# Retrieving Data Using Script

Retrieving Data Using Script
Authentication Using Script

3 readers found this helpful (6 ratings - Rate this content)

This topic includes an example of how to write a script that obtains data through Microsoft Windows HTTP Services (WinHTTP) either synchronously or asynchronously. The concepts demonstrated in this example provide the basis for writing client or middle-tier server applications that require access to data using the HTTP protocol.

- Prerequisites and Requirements
- Retrieving Data Synchronously
- Retrieving Data Asynchronously
- Related topics

## Prerequisites and Requirements

In addition to a working knowledge of Microsoft JScript, this example requires the following:

- The current version of the Microsoft Windows Software Development Kit (SDK).
- The proxy configuration tool to establish the proxy settings for Microsoft Windows HTTP Services (WinHTTP), if your connection to the Internet is through a proxy server. For more information, see ProxyCfg.exe, a Proxy Configuration Tool.
- A familiarity with network terminology and concepts.

## Retrieving Data Synchronously

**To create a script that obtains the text from a Web page synchronously, do the following:**

1. Open a text editor.
2. Copy the following code into the text editor.

```
function getText(strURL)
{
    var strResult;

    try
    {
        // Create the WinHTTPRequest ActiveX Object.
        var WinHttpReq = new ActiveXObject(
                                  "WinHttp.WinHttpRequest.5.1");

        //  Create an HTTP request.
        var temp = WinHttpReq.Open("GET", strURL, false);

        //  Send the HTTP request.
        WinHttpReq.Send();

        //  Retrieve the response text.
        strResult = WinHttpReq.ResponseText;
    }
    catch (objError)
    {
        strResult = objError + "\n"
        strResult += "WinHTTP returned error: " +
            (objError.number & 0xFFFF).toString() + "\n\n";
        strResult += objError.description;
    }

    //  Return the response text.
    return strResult;
}

WScript.Echo(getText("http://www.microsoft.com/default.htm"));
```

3. Save the file as "Retrieve.js".
4. From a command prompt, type "cscript Retrieve.js" and press ENTER.

You now have a script that uses a **WinHttpRequest** object to obtain the HTML source code for the Web page at http://www.microsoft.com. You might have to wait several seconds for the code to appear.

The application contains only one function, "getText". The first line of the script creates the **WinHttpRequest** object.

```
// Create the WinHTTPRequest ActiveX Object.
var WinHttpReq = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
```

When the JScript engine encounters this line, it creates an instance of this object. If you get the error message, "ActiveX component can't create object", on this line, most likely the WinHttp.dll was not properly registered or is not present on the system.

The next line of the script calls the **Open** method.

```
//  Create an HTTP request.
WinHttpReq.Open("GET", "http://www.microsoft.com", false);
```

Three parameters specify which *HTTP verb* to use, the name of the resource, and whether to use WinHTTP synchronously or asynchronously. In this example, the method is using the HTTP verb"GET" to obtain data from http://www.microsoft.com. Specifying **FALSE** for the last parameter determines that the transaction occurs synchronously. The **Open** method does not establish a connection to the resource as the name might imply. Rather, it initializes the internal data structures that maintain information about the session, connection, and request.

The **Send** method assembles the request headers and sends the request. When called in synchronous mode, the **Send** method also waits for a response before allowing the application to continue.

```
    // Send the HTTP request.
    WinHttpReq.Send();
```

After sending the request, the script returns the value of the **ResponseText** property of the **WinHttpRequest** object. This property contains the entity body of the response, in this case, the source of a document.

```
    // Get the response text.
    return WinHttpReq.ResponseText;
```

Execution of the script pauses while the entire text of the resource is retrieved. The resource text is returned from the function and displayed.

The **WinHttpRequest** object ensures that any internal resources that are allocated for the HTTP transaction are released.

## Retrieving Data Asynchronously

Retrieving data asynchronously using WinHTTP is very similar to retrieving data synchronously. Modify the script from the previous section by making two small changes.

1. Set the third parameter of the **Open** method to "true" instead of "false" to specify that WinHTTP methods should be performed asynchronously.

   ```
       //  Create a HTTP request.
        var temp = WinHttpReq.Open("GET", strURL, true);
   ```

2. Invoke the **WaitForResponse** method before accessing the **ResponseText** property to ensure that the entire response has been received.

   ```
        //  Send the HTTP request.
        WinHttpReq.Send();

        // Wait for the entire response.
        WinHttpReq.WaitForResponse();

        //  Retrieve the response text.
        strResult = WinHttpReq.ResponseText;
   ```

The main advantage to using WinHTTP asynchronously in script is that the **Send** method returns immediately. The request is prepared and sent by a worker thread. This enables your application to do other things while it is waiting for the response. Before attempting to access the response, ensure that the entire response has been received by calling the **WaitForResponse** method. Otherwise an error can occur.

The **WaitForResponse** method can also be used to specify a time-out value for the transaction. An optional parameter enables you to specify the time-out value in seconds.

## Related topics

**WinHttpRequest**
HTTP/1.1 Request for Comments (RFC 2616)

Did you find this helpful? Yes No

## Community Additions

ADD

Network sites

msdn

Other links

Support

Microsoft Connect

Windows dev center

Metro style apps

Internet Explorer

Desktop

Hardware

Microsoft