication is complete
- Modify a response going back to the client
- Run custom processing on "access denied" responses
- Run processing when a request is complete
- Run processing when a connection with the client is closed
- Perform special logging or traffic analysis.
- Perform custom authentication.
- Handle encryption and compression.

Note

ISAPI filter DLLs cannot be requested explicitly, like ISAPI extensions can.

Every ISAPI filter is contained in a separate DLL that must export two entry-point functions, GetFilterVersionand HttpFilterProc, and optionally export the TerminateFilter function. The metabase property, FilterLoadOrder,

contains a list of all filters that IIS loads when the Web service is started.

1. When IIS initially loads an ISAPI filter, it also creates and partially populates an HTTP_FILTER_VERSION structure. It then calls the filter's GetFilterVersion function, passing a pointer to the new structure as a parameter.

2. The ISAPI filter populates the **HTTP_FILTER_VERSION** structure with version information and descriptive information. More importantly, the filter also uses **HTTP_FILTER_VERSION** to specify which event notifications it should receive, and to declare the general priority level for the filter. In addition, the filter also indicates whether it is interested in events from secure ports only, unsecure ports only, or both.

3. Each HTTP transaction between IIS and a client browser triggers several distinct events. Every time an event occurs for which an ISAPI filter is registered, IIS calls the filter's **HttpFilterProc** entry-point function.

   If more than one ISAPI filter is registered for a given event, IIS notifies the filters that the event occurred. The filters, which are marked as high, medium, or low priority, are notified according to priority in descending order. If more than one ISAPI filter is declared the same general priority level, IIS uses the order in which the filters appear in the FilterLoadOrder property to resolve the tie.

4. The ISAPI filter uses the notification type information, passed by IIS as a parameter to **HttpFilterProc**, to determine which particular data structure is pointed to by the other **HttpFilterProc** parameter, pvNotification. The ISAPI filter then uses the data contained in that data structure, as well as in the context structure HTTP_FILTER_CONTEXT, to perform any custom processing.

5. Once processing is complete, the filter returns one of the SF_STATUS status codes to IIS, and IIS continues processing the HTTP request or response until another event occurs for which ISAPI filters are registered.

6. When the Web service is stopped or unloaded, IIS calls **TerminateFilter** in all ISAPI filters as part of its shutdown sequence, for any filters that implemented and exported the function. **TerminateFilter** is typically used to perform cleanup and de-allocation of allocated resources.

GetFilterVersion is called exactly once, when the ISAPI filter is initially loaded. Any per-connection initializations the developer wants to perform must be managed internally within the context of the **HttpFilterProc** function call.

Note

The priority setting for ISAPI filters is per filter, not per notification. For example, a low priority rating cannot be assigned for one type of notification and a high priority rating for another type.
