ILLiad ISAPI Filter – PDSAuth

Created by: Barnaby Alter, NYU Division of Libraries, Web Services
Last Updated: 3/26/2012

About the Project

The PDSAuth application is an ISAPI Filter developed in Visual C++ for Windows Server and IIS. Its function is to intercept requests to ILLiad in order to verify authentication via PDS and then make an authorization decision based on determining factors discussed below. It was developed as a modification to the CASAuthN filter developed at UC Davis, https://confluence.ucdavis.edu/confluence/display/IETP/CAS+ISAPI+Client.

Application Logic

Summary

The application checks on each request to see if there is a valid PDS session and if so redirects back to ILLiad with a local cookie containing the encrypted username. Then the filter decrypts this username and, if she is a valid ILLiad user, sets an HTTP response header that logs her in. The name for the custom HTTP header is set within the Customization Manager settings (see below).

Authorized?

Once authenticated via PDS, the filter determines if the user is authorized to use ILLiad based on two conditions: **borrower status** and **ILL permission flag**.

- If the user has a valid borrower status and has his ILL permission flag set to "Y" he will be passed through the initial authorization.
- Otherwise, if either of these conditions returns false, the user will be redirected to an access denied page managed in the CMS, this page can be set in the registry.

Existing Users

If the user is of a correct borrower status and exists in ILLiad's **Users** table, she will be logged in. This is the case of an **existing user**, that is, the user has successfully logged into the application before.

New Users

First time authorized users are a special case.

Note: Users in the **UserValidation** table are part of a load that our DBA performs each semester. Users created manually in Aleph during the course of the semester will not be in the UserValidation table unless manually entered.

- ILLiad will redirect the user to the authenticated user registration page (NewAuthRegistration.html, which we have appropriated as a copyright policy page). This is built-in ILLiad functionality: once the patron is authenticated they are recognized as a new user (i.e. entered into the Users table) and are therefore sent to a registration page to fill out additional user information. We require E-mail Address, First Name and Last Name.

- If the user is in the **UserValidation** table (editable through the Customization Manager) her information will be automatically transferred to the Users table and, after agreeing to the copyright policy, she will be logged in. This is thanks to a database level trigger setup by our DBA to run after each database insert (i.e. entering a new user).
- Otherwise, if the user is not in the UserValidation table, they will be presented with an error message saying to contact ILL staff about getting permission.
- **Note:** In this last case, if the user should indeed have access to ILL, ILL staff can enter the user into the UserValidation table and David will run a nightly script to sync up the data between the UserValidation and Users tables. Hence, users entered manually into the UserValidation table will experience a lag of at most 24 hours before having access to ILL.
- **Technical Note:** These error messages are actually manipulations of the required field error messages for E-Mail Address, First Name and Last Name. This seeming hack is in place because the case where that required information cannot be found is the case where the user is valid according to PDS but ILLiad doesn't know about him.

Environments

Development

Windows Server 2008 64-bit
IIS 7.5
MSXML >= 4.0 SP2
URL: http://illdev.library.nyu.edu

Production

Windows Server 2008 Standard Edition
IIS 7.5
MSXML >= 4.0 SP2
URL: http://ill.library.university.edu

Integration with ILLiad

RemoteAuth

You will need to make some changes to the ILLiad Customization Manager in order to tell ILLiad to look for a different authentication method.

The Customization Manager is a desktop application that can be found on the machines with ILLiad installed.

Follow these steps to activate RemoteAuth:

1. Open the Customization Manager. Browse to Web Interface>Authentication.
2. Change the following fields to match the subsequent values:

- **For Dev:**
    - RemoteAuthSupport = Yes
    - RemoteAuthUserVariable = PDSIlliadUser
    - RemoteAuthWebLogoutURL = https://pds.library.university.edu/logout?url=http://primo.library.university.edu
    - RemoteAuthWebPath = c:\inetpub\wwwroot\illiad\

    - WebAuthType = RemoteAuth (this field isn't used when RemoteAuth is active, but could be changed as a reminder)
- **For Production:**
    - RemoteAuthSupport = Yes
    - RemoteAuthUserVariable = PDSIlliadUser
    - RemoteAuthWebLogoutURL = https://pds.library.university.edu/logout?url=http://primo.library.university.edu
    - RemoteAuthWebPath = d:\inetpub\wwwroot\illiad\
    - WebAuthType = RemoteAuth (this field isn't used when RemoteAuth is active, but could be changed as a reminder)

No application or server restart is necessary to make these changes go into effect.

Installation – From Installer (recommended)

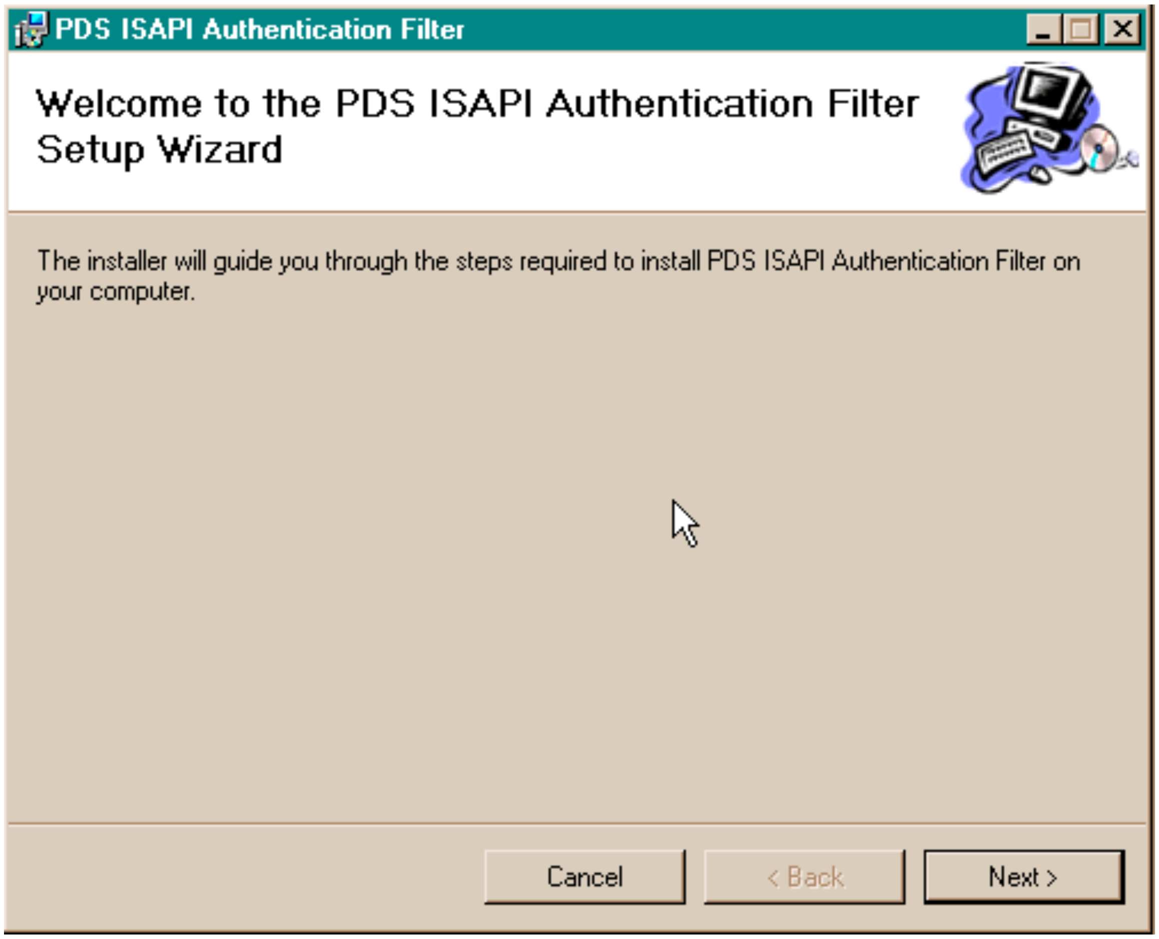
Dependencies

- MSXML 4.0 Service Pack 2 or higher - http://www.microsoft.com/downloads/details.aspx?familyid=3144b72b-b4f2-46da-b4b6-c5d7485f2b42&displaylang=en
- Create the HTTP wrapper to get SSL data from PDS **if your ILLiad is not served over HTTPS**. An example of what this wrapper can look like: http://webapps.library.university.edu/http_wrapper.php

```
<?php
header ("content-type: text/xml");
$calling_system = (isset($_REQUEST['calling_system'])) ?
$_REQUEST['calling_system'] : "illiaddev";
if (isset($_REQUEST['pds_handle'])) {
      echo
      file_get_contents("https://pds.library.univeristy.edu/pds?f
      unc=get-
      attribute&attribute=bor_info&calling_system=".$calling_syst
      em."&pds_handle=".$_REQUEST['pds_handle']);
}
```

Compile/Install the DLL

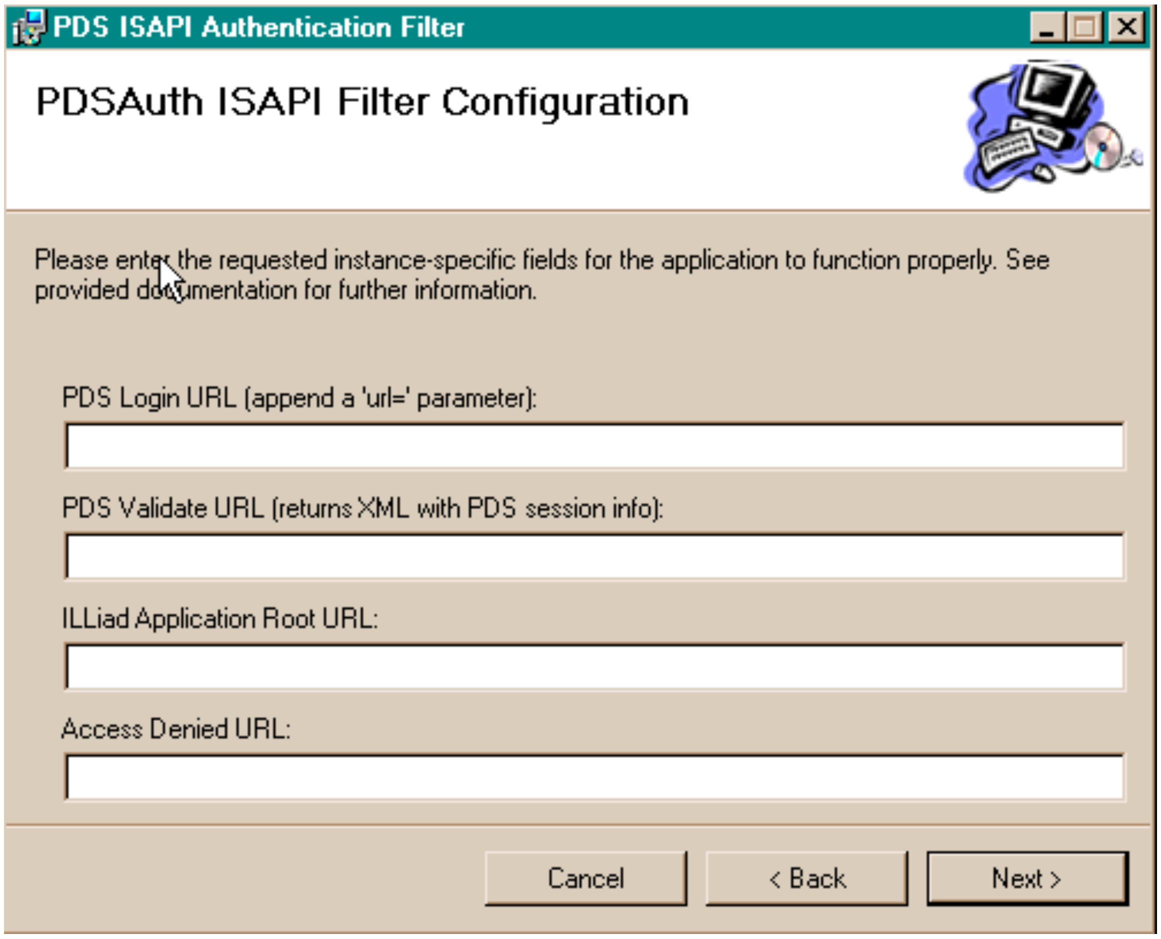
- Log into the computer on which you wish to install the filter
- After you've downloaded the application from subversion, navigate to *PDSAuth Installer/Release* and run the 'PDSAuth Installer.msi'
- You will be prompted with several windows:


PDS ISAPI Authentication Filter

Welcome to the PDS ISAPI Authentication Filter Setup Wizard

The installer will guide you through the steps required to install PDS ISAPI Authentication Filter on your computer.

Figure 1 – Click next


PDS ISAPI Authentication Filter

PDSAuth ISAPI Filter Configuration

Please enter the requested instance-specific fields for the application to function properly. See provided documentation for further information.

PDS Login URL (append a 'url=' parameter):

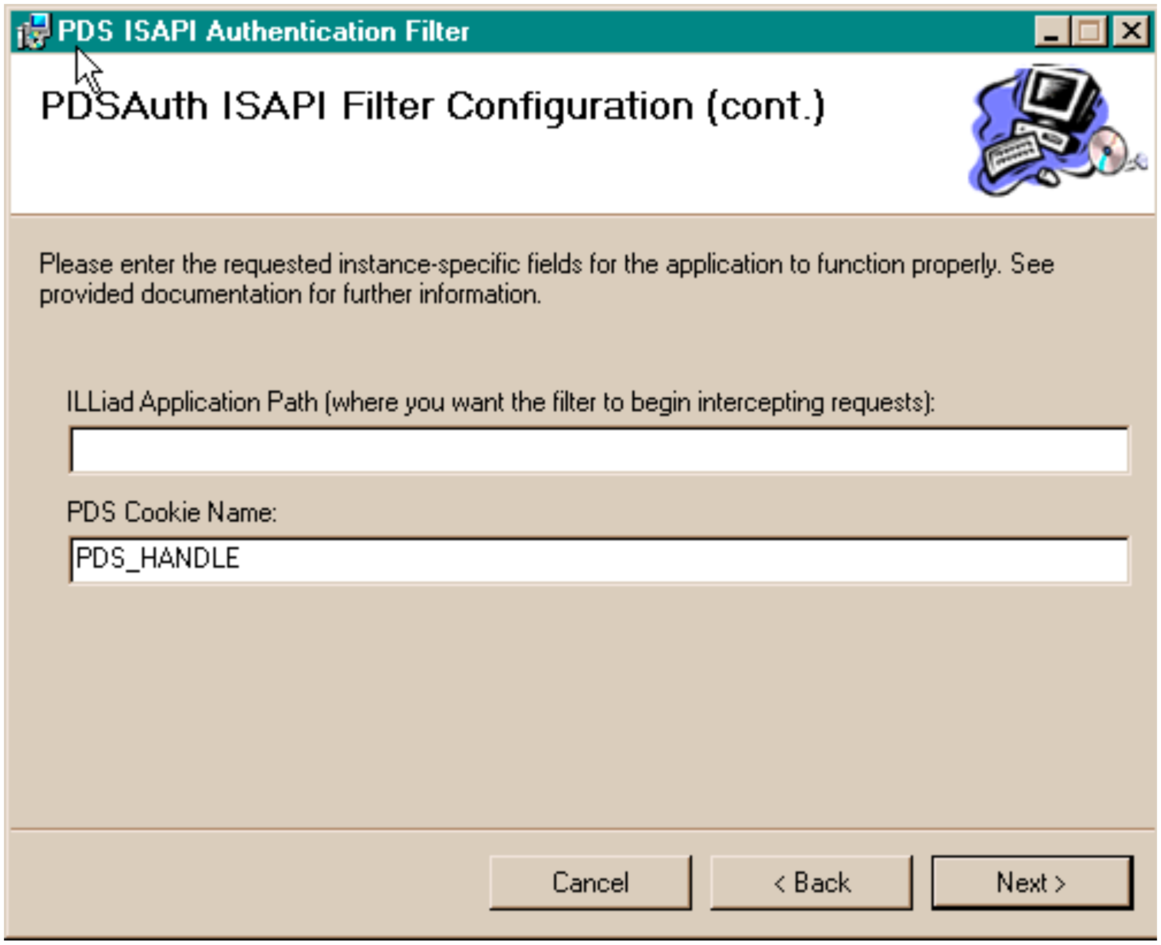
PDS Validate URL (returns XML with PDS session info):

ILLiad Application Root URL:

Access Denied URL:

Fig 2 - Fill in the fields as shown

- PDS Login URL: https://pds.library.university.edu/pds?func=load-login&institute=PRIMO&calling_system=illiad&url=
- PDS Validate URL: http://webapps.library.university.edu / http_wrapper.php?pds_handle=
- ILLiad Application Root URL: http://ill.library.university.edu
- Access Denied URL: http://library.university.edu/errors/access_denied.html


PDS ISAPI Authentication Filter

PDSAuth ISAPI Filter Configuration (cont.)

Please enter the requested instance-specific fields for the application to function properly. See provided documentation for further information.

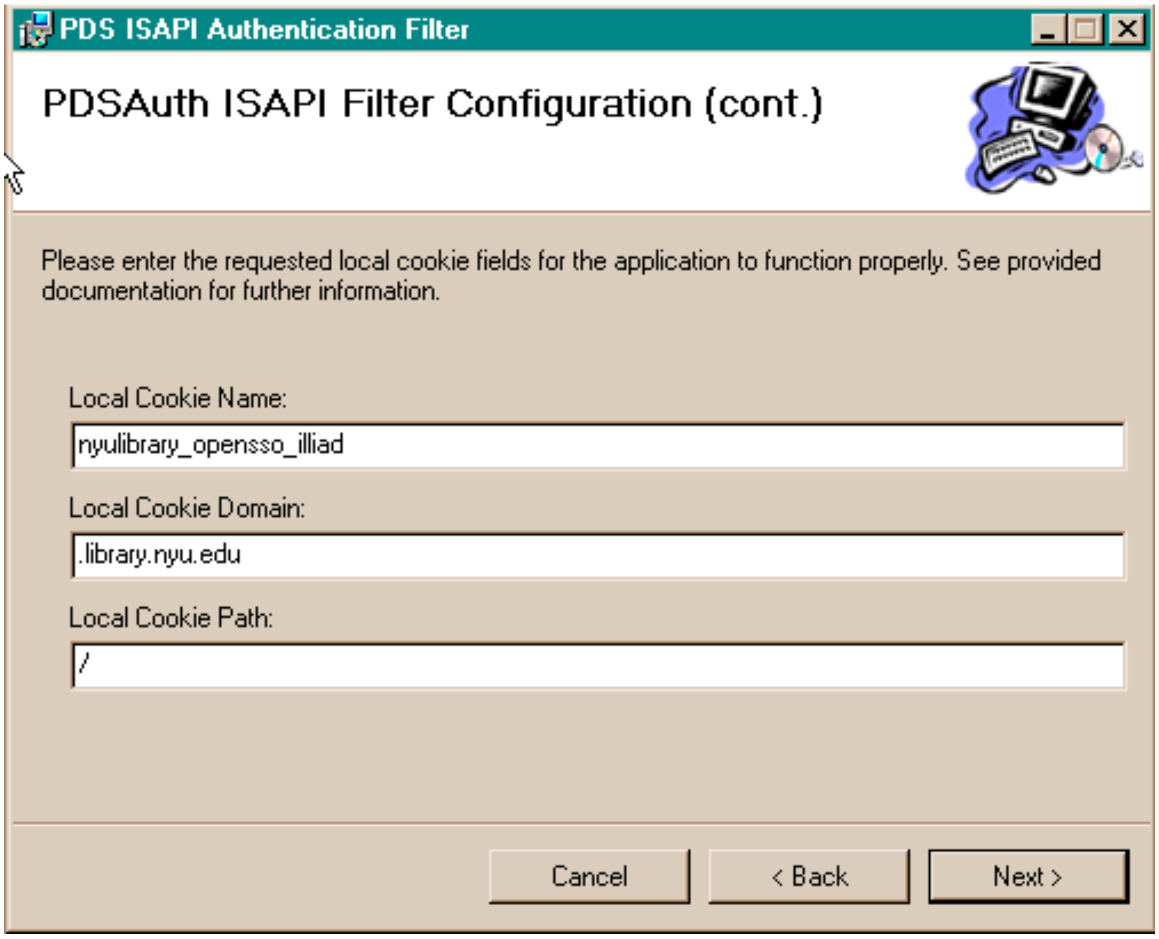
ILLiad Application Path (where you want the filter to begin intercepting requests):

PDS Cookie Name:

PDS_HANDLE

Fig 2b - Fill in the fields as shown

- ILLiad Application Path: /illiad/illiad.dll
- PDS Cookie Name: PDS_HANDLE



PDSAuth ISAPI Filter Configuration (cont.)

Please enter the requested local cookie fields for the application to function properly. See provided documentation for further information.

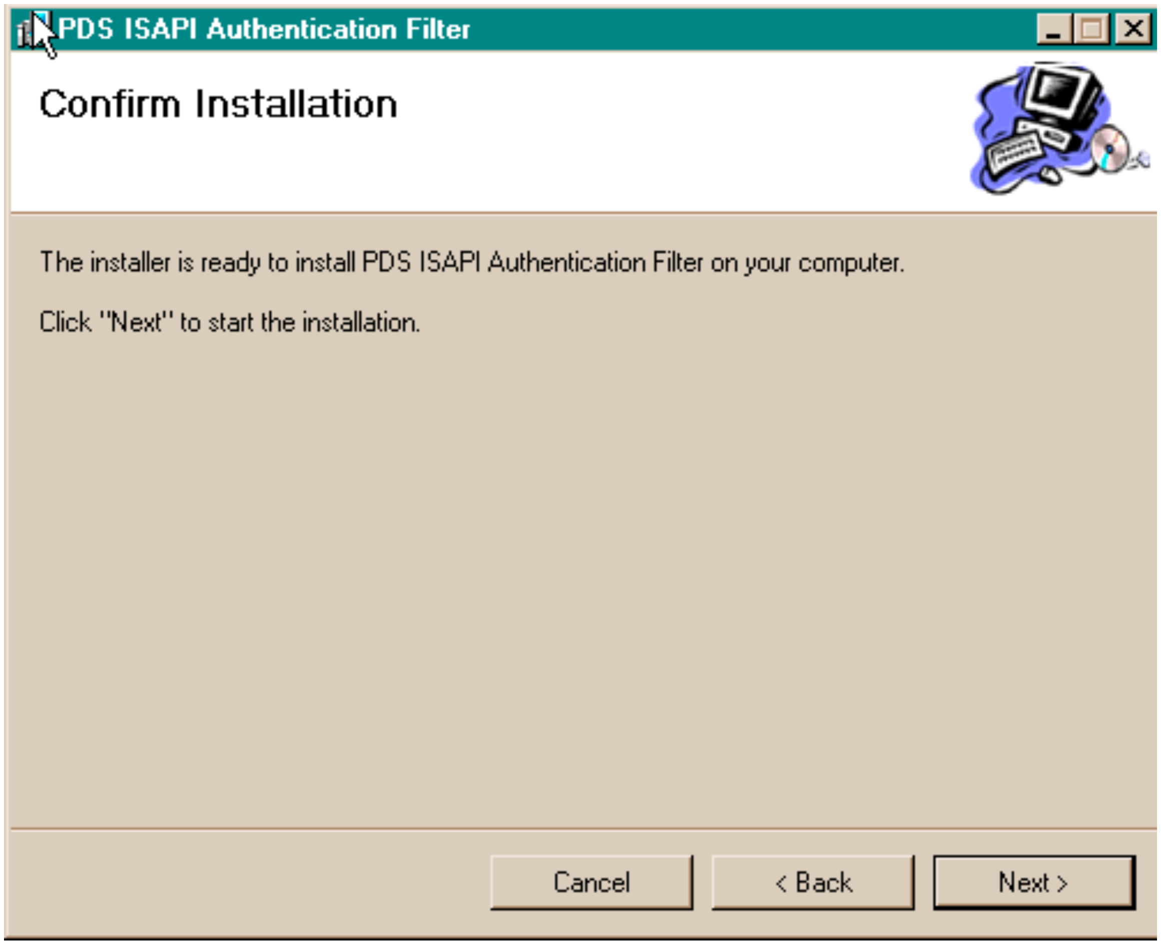
Local Cookie Name: nyulibrary_opensso_illiad

Local Cookie Domain: .library.nyu.edu

Local Cookie Path: /

Fig 2c - Fill in the fields as shown

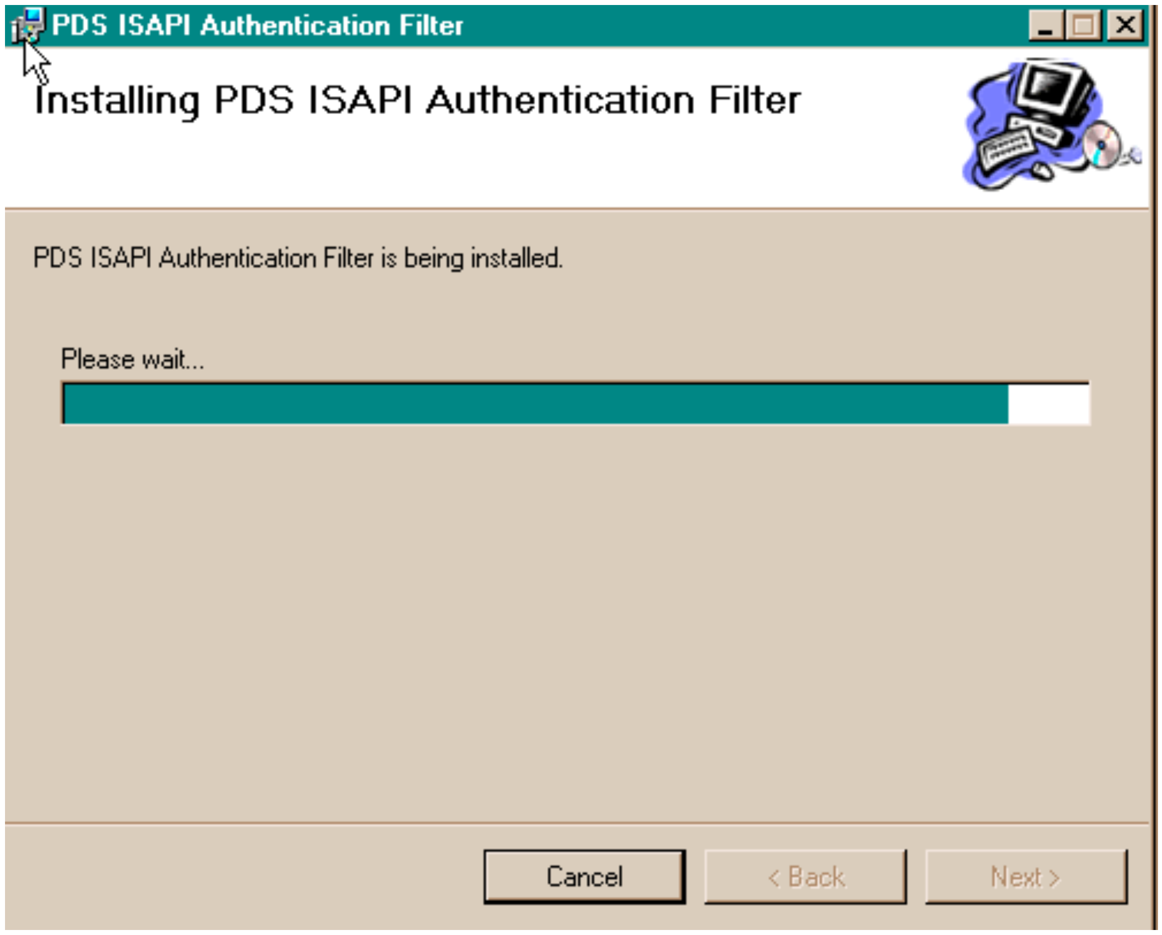
- Local Cookie Name: library_sso_illiad
- Local Cookie Domain: .library.university.edu
- Local Cookie Path: /



The installer is ready to install PDS ISAPI Authentication Filter on your computer.

Click "Next" to start the installation.

Fig 3 - Click next

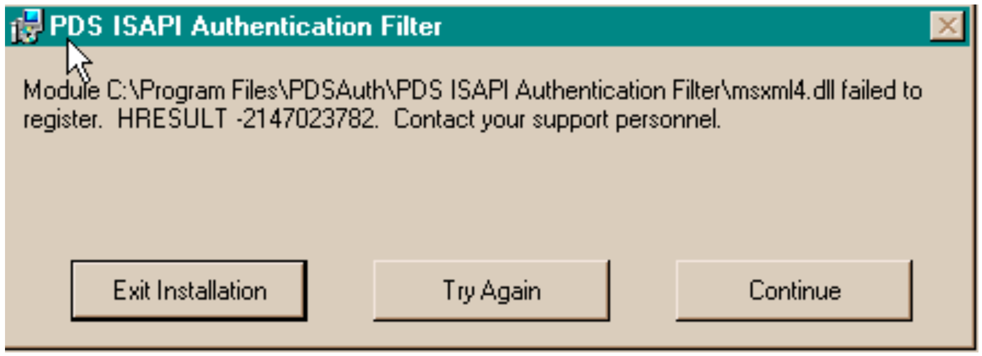

PDS ISAPI Authentication Filter
Installing PDS ISAPI Authentication Filter

PDS ISAPI Authentication Filter is being installed.

Please wait...

Fig 4 - Installing

You may see an error here, before canceling see *Known Issues* below



Module C:\Program Files\PDSAuth\PDS ISAPI Authentication Filter\msxml4.dll failed to register. HRESULT -2147023782. Contact your support personnel.

Install ISAPI filter

1. The above process should install the application on the local computer at *C:/Program Files/PDSAuth/PDSAuth ISAPI Authentication Filter/PDSAuth.dll* on 32-bit systems or *C:/Program Files (x86)/PDSAuth/PDSAuth ISAPI Authentication Filter/PDSAuth.dll* on 64-bit systems

   Note: The application was updated to run on 64-bit systems but is still itself a 32-bit application, so 64-bit systems run it in a compatibility mode.

2. Copy the new PDSAuth.dll file into the directory *C:\WINDOWS\system32\inetsrv* for 32-bit systems and into *C:\WINDOWS\SysWOW64\inetsrv*. WOW64 denotes Windows on Windows 64-bit. On 64-bit Windows, 32-bit applications are the exception, so even though system32 denotes 32-bit, it is the standard folder and is hence used for 64-bit applications on 64-bit Windows. The same is true for Program Files (x86), which denotes a 32-bit architecture explicitly.
3. Browse to the ISAPI Filters in IIS Management and add a new filter called "PDSAuth" pointing to the location at "C:\WINDOWS\system32\inetsrv\PDSAuth.dll" or "C:\WINDOWS\SysWOW64\inetsrv\PDSAuth.dll" depending on your architecture
4. Restart IIS and the filter should be activated.
5. Browse back to the ISAPI Filters management page: **In IIS 6.0** make sure the filter has an upwards pointing green arrow. If there is a downwards pointing red arrow, check below for troubleshooting; **In IIS 7.5** check the EventLog to see if there is an Application Error for PDSAuth.

Edit the Registry

1. Go to Start>Run and type in regedit
2. Go to File>Import in the Registry Editor menu
3. Choose the file PatronStatuses.reg included with the installer you downloaded, PDS Installer\PatronStatuses.reg.
4. This creates the list of valid patron statuses - **each downloaded instance of this application will need to edit these values to match their configuration**
5. Restart IIS after editing the Registry

Edit HTML files

After the ISAPI filter is setup on the illiad.dll you want to make sure that all ILLiad login requests redirect to the illiad.dll

1. In the Website root folder (/inetpub/wwwroot) create a file called index.htm redirecting to the illiad.dll.
2. In the ILLiad root folder (/inetpub/wwwroot/illiad) create a file called index.htm redirecting to the illiad.dll.
3. Backup the file /inetpub/wwwroot/illiad/Logon.html and edit the file to redirect to the illiad.dll.
4. The following source code can be used to create a meta tag redirect with a JavaScript backup.

```
<html> <head> <meta http-equiv="refresh"
content="0;url=http://ill.library.university.edu/illiad/illiad.dll"> <script
type="text/javascript"> <!-- window.location =
"http://ill.library.university.edu/illiad/illiad.dll" //--> </script>  </head> <body> <noscript> If
you are not automatically redirected, <a href="http://ill.library.
university.edu/illiad/illiad.dll">click here</a>. </noscript> </body> </html>
```

5. **Since this runs on an IIS server you could also make a Default.aspx page in these places with a server redirect**

Installation – From Source (not recommended)

Dependencies

Before attempting to compile you must have:

- A non-express version of Microsoft Visual C++, that is, a non-free version. It's a pain, but there are certain libraries (namely, atlbase) this application relies on to compile that are only provided in the paid versions of MS Studio.
- MSXML 4.0 Service Pack 2 or higher - http://www.microsoft.com/downloads/details.aspx?familyid=3144b72b-b4f2-46da-b4b6-c5d7485f2b42&displaylang=en

Compile/Install the DLL

1. On a Windows machine with Visual C++ Professional (or any non-Express edition of Visual C++) open the project solution in the root of the project: PDSAuth.sln
2. Make sure the project is in "Release" mode if you plan to use it in a production environment, or in "Debug" mode otherwise.
3. Do a clean build of the project and make sure it compiles (if it fails to compile see Troubleshooting section below).
4. In the root folder of the project there will be a Release/ directory which will contain a file called *PDSAuth.dll*, this is the executable ISAPI filter.
   Note: If you are deleting a previous version of the filter first,
   1. Go to Start>Administrative Tools>IIS Manager>Web Sites.
   2. Right click "Default Web Site" and select "Properties."
   3. Under the "ISAPI Filters" tab remove the filter titled "PDSAuth" and apply your changes.
   4. Close the management console, open a command prompt and type in "iisreset" to restart IIS. If you don't first do this the new ISAPI filter will not load because the old one is stored in memory.
5. If it exists, delete the previous PDSAuth.dll by browsing to C:\WINDOWS\system32\inetsrv and deleting the file PDSAuth.dll
6. Copy the new PDSAuth.dll file into the directory C:\WINDOWS\system32\inetsrv
7. Browse to the ISAPI Filters (following the above directions) and Add a new filter called "PDSAuth" and point the location to "C:\WINDOWS\system32\inetsrv\PDSAUth.dll"
8. Restart IIS and the filter should be activated.
9. Browse back to the ISAPI Filters tab and make sure the filter has an upwards pointing green arrow. If there is a downwards pointing red arrow, check below for troubleshooting.

Edit the Registry

The filter should load even if the registry keys are not there, but it will fail once you attempt to run it. Since there is no installer you must manually add the following registry keys: (load the registry editor by going to Start>Run and typing in "regedit")

- HKEY_LOCAL_MACHINE\SOFTWARE\PDSAuth or HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\PDSAuth
  - new String Value: AccessDenied = url of an external error page explaining why the user has been denied
  - new String Value: AccessDeniedByError = url of an external error page explaining why the user has been denied
  - new DWORD Value: CookieTimeout = 0 (no timeout for the local auth cookie)

    - new String Value: LoginURL = https://pds.library.university.edu/pds?func=load-login&institute=PRIMO&calling_system=illiad&url=
    - new String Value: PDSCookie = PDS_HANDLE
    - new String Value: ValidateURL = http://webdev1.library.university.edu/comon/https_wrapper.php?pds_handle=
    - new String Value: ApplicationPath = /illiad/illiad.dll
    - new String Value: ServiceURL = http://ill.library.university.edu
- HKEY_LOCAL_MACHIE\SOFTWARE\PDSAuth\PatronStatuses or HKEY_LOCAL_MACHIE\SOFTWARE\Wow6432Node\PDSAuth\PatronStatuses
    - new String Value: PatronStatus = one entry each for many patron codes corresponding to the following list (number codes only)
      01 Faculty
      02 Staff
      03 Undergraduates
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application\PDSAuth
    - new Expandable String Value: EventMessageFile = %SystemRoot%\system32\inetsrv\PDSAuth.dll
    - new DWORD Value: TypesSupported = 7 (decimal)

Restart IIS after editing the registry.

Project Files

Header files

- Logging.h
- Cookie.h
- Config.h
- PDS.h
- PDSAuth.h
- messages.h - defines the error messages used in the applications
- atlrx.h - a required dependency

C++ Files

- Logging.cpp - defines the functions that log events
- Cookie.cpp - defines the functions for creating, encrypting and decrypting the authentication cookie
- Config.cpp - defines the function for instantiating and subsequently freeing the registry values
- PDS.cpp - defines the functions for verifying a PDS connection and retrieving user info
- PDSAuth.cpp - contains the main DLL execute functions and lays out the logic for authentication
- PDSAuth.def - the definition file used for creating a .lib export file

Configuration Issues/Troubleshooting

First check the CASAuthN troubleshooting section to see if those answers help: https://confluence.ucdavis.edu/confluence/display/IETP/CAS+ISAPI+Client#CASISAPIClient-Troubleshooting

Can't compile

- **If you can't compile because of missing dependencies** make sure you have a non-express version of Visual C++, this will often occur with the error message complaining about the missing header file 'atlbase.h'

- **If you can't compile because of "unexpected end of file while looking for precompiled header"** right click the project in the project pane and open up the project properties. Under Configuration Properties>C/C++>Precompiled Headers set Create/Use precompiled headers to "Not Using Precompiled Headers"

- **If you can't compile because of a large amount of "Types pointed to are unrelated" and other type-related errors** right click the project in the project pane and open up the project properties. Under Configuration Properties>General set Character Set to "Use Multi-Byte Character Set"

- **Don't attempt to compile code on the local computer if it is mounted on a network drive.** Copy it to the local computer first.

Downward pointing red arrow in IIS 6.0

- **If you created a new project and copied the code over there are many VC++ properties that you need to set.** This is a tedious process but you should go through each one and compare them back to back with the working distribution. It is probably a Linker issues dealing with correctly linking the library with it's dependencies at runtime. Some key property values to check are:
    - Linker > Module Definition File = "PDSAuth.def"
    - Linker > Advanced > Import Library = "$(OutDir)/PDSAuth.lib"

  But there are others that could also cause problems and the only real way to be sure that this is not the issue is by going through each property and comparing it to the working copy.

- **Check if IIS 5.0 isolation mode is enabled.** I've gotten the application to work with this mode disabled. To enable/disable this go to Start>Administrative Tools>IIS Manager. Right click the "Web Sites" folder and select Properties. Navigate to the "Service" tab and make sure the check box that says "Run WWW Service in IIS 5.0 isolation mode" is unchecked.

- **If the ISAPI filter is enabled initially but then gets red arrow status after being run,** you probably have a registry error. Check your registry values as indicated above. Also, look at the Application EventLog for more runtime failure information. **Remember:** IIS needs to be restarted each time you edit the registry.

Illiad.dll prompts a download; or "Incorrect function" error

- **If either of these errors occur it is probably because the folder has incorrect executable privileges.** To check/fix this got to IIS Manager and right click the application folder under "Default Web Site." On the "Virtual Directory" (or "Directory" depending on what your application is setup as) tab make sure the "Execute Permissions" field is set to "Scripts and Executables." Restart IIS.

msxml4.dll failed to register

- You may see this error message after completing the install from the Installer. MSXML4 is a dependency of this program and so is packaged with the Installer. The inability to register message is probably just a permissions issue with either the location of the DLL or your user's specific permissions. Just click continue and install the filter as described above. If you see this error, chances are likely that MSXML4 is already installed anyway. Check in /WINDOWS/system32/ and /WINDOWS/system32/inetsrv for the file msxml4.dll or higher. Also you can look in your "Add/Remove Programs" in your Control Panel for MSXML4 SP2.
- **Running the installer actually produces an msxml4.dll in the same installation directory if necessary, but almost all current and future Windows versions have a later version anyway.**

Known Issues

Web timeout

The ILLiad customization manager defines a web timeout variable (currently set to 60 mins). The ILLiad application will automatically log a user out of the local system after this period, however it will not call the logout url defined by the RemoteAuth settings. So you will be logged out of ILLiad but not PDS. The simple logic would dictate to send you back to PDS since you're logged out, but since the PDS logout page was never called this will result in an infinite loop. I have added handling for this as a workaround for what seems like an ILLiad bug.

EventLog errors

The errors and messages in the Windows EventLog display incorrectly for this application. They claim that information about the error cannot be found:
"The description for Event ID (6) in Source (PDSAuth) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. You may be able to use the /AUXSOURCE= flag to retrieve this description; see Help and Support for details. The following information is part of the event: , ."

This error is supposed to indicate a registry problem with the files that produce the correct logging information but I have checked those values many times and they appear correct. I will revisit this issue, but it doesn't have any effect on the functionality of the application.

Unencrypted data sent to ILLiad from PDS

The application retrieves patron data from an AJAX call to PDS. ILLiad is on HTTP, PDS is on HTTPS. The Visual C++ functions the application is using does not allow cross-protocol scripting (adhering to the general AJAX security guidelines). Therefore, I have had to create a function that wraps the HTTPS call in an HTTP call. The ISAPI filter calls this non-secure application instead to retrieve. The result is unencrypted patron status and ID being sent.

We've determined that this is not a significant security threat to us since establishing a PDS session cannot establish an OpenSSO session. Possibility for the future would be to run ILLiad over HTTPS, but it's not clear if there are vendor limitations.

Upgrading to 64-bit Windows and IIS 7.5

There were a number of obstacles in upgrading to 64-bit Windows and IIS 7.5. There were two different sets of issues with the upgrade, one set stemmed from running a 32-bit application on a 64-bit OS, the other set stemmed from memory leaks in the application which surfaced because of changes in the way the OS processes the application buffer.

Installing the ISAPI Filter 32-bit on 64-bit

Installing a 32-bit application on 64-bit Windows installs into the folder Program Files (x86) instead of the standard “Program Files” directory and registry values install in the Wow3264Node subfolder.

32-bit ISAPI DLLs have to be housed inC:\Windows\SysWOW64\inetsrv as well. They will not run from System32 folder, which despite its name is for 64-bit applications in 64-bit Windows. SysWOW64 indicates Windows-on-Windows, that is attempting to run an application for a previous Windows version on this new 64-bit version.

Adding the filter to IIS 7.0

First make sure that ISAPI Filters have been installed as a component of IIS

IIS 7.0 has application pools, which are grouped applications or sites running under shared settings. For the ILLiad application pool turn on setting to ‘Enable 32-bit applications’ to allow this 32-bit ISAPI filter to run.

For added security, IIS 7.5 allows you to list DLLs that are allowed to run. Add ISAPI filter to this and set them to run as Application Pool Identity. This is not necessary if this DLL access list allows anonymous applications to run.

Memory Leaks

Once the 32-bit on 64-bit issue was resolved, the application began crashing with obscure EventLog errors. Microsoft services helped us locate the problem by setting up a crash dump rule on the PDSAuth.dll process. The crash dump pointed to memory leaks in the application. Using debug dialog tool we loaded the crash dump file, first compiling with the DEBUG symbol in Visual Studio (VS) and using the .pdb debug symbol file. The debug symbols are generated by building the VS project when in DEBUG mode. These symbols helped locate the general place in the source code where the memory leak was taking place. The full chain of emails with Microsoft Support is attached on the wiki page for this product. It provides detailed steps for setting up debug rules.

Adding custom status error messages

Custom errors must be added to the top level **and** the illiad site level. For the top level, a top-level script should be used. For the illiad site, the error script must live within that site, so at least under /illiad.

In IIS Manager under the custom errors screen the following lines were added to the associated status errors:

```
/common/error.asp?site=ill-library-university-edu&error=500&tail=1
/common/error.asp?site=ill-library-university-edu&error=404&tail=1
/common/error.asp?site=ill-library-university-edu&error=403&tail=1
```

Sub applications not requiring ISAPI Filter authentication

The path where the filter should intercept requests (ApplicationPath in the registry) needs to be /illiad/illiad.dll so that areas where separate login is required, such as /illiad/lending, will still work.

To filter most traffic through the main dll, add redirects to /illiad/illiad.dll from /, /illiad and /illiad/logon.html.