Lab 14.3: Building an ISAPI Filter

Objectives

After completing this lab, you will be able to:

- Use the ISAPI Extension Wizard to create an ISAPI filter project with Microsoft Visual C++.
- Implement the project, including:
  - Implementing handler member functions corresponding to the server notification events.
  - Implementing the handlers and supporting functions and classes.
- Install and test the ISAPI filter.

Prerequisites

Before attempting this lab, you be thoroughly familiar with the material presented in Chapter 14, "Creating and Using ISAPI Extensions."

Lab Setup

To see a demonstration for the solution to this lab, click this icon.

Estimated time to complete this lab: **90 minutes**.

Exercises

The following exercises provide practice working with the concepts and techniques covered in this chapter.

Exercise 1: Creating the ISAPI Filter Project

In this exercise, you will use Microsoft Visual C++ to create an ISAPI filter project. You will select and edit various options in the ISAPI Extension Wizard dialogs, as appropriate for an employee-authentication filter.

Exercise 2: Implementing the ISAPI Filter

In this exercise, you will implement the ISAPI filter you created in Exercise 1. To accomplish this, you will implement event-handler member functions, and add and implement helper member functions.

Exercise 3: Testing the ISAPI Filter

In this exercise, you will install the ISAPI filter you implemented in Exercise 2 onto your ISAPI-compliant Web server, and test it with Internet Explorer.

There is no setup for this lab. The completed code for these exercises is in \Labs\C14\Lab03\Xxx, where Xxx is the exercise number.

To test this component, you must have administrative privileges on a computer running Windows NT and a Microsoft Web service, such as Internet Information Server or Peer Web Services. Alternatively, the labs can be run on a Windows 95 machine running Personal Web Server.

Exercise 1: Creating the Project

In this exercise, you will use Microsoft Visual C++ to create the EmpOnly ISAPI filter project. You will select and edit various options in the ISAPI Extension Wizard dialogs, as appropriate for an employee-authentication filter.

The EmpOnly filter will intercede in the Web server processing at the point where the URL is mapped to a physical resource (a file). This filter will authenticate users trying to get access to the Employees Only Web page of the Main Street Market site. If they are not on a list of authorized employees, then EmpOnly will redirect the user to a "consolation" page.

➢ **Create a new project for EmpOnly**

1. Start Microsoft Developer Studio. From the File menu, choose New.
2. In the New dialog box, select the Projects tab.
3. Supply the following information in the New Project Workspace dialog:
   - Type – Select ISAPI Extension Wizard
   - Name – Enter **EmpOnly**
   - Platform – Check Win32 (default)
   - Location – Enter or browse for \LABS\C14\LAB03

Then choose the OK button. The ISAPI Extension Wizard dialog, Step 1 of 1, will display.

4. In the Wizard dialog, make the following corrections and verifications:
   - The Generate A Filter Object check box should be selected, and Generate A Server Extension Object should be cleared.

Note When the Generate A Filter Object option is selected, the ISAPI Extension Wizard adds a second dialog to its interface. This is reflected by the Wizard title, which changes to Step 1 of 2.

   - Change the Filter Description to EmpOnly ISAPI Filter.
   - Use the MFC libraries As A Shared DLL.

When you are done, choose Next.

5. In Step 2 of 2, set the following options:
   - Medium priority level
   - Non-Secured Port Session connection type only
   - URL Mapping Requests notification type only

   Here it is assumed that if the user is trusted enough to make a secured connection to the Web server, then there is no need for further authentication.

When you are done, choose Finish.

6. A Summary dialog box appears. After you review the information in it, choose OK to generate the new project.

➢ **Use the Project Workspace to investigate EmpOnly**

1. Select the ClassView pane of the Project Workspace if it is not already displayed. Expand all the branches.
2. Double-click the following entries to view the associated source code:
   - **CEmpOnlyFilter** – to view the CHttpFilter-derived class declaration. Note the declaration of the overridden notification event handler, CHttpFilter::OnUrlMap.
   - **GetFilterVersion** – to view the initialization member function that the Wizard generated for the ISAPI filter.
   - **OnUrlMap** – to view the (mostly empty) event handler generated by the Wizard.
   - **theFilter** – to view the single instantiation of the CHttpFilter-derived class.
3. Select the ResourceView pane. Expand all the branches.
4. Double-click the following entries to view the associated Windows resource.
   - String Table – has a single string resource, IDS_FILTER, the string description used by **CEmpOnlyFilter::GetFilterVersion** to register an extension description with the Web server. You supplied this string when you ran the ISAPI Extension Wizard in the previous section.
   - VS_VERSION_INFO – a standard editable program version resource.
5. In the FileView pane, open EmpOnly.Def. Note that it exports only the two entry points that every ISAPI filter must have: the global C functions **HttpFilterProc** and **GetFilterVersion**.

➢ **Test the EmpOnly project**

1. Build the project, targeting Win32 Debug. EmpOnly should compile and link cleanly.
2. Close all the source windows, except the ones for files EmpOnly.H and EmpOnly.Cpp. You will use these files in subsequent exercises of this lab.

The completed code for this exercise is in \Labs\C14\Lab03\Ex01.

Exercise 2: Implementing the ISAPI Filter

In this exercise, you will implement the EmpOnly ISAPI filter. To accomplish this, you will implement the filter's constructor and the override of the **CHttpFilter::OnUrlMap** event-handler member function, and add and implement helper member functions.

You also will examine the two supplemental files that the EmpOnly filter will use: Employ.Dat and Consoltn.Htm.

➢ **Examine the supplemental files Employ.dat and Consoltn.htm**

1. Use the Windows Explorer to locate Employ.Dat and Consoltn.Htm in \Labs\C14\Lab03.
2. Open and examine the file Employ.Dat in a text editor.

   Employ.Dat is a text file that represents the database of authorized Main Street Market employees. This file contains a sample list of employee names and client machine IP addresses.
3. In the Windows Explorer, double-click Consoltn.Htm to open it in Internet Explorer.

   Consoltn.Htm is the consolation page that unauthorized users will see when they try to access the Employees Only page.

➢ **Edit the CEmpOnlyFilter class declaration**

To complete the EmpOnly ISAPI filter, you will create two new helper functions and a data member. Both will be declared in a new, protected implementation section.

1. In Class View, right-click **CEmpOnlyFilter** and add the following two protected functions (these are helper functions).

```
BOOL IsAuthUserIP(LPCSTR lpszUserName)
BOOL IsAuthUserName(LPCSTR lpszIPAddress)
```

2. In Class View, right-click **CEmpOnlyFilter** and add the following protected member variable.

```
CString strAuthUsers
```

➢ **Implement the constructor for CEmpOnlyFilter**

A filter's constructor is called only once. This occurs when the Web publishing service is started and it loads all ISAPI filters into memory. Therefore, the constructor is a good place to perform one-time, filter-initialization processing. EmpOnly will use it to read the employees database file into the class member strAuthUsers.

1. Locate **CEmpOnlyFilter::CEmpOnlyFilter** in the file EmpOnly.Cpp for editing.
2. Define the following two local variables:
   - A CString named **buf**.
   - A CStdioFile object named **infile**. Create this object by opening the file Employ.Dat in read-only mode. You must supply the absolute directory path to your server's \Scripts subdirectory. For example:

```
c:\\Winnt\\System32\\InetSrv\\Scripts\\Employ.dat
```

or

```
c:\\Webshare\\Scripts\\Employ.dat
```

   Note that the CStdioFile constructor can cause a file exception (CFileException).

3. Create a **while** loop whose condition is based on the result of a read operation. Use CStdioFile::ReadString to fill the local buffer (buf) by reading a line from Employ.Dat. In the body of the loop, concatenate strAuthUsers with buf and a semicolon (as a record separator). Then empty buf to prepare it for the next read.
4. The code for the completed function follows:

```
CEmpOnlyFilter::CEmpOnlyFilter()
{
    // Read in employee "database" text file - Employ.dat.
    // Employ.dat must be located in Scripts subdirectory.
    // Since ctor is only called when EmpOnly.dll is loaded into memory,
    // any changes to database demands that web service be restarted.

    CString buf;

    // This next path is specific to the web server installation!
    // CStdioFile infile("c:\\Winnt351\\System32\\InetSrv\\Scripts\
\Employ.dat",
    //                   CFile::modeRead);

    CStdioFile infile("c:\\webshare\\Scripts\\Employ.dat",
                       CFile::modeRead);

    while (infile.ReadString(buf))
    {
        strAuthUsers += buf + "; ";
        buf.Empty();
    }
}
```

➢ **Implement the body of the event handler CEmpOnlyFilter::OnUrlMap**

This member function is responsible for the main logic of the filter. In **OnUrlMap**, you will determine whether the request is for the Employees Only page. If it is, you will check the client name and IP address for a match in the employee database, and take the appropriate action.

1. In the body of **CEmpOnlyFilter::OnUrlMap**, define the following new local variables:
   - Two character arrays, **pstrName[100]** and **pstrIP[20]**, and a character pointer **pstrSearch**.
   - A DWORD named **dwSize**.
   - A Boolean variable named **bIsAuth**, initialized to FALSE.
2. Check to see whether the client is requesting the Employees Only page:
   a. Prepare the pszPhysicalPath member of the HTTP_FILTER_URL_MAP structure by invoking _strlwr on it. This is necessary because the file names in Windows are case-insensitive.
   b. Use the ANSI function strstr to determine whether the file EmpOnly.Htm can be located in the pszPhysicalPath member. Use the variable pstrSearch to store the name match location.
   c. If this search failed, return SF_STATUS_REQ_NEXT_NOTIFICATION.
3. Obtain the employee's name, and check it against the employee database. Assume that all employee names are required to be at least six characters.
   a. Set dwSize to the size of the pstrName array.
   b. Call **CHttpFilterContext::GetServerVariable** to place the name of the remote user into pstrName.
   c. If the size of the employee name is greater than five, call the **CEmpOnlyFilter::IsAuthUserName** helper function to determine whether the user is an authorized employee. Store the result in bIsAuth.
4. If the user's name was not found, check the IP address for authorization. (Assume that an IP address must be at least seven characters.) If bIsAuth is FALSE, then:

a. Set dwSize to the size of the pstrIP array.

b. Call **CHttpFilterContext::GetServerVariable** to place the address of the remote user into pstrIP.

c. If the size of the address is greater than six characters, call the **CEmpOnlyFilter::IsAuthUserIP** helper function to determine whether the user is an authorized employee. Store the result in the variable bIsAuth.

5. Finally, check the value of bIsAuth to determine which resource the user will see. If this value is TRUE, return SF_STATUS_REQ_NEXT_NOTIFICATION to allow the user access to the Employees Only page (pending other filter processing).

   If bIsAuth is FALSE, then redirect the user to the consolation page:

   a. Call strcpy to copy the literal Consoltn.Htm file over the value EmpOnly.Htm, which is stored in pstrSearch.

   b. Return SF_STATUS_REQ_HANDLED_NOTIFICATION to indicate that no further processing of this notification event should occur.

6. The completed code for the **OnUrlMap** function follows.

```
DWORD CEmpOnlyFilter::OnUrlMap(CHttpFilterContext* pCtxt,
    PHTTP_FILTER_URL_MAP pMapInfo)
{
    char pstrName[100], pstrIP[20], *pstrSearch;
    DWORD dwSize;
    BOOL bIsAuth = FALSE;

    // Check to see if client is requesting Employees Only page.
    // Use of _strlwr is possible because NT does not differentiate on
case.
    _strlwr(pMapInfo->pszPhysicalPath);
    pstrSearch = strstr(pMapInfo->pszPhysicalPath, "emponly.htm");
    if (pstrSearch == NULL) //not asking for Employees Only page
        return SF_STATUS_REQ_NEXT_NOTIFICATION;

    dwSize = sizeof(pstrName);
    pCtxt->GetServerVariable("REMOTE_USER", pstrName, &dwSize);
    if (dwSize > 5)  //check that is valid user (not anonymous)
        bIsAuth = IsAuthUserName(pstrName);

    if (bIsAuth == FALSE) //if not authorized name, then check IP
    {
        dwSize = sizeof(pstrIP);
        pCtxt->GetServerVariable("REMOTE_ADDR", pstrIP, &dwSize);
        if(dwSize > 6) //should always have client's IP, but...
            bIsAuth = IsAuthUserIP(pstrIP);
    }

    if (bIsAuth == TRUE)  //if user is an authorized employee
    {
        return SF_STATUS_REQ_NEXT_NOTIFICATION;
    }
    else  //if not, then redirect to consolation page
    {
        strcpy(pstrSearch, "Consoltn.htm");
        //preclude any other filter processing on this event
        return SF_STATUS_REQ_HANDLED_NOTIFICATION;
    }
}
```

➢ **Implement the helper functions**

For this example, the implementations of **CEmpOnlyFilter::IsAuthUserName** and **CEmpOnlyFilter::IsAuthUserIP** will be almost identical. Because interaction with the employee database has been encapsulated into the filter's constructor and the two helper functions, replacing it with a more sophisticated scheme should be straightforward.

1. In the body of IsAuthUserName, define an integer named **hit**.
2. Invoke the member function **CString::Find** on **CEmpOnlyFilter::strAuthUsers** to see whether the employee database contains the argument string. Store the returned value in hit.
3. If hit has a value of -1, return FALSE; otherwise, return TRUE.
4. Copy this implementation and paste it into **CEmpOnlyFilter::IsAuthUserIP** body. Ensure that the CString::Find function's argument name is correct.
5. The completed code for both functions follows.

```
BOOL CEmpOnlyFilter::IsAuthUserName(LPCSTR lpszUserName)
{
    int hit = strAuthUsers.Find(lpszUserName);
    if (hit == -1)
        return FALSE;
    else
        return TRUE;
}

BOOL CEmpOnlyFilter::IsAuthUserIP(LPCSTR lpszIPAddress)
{
    int hit = strAuthUsers.Find(lpszIPAddress);
    if (hit == -1)
        return FALSE;
    else
        return TRUE;
}
```

Note This implementation of the EmpOnly filter is not secure. With some knowledge of employee accounts, simple IP address and user name spoofing would allow access to the Employees Only page.

➢ **Build the EmpOnly Filter**

The completed code for this exercise is in \Labs\C14\Lab03\Ex02.

Exercise 3: Testing the ISAPI Filter

In this exercise, you will install the EmpOnly ISAPI filter onto your ISAPI-compliant Web server. To accomplish this, you will copy the filter DLL and employee database file to the \Scripts subdirectory of the Web server. Then, you will copy the EmpOnly.htm and Consoltn.htm files to the \WWRoot subdirectory. Finally, you will edit the registry to add an entry for this filter.

You also will test the EmpOnly filter with Internet Explorer. You will try to access the Employees Only page, EmpOnly.htm, before and after you add a client entry for yourself in the employees database.

➢ **Install the EmpOnly ISAPI filter on a Microsoft Web server**

Installing an ISAPI filter is slightly more complicated than installing an ISAPI application. You copy the filter DLL and supporting program files into the \Scripts subdirectory of your Web server. Then you copy the supporting Web pages to the \WWWRoot subdirectory.

In addition to these steps, you register ISAPI filters in the Windows system registration database. When a Microsoft Web server starts, it loads all the registered ISAPI filters.

1. To stop the Web service, forcing it to unload an ISAPI application, use one of the following techniques:

- Reboot Windows to initialize the Web server.
- Use the Internet Service Manager to stop the Web service.
- Use the Web-based Service Administrator to stop the Web service.
- Use the Services applet of the Control Panel to stop the Web service.
- From the command line, issue the command **net stop W3Svc** (Use **net start W3Svc** to restart the Web publishing service.)
- If you are running Windows 95 and Personal Web Server, open the Personal Web Server applet from the Control Panel. Select the Startup tab and choose the Stop button. If the Personal Web Server icon appears in the taskbar for Windows 95, you can right-click the icon and select Properties instead going through the Control Panel.

2. Copy the file EmpOnly.dll to the \Scripts subdirectory of your Microsoft Web server.
3. Copy the employee database, Employ.dat, to the \Scripts subdirectory. This file can be found in the \Labs\C14\Lab03 directory.
4. Copy the EmpOnly.htm and Consoltn.htm files, to the server's \WWWRoot directory. This file also can be found in the \Labs\C14\Lab03 directory.
5. Add an entry for EmpOnly to the Windows registration database:
   a. Start the Windows system registration database editor, RegEdit.exe.
   b. Locate the entry Filter DLLs under the key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W3Svc\Parameters
   c. Modify this entry's value data by appending the absolute path of the EmpOnly filter. For example:

```
c:\WinNT\System32\InetSrv\Scripts\EmpOnly.dll
```

or

```
c:\Webshare\Scripts\EmpOnly.dll
```

   If there are already filters registered, precede the new entry with a comma separator.
6. Restart the Web service, using one of the techniques listed in Step 1.

➢ **Test EmpOnly with the Internet Explorer**

1. Test the action of the EmpOnly ISAPI filter for an unauthorized user by entering the following URL in Internet Explorer:

   `http://<server-name>/EmpOnly.htm`

   Because your name and IP address are not in the employee database file, EmpOnly.dll redirects your request to the Consolation page, Consoltn.htm.
2. Edit the text file Employ.dat, and add a line with your full name, user name, and the IP address of your client machine.

   If you are using anonymous access, then your user name is unimportant. To obtain the IP address of your client machine, run IPConfig.exe or WinIPCfg.exe from the command line.
3. Stop the WWW publishing service by using one of the techniques listed in Step 1, and then restart it.

   This will force the EmpOnly filter to reread the employee database.
4. Choose Refresh in Internet Explorer to update the page. This time, you should be an authorized employee, so the Employees Only page should appear.

As an alternate method, you can use the Windows NT challenge-response logon mechanism, which is supported by Microsoft Internet Information Server and Internet Explorer. You can restrict access by setting the security permissions in the file EmpOnly.htm, however, redirection to another page would not occur for unauthorized personnel, but would only display a server error message.

The complete solution for this lab is in \Labs\C14\Lab03\Ex03.