# ServiceProviderHandler

In the SP software, the term *handler* is used to refer to functionality that is exposed by the SP filter to the browser at a particular URL. When a URL corresponding to a handler is requested, the SP filter services the request instead of passing it on to the web server. You can think of it like a "virtual" resource that isn't a real file in your document tree, but looks real to the client.

## Getting Handlers to Run

Exposing a handler is very different across web server environments, but is generally done by reserving a file extension and sometimes an internal content-type for all handlers. As of ShibOnedotThree, the default extension used is **.sso**, but this can be changed with some configuration work if necessary.

In early versions of Apache, a handler can be assigned to the file extension like so:

```
<Files *.sso>
SetHandler shib-handler
</Files>
```

In later versions, the handler mechanism hasn't worked quite so well, and the software instead relies on intercepting every web request and determining from the URL whether the request should be routed to a handler or not.

In IIS, the handler mechanism is implemented as an ISAPI *extension* contained in the same library with the ISAPI *filter*. Extensions can process web requests properly, including POST data, while filters alone cannot. To cause the extension to run properly, a "script mapping" must be defined for the IIS web site for the file extension to the ISAPI extension DLL.

In iPlanet, the handler mechanism is implemented as an NSAPI service function, which is installed based on the file extension in **obj.conf**.

## Structure of Requests

A request to a handler is just a normal URL whose path component begins with a prefix that corresponds to the `handlerURL` attribute that is defined in the `<Sessions>` element in the ShibbolethApplication that corresponds to the request. In most deployments, the prefix is `/Shibboleth.sso`

This prefix is what signals to the filter that instead of processing the request in a standard way, the handler mechanism should (or will) be invoked later to handle it. This prevents paradoxes like "protecting Shibboleth functionality with itself" that would result in endless loops.

The rest of the URL (up to the query string, if any) is interpreted as something called PATH-INFO, which isn't used that often anymore, but allows data on the URL path following a resource to be evaluated by that resource. In this case, you can think of the PATH-INFO as the "function" for the handler to run.

The ShibbolethXml file, within the `<Sessions>` element, allows a deployer to install a variety of functions to be executed by the Shibboleth handler mechanism by defining the PATH-INFO string that corresponds to them. A few are well-defined by the software to perform functions critical to the system, but an extension mechanism also exists for user-defined functionality or future use.

The following are built-in function types:

- SessionInitiator
- AssertionConsumerService
- SingleLogoutService
- DiagnosticService (none implemented so far)

The rest of the data that makes up the request (query string, form parameters, etc.) is all simply raw data to Shibboleth. The handler function that executes based on the PATH-INFO is responsible for doing whatever it wants to do with the request or signaling an error.